東海大学 2010 年度 卒業研究

アクティビティ図からのハードウェア動作の自動生成

指導教員 清水尚彦 教授

7ADF1109 山崎亮太
東海大学 情報理工学部 ソフトウェア開発工学科

平成２３年３月３０日

目 次

1 はじめに **5**
1.1 研究の背景 5

2 開発 **5**
2.1 自動合成の流れ 5
2.2 自動生成ツールの開発 6
2.2.1 モデリングツールの選定 7
2.2.2 モデリングツール 7
2.2.3 XMI ファイルの処理 9
2.2.4 XSLT を用いた解析 9
2.2.5 XSLT プロセッサ 10
2.2.6 JAVA 言語を用いた解析 10
2.2.7 DOM 10
2.3 NSLCOMP の開発 11
2.4 モデルコンパイラの動作検証 11

3 まとめ **12**

4 参考文献 **13**

図 目 次

1　UML を用いたハードウェア設計の流れ 6
2　par-while-if 12

表 目 次

1 EclipseUML の要素と属性 8
2 ArgoUML の要素と属性 9
3 DOM と SAX の比較 11

1 はじめに

1.1 研究の背景

近年，半導体技術の向上によって LSI（Large Scale IC）の集積度は増し，SoC（System On Chip）といった技術により半導体の開発の規模が拡大している．しかし，開発の規模が拡大する一方，製品の開発サイクルは短期間化する傾向にあり，開発者の負担となっている．

上記の問題を解決する 1 つの手段としてあるのが，ソフトウェア開発などで注目をされている UML（Unified Modeling Language）を用いた開発方法である．現在，システム全体を UML で設計することが多く，ソフトウェア開発では UML からの自動生成を行うなど，システム全体を通して一貫性のある設計手法が確立されてきた．しかし，ハードウェアは UML を用いた設計手法も確立されておらず，UML から自動生成を行う方法も数少ない．そこで，システム全体を UML を用いて設計開発を行うためにも，UML を用いたハードウェアの設計手法の確立が重要である．

我々は UML 図からハードウェア設計を自動で行うシステムの開発を目的とし，研究を進めている．過去には，クラス図からハードウェアの一部分（ハードウェアの静的な構造，構成）を自動生成をするシステムは過去に開発を行った [1]．しかしながら，論理回路を完成するためには，ハードウェア振る舞いの記述が必須である．そこで，我々は新しくハードウェアの振る舞いを自動生成するシステムを開発し，UML 図からの完全なハードウェア設計手法を開発した [2]．

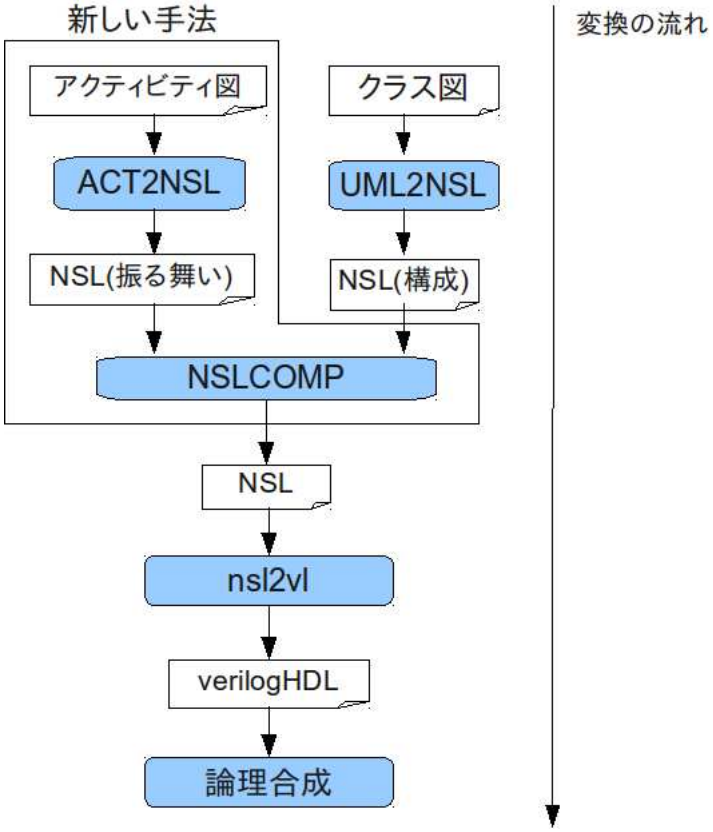
本研究では，内部動作を視覚的にとらえやすいアクティビティ図を用いてハードウェアの振る舞い，内部動作を示し，アクティビティ図からハードウェア記述言語の自動生成を行うアプローチを行った．当論文では，アクティビティ図からハードウェア記述言語への変換対応ルールに基づいて行ったモデルコンパイラの開発について示す．

2 開発

2.1 自動合成の流れ

図 3 は我々の開発したシステムを用いて，UML から論理合成を行うまでの流れを示している．まず，ハードウェアの構成を示したクラス図を作成．次にクラス図から UML2NSL を用いてハードウェアの構成を記述した NSL（Next Synthesis Language）[3] ファイルを生成．この時，生成される NSL ファイルはハードウェアの振る舞いの部分は記述されていないスケルトンコードとなる．次に，クラス図の構成に対応する動作，振る舞いを示したアクティビティ図を作成．次にアクティビティ図から ACT2NSL を用いてハードウェアの振る舞いを記述した NSL 言語記述ファイル [3] の生成を行う．次に UML2NSL と ACT2NSL から生成された NSL ファイルを NSLCOMP を用いて verilog（Verilog HDL）に変換可能な NSL ファイルを合成．その後，合成した NSL ファイルを nsl2vl を用いて verilog 記述に変換し，論理合成を行う流れとなる．

図 1: UML を用いたハードウェア設計の流れ



2.2 自動生成ツールの開発

我々が本研究において新たに開発したツールは ACT2NSL と NSLCOMP である．ACT2NSL とはアクティビティ図からハードウェアの構成を示す NSL 記述言語を生成するツールである．ハードウェアの内部動作を対応ルールに基づいてアクティビティ図でモデリングし，モデリングツールが生成した XMI（XML Metadata Interehange）ファイルを ACT2NSL に渡すことで自動的に NSL 記述ファイルを生成する．NSLCOMP はクラス図，アクティビティ図，それぞれから生成された NSL 記述を合成するツールである．クラス図，アクティビティ図，それぞれから出力された NSL 記述は NSLCOMP を使用することで自動的に NSL 記述の合成を行い，構成と振る舞いが記述された NSL 記述を出力する．これら２つのツールと，我々が以前に開発したクラス図からハードウェアの入出力端子，レジスタ，制御端子などの静的構造の NSL 記述を生成できる UML2NSL ツールをもって自動生成システムとした．以下に自動生成システムの開発段階で行ったアプローチを示す．

・アクティビティ図とハードウェアの振る舞い，内部動作の対応．
・アクティビティ図とハードウェア記述言語の対応（NSL 言語のみ）．
・アクティビティ図を作図するモデリングツールの選定．
・選定したモデリングツールから生成されるファイルの解析．
・UML ファイルを対応ルールに基づいて言語変換する手法の検証．

・対応ルールを用いた，ACT2NSL の実装.
・クラス図から生成した NSL 記述とアクティビティ図から生成した NSL 記述を合成する NSLCOMP の実装.
・実装したモデルコンパイラの動作検証.

また，アクティビティ図とハードウェアの振る舞い，内部動作の対応とハードウェア記述言語の対応は以前に明らかにしている [2] ため，本論文では示していない．時節以降は，アクティビティ図を作図するモデリングツールの選定，選定したモデリングツールから生成されるファイルの解析，UML ファイルを対応ルールに基づいて言語変換する手法の検証，対応ルールを用いた，ACT2NSL の実装，クラス図から生成した NSL 記述とアクティビティ図から生成した NSL 記述を合成する NSLCOMP の実装，実装したモデルコンパイラの動作検証と言う流れで示す.

2.2.1 モデリングツールの選定

ハードウェアの動作をアクティビティ図で示すため, アクティビティ図を作成できるモデリングツールを選定した. 下記は選定の条件である.

・アクティビティ図とハードウェアの振る舞い，内部動作の対応を作図でき，XMI ファイルを出力可能である.
・ツールの更新が停止していない
・ライセンスフリーである

一つ目の条件は，アクティビティ図の作図が行えるモデリングツールに限定するための条件である．二つ目の条件は，ソフトのアップデートが頻繁に行われており，今後のシステム変動に対応が行えるモデリングツールに限定するための条件である．三つ目の条件は，万人が利用できるモデリングツールに限定するための条件である.

条件を満たすツールを探した結果, 条件を全て満たすツールの発見には至らなかった．そこで以下のツールで段階的にアプローチを行い，システムの有効性を証明した.

・ライセンスフリーではないが, 作成したアクティビティ図とハードウェアの振る舞い，内部動作の対応ルールを全て表現できる EclipseUML.
・ライセンスフリーであるが, 我々の作成したアクティビティ図とハードウェアの振る舞い，内部動作の対応ルールに全ては対応していないが十分に表現可能な ArgoUML.

作成した対応関係が正しいことを内部動作の対応を全て表現できる EclipseUML で証明し, ライセンスフリーな ArgoUML で最終的な実装を行った.

2.2.2 モデリングツール

アクティビティ図は NSL 記述の対応ルールに基づいて EclipseUML 及び，ArgoUML で表記を行うことができる．EclipseUML は UML 図から XMI ファイルを自動的に生成する機能を有しており，アクティビティ図から XMI ファイルへ内部動作の自動変換が可能である．同じく ArgoUML

は UML 図から XMI ファイルを生成する機能を有しており，アクティビティ図から XMI ファイルへ内部動作の自動変換が可能である．

これら，EclipseUML と ArgoUML より生成された XMI（Extensible Markup Languge）ファイルを解析し NSL 記述へ変換を行った．表 1 は EclipseUML から出力される XMI ファイル”*.uml”の要素と属性の特徴をまとめたものである．表 2 は ArgoUML から出力される XMI ファイル”*.xml”の要素と属性の特徴をまとめたものである．

表 1: EclipseUML の要素と属性

要素・属性	データ
node 要素	アクティビティ図のノード要素
xmi:type	ノードの種類
xmi:id	ノードの固有 ID
name	ノードへの記述（16 進数表現）
outgoing	出力の配線 ID 2 つある場合半角スペースが入る
incoming	入力の配線 ID
sdge 要素	配線の要素
xmi:type	上記と同様
xmi:id	上記と同様
name	上記と同様
terget	出力先 ID
source	入力元の ID PackageImport 要素
PackagedElement	パッケージの要素 属性は type と id
eAnnottations	uad ファイルの参照

表 2: ArgoUML の要素と属性

要素・属性	データ
UML:*	アクティビティ図のノード要素
UML:StateVertex.incoming	入力の配線要素
UML:StateVertex.outgoing	出力の配線要素
xmi.idref	配線 ID 2つある場合半角スペースが入る
kind	ノードの種類
xmi.id	ノードの固有 ID
body	Guard 記述

2.2.3 XMI ファイルの処理

XMI とはモデルを，各種ツール，リポジトリ，ミドルウェア間で XML 文書によって交換するための標準規格である．我々は，モデリングツールから出力された XMI ファイルを NSL 記述ファイルへ変換を行うため，XML ファイルのスクリプト言語である XSLT（XSL Transformations）言語及び，汎用的処理が可能な JAVA 言語によるアプローチを行った [3].

2.2.4 XSLT を用いた解析

選定したツール EclipseUML と ArgoUML より生成された XMI ファイルを解析し NSL 記述へ変換を行うため，XSLT 言語によるアプローチを行った. 使用した XSLT プロセッサについては次節に記述する．XSLT 言語は XML 文書を他の XML 文書または，他の文書に変換するための簡易言語である. 記述は XML 文書の構造を別の形式に変形するための変換ルールを記述する. 記述された XSLT 文書は「スタイルシート」と呼ばれる．

XSLT 言語は XML 文書から HTML（HyperText Markup Language）文書やテキスト文書への変換などに使用される．XSLT 言語で作成できるスクリプトは XPath 構造に対応しており，現在作業を行っているカレントパスに対し，2 層ルートパスに近いパスに基づいた処理は扱うことができないことが分かった. このことから，今回扱う XMI ファイルの解析は困難であると判断し，XSLT 言語での解析を諦め JAVA 言語での解析を行った.

2.2.5 XSLT プロセッサ

XSLT 言語を扱い,XML によって記述された文書を他の XML 文書に変換するには XSLT プロセッサが必要である. 以下の項目は研究段階にて確認した XSLT プロセッサである．

・msxml
・XT
・LotusXSL
・XSL:P
・Saxon
・Xalan
・xsltproc

今回は，現在でも更新を行われておりサポート体勢が整っている，Saxon の XSLT プロセッサを使用した．

2.2.6 JAVA 言語を用いた解析

XSLT 言語での解析が困難なことから，JAVA 言語でのアプローチを行った.

JAVA の Version1.6 は XML 文書を扱うことのできる JAXP が標準化されており,JAXP では XML 文書を処理するため，ストリーム，DOM，SAX 用の三つのパッケージとこれらに共通する上位のパッケージを提供している．

XMI ファイルからアクティビティ図の要素等を読み込むため，DOM(Document Object Model) を使用した．DOM を使用した理由については次節に記述する．我々は DOM に読み込まれた要素が対応ルールに基づいて NSL 記述への変換がされるようプログラミングを行い，ACT2NSL モデルコンパイラを実装した．ACT2NSL によりコンパイルされた XMI ファイルは対応ルールに基づいた変換がされ，ハードウェアの内部動作が記述された NSL ファイルを生成する．JAVA 言語にて記述されたプログラミング行数は４１１行であり，JAVA の Version は”1.6.0_18”，使用した Package は”java.xml.parsers.*”，”org.w3c.dom.*”及び”java.io.*”である．

2.2.7 DOM

XMI ファイルを解析する API として SAX（Simple API for XML）と DOM と呼ばれる API がある．表２は DOM と SAX の違いをまとめたものである．我々は SAX ではなく DOM を使用することによって XMI ファイルの解析を行った.

DOM と SAX は競合するものではなく，長所と短所を併せ持っている．違いを把握したうえで，必要に応じて使い分けることで効率的な処理が行える．代表的な特徴として，SAX はイベント駆動型，順次アクセス方式であり，高速処理とメモリ消費量が少ない．DOM は対話，選択的処理であり，XMI ファイルを直接操作する感覚でプログラミングを行うことが可能である．

このことから，要素の順次参照での処理が可能であるなら SAX を使用することが望ましく，様々な要素を繰り返し参照し，対話的，選択的処理を行うには DOM が適していることがわかり，対象となる XMI ファイルの変換では様々な要素を繰り返し参照を行うため DOM を使用した.

表 3: DOM と SAX の比較

	DOM	SAX
アクセス	ランダム	順次
アクセスの タイミング	データを全部 読み込んだ後	データを読み込み ながらその都度
処理の方式	対話的，選択的処理	バッチ式
メモリーの占有	大	小

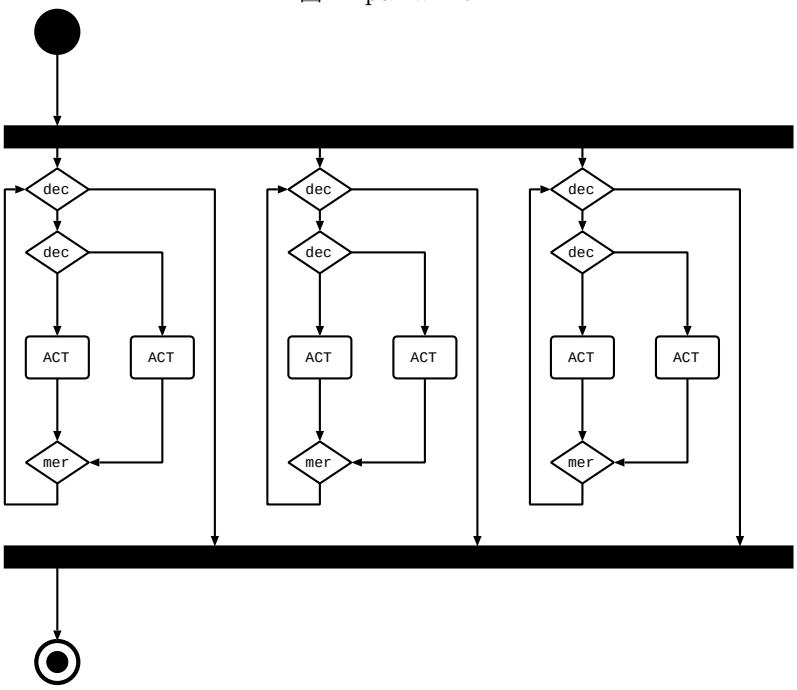
2.3 NSLCOMP の開発

NSLCOMP はクラス図，アクティビティ図，それぞれから生成された NSL 記述を合成するツールである．クラス図，アクティビティ図，それぞれから出力された NSL 記述は NSLCOMP を使用することで自動的に NSL 記述の合成を行い，構成と振る舞いが記述された NSL 記述を出力する. 従って NSLCOMP の行う操作はファイルの操作のみである．NSLCOMP は JAVA 言語でのプログラミング行数は８５行であり，JAVA の Version は”1.6.0_18”，使用した Package は標準入出力”java.io.*”である．

2.4 モデルコンパイラの動作検証

実装した ACT2NSL が我々の定めた対応ルールに基づき NSL 記述を生成する事を確認するため．EclipseUML 及び，ArgoUML モデリングツールにて 27 種類のルールの組み合わせの正常生成を確認した．図４は検証を行ったモデルの一例である．シーケンス動作に共通動作ブロックが入り，共通動作の中にループブロックが 3 つ入り，ループブロックに中には if ブロックが入る形である．この様に，seq ブロックに対し，par ブロック，while ブロック，if ブロックの組み合わせの 27 種類によるデバックを行い，ACT2NSL が我々の期待する NSL 記述を生成する事を確認した．なお，当設計手法の研究開発後，大規模なシステムの検証として 16bitCPU を UML 図で完全に表現し論理合成を行い，FPGA（Field-Programmable Gate Array）上での正常動作を確認済みである．

図 2: par-while-if



3 まとめ

本論文では，アクティビティ図からハードウェア動作を生成するツールの開発について示した．開発はアクティビティ図からハードウェア記述 NSL を生成する対応ルールに準じて行われ，我々は ACT2NSL 及び NSLCOMP を開発した．ACT2NSL は EclipseUML 及び，ArgoUML にて出力されるアクティビティ図のモデルファイルに対応しており，ハードウェア動作が記述された NSL 記述を出力する．NSLCOMP は ACT2NSL から出力された NSL 記述と UML2NSL から出力された NSL 記述を合成し，論理合成まで可能な NSL 記述ファイルを出力する．モデルコンパイラは seq ブロックに対し，par ブロック，while ブロック，if ブロックの 4 つのブロックの組み合わせの 27 種類によるデバックを行い，正常動作を確認．大規模なシステムとして 16bitCPU の正常動作を FPGA 上にて確認した．今後は，他の UML モデル図での自動生成手法を考案し，今回開発したアクティビティ図からの自動生成手法との併用，または使い分けを行うなど，UML 図からのハードウェア設計へのアプローチが行える．

4　参考文献

参考文献

[1] N.Shimizu, M. Ikura, W. Wiriya, S. Chivapreecha, "A New Logic Circuit Design Methodology with UML," The 24th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC 2009), pp.62-65, May. 2009.

[2] 上陰敏弘，山崎亮太，清水尚彦，" UML アクティビティ図のハードウェア記述言語 NSL への動作合成 "，コンピュータシステム研究会，2011.

[3] Overtone Corporation，http://www.overtone.co.jp/

[4] 浅海智晴，"XML/DOM Programming"，株式会社 秀和システム，pp.766，May.2001.

[5] 並木滋，清水尚彦，"UML からの論理合成可能 HDL の自動生成 合成ルールならびにモデルコンパイラ、HDL 生成試行" ，東海大学大学院 工学研究学科 情報通信制御システム工学専攻 修士論文，2007.

[6] 大槻博之，清水尚彦，"UML から SFL を自動合成するシステムの研究" ，東海大学大学院 工学研究学科 情報通信制御システム工学専攻 修士論文，2008.