1J−2　UDL/Iのセマンティクス定義に基づく処理系の試作

淡海功二　大高史嗣　安浦寛人　田丸啓吉
京都大学工学部

1　はじめに

現在、国内でハードウェア記述言語 UDL/I(Unified Design Language for Integrated circuit) の標準化作業が進められている [1]。UDL/I では、セマンティクスがシミュレータから分離して厳密に定義され、自然な意味づけを行なうために非決定性が導入されている。そのセマンティクスの定義手法は、UDL/I の記述を UDL/I のサブセットからなる核言語に変換し、この核言語に対して意味を厳密に定義することにより与えられるものである。非決定性は UDL/I の意味定義で重要な役割を果たしているが、処理系開発においては本質的な問題となる。現実的な処理系では、非決定性は何らかの決定的なアルゴリズムを用いて近似的に実現される。本稿では、UDL/I のセマンティクスを厳密に反映し、この近似手法を任意に与えることが可能な処理系を試作したので報告する。

2　UDL/Iのセマンティクス定義

UDL/I の開発の背景にはシミュレータと分離された厳密なセマンティクスの定義を与えるという基本方針がある。意味の与え方については、特に、意味の誤解が生じやすい動作記述セクションについて、UDL/I の本質的な記述能力を保存した核言語（コアサブセット）を定義し、UDL/I の各構文要素の意味をこの核言語によって記述する手法が採用されている。

核言語は、UDL/I の動作記述セクションの部分集合であり、本質的に 2 種の基本文からなる。1 つは論理関数を計算する組合せ回路に対応する **Terminal Statement** で、もう 1 つは記憶素子に対応する **Register Statement** である。UDL/I の記述は、これら 2 種の基本要素からなる回路網の動作を記述していると解釈される。UDL/I の各構文要素の意味は、等価な動作を表す核言語の記述で与えられる。

UDL/I の各構文ごとに独立に変換された核言語記述では、同一のファシリティ（ターミナルやレジスタ）に複数の代入が定義されたものとなる。この複数の代入は、条件が重なった場合に「信号の衝突」を引き起こす。UDL/I では、このような衝突を含む核言語を、最終的な動作を明確に表す「衝突のない」記述へ変換する規則を定めている。CR 関数と呼ばれる関数が、衝突の調停規則を指定する関数として導入されている。

UDL/I の意味定義に採用されている上記の手法は、言語の文法と比較的独立に意味を定義できる。ハードウェア設計言語のセマンティクスのもっとも微妙な部分は、核言語のしかも CR 関数を始めとするその一部に集約されている。よって、UDL/I の文法や核言語を大幅に変えることなく言語の微妙な意味を変更することができる。

3　処理系の概要

3.1　中間言語

本処理系では核言語に準拠した中間言語を設定した。UDL/I 記述はコンパイラにより、この中間言語に変換される。UDL/I 記述を中間言語に変換する目的は、シミュレーションアルゴリズムにひきずられることなく、UDL/I 記述の持つセマンティクスを明確にすることである。このことで、シミュレータと UDL/I のセマンティクスは完全に分離されたものとなる。

3.2　シミュレータ

UDL/I の意味定義、すなわち核言語の意味定義においては非決定性の概念が導入されている。具体的には、(1) 因果関係のないイベントの発生順序、(2) 最大 / 最小遅延、(3) 信号値の衝突、(4) 記述の不足による不確定値、に関して、基本的に非決定的な意味付けがなされている。しかし、(3) の中の同一クロックに対する信号衝突、および (4) に関しては、不定値 X を用いた決定的な意味付けが、処理系に依存しない形で行なわれている。その他については、非決定的な動作の解釈は処理系依存である。この形は本システムの中間言語にもそのまま反映されている。

ところが一般に、上記の処理系依存の非決定的な動作をすべてシミュレートするための計算複雑度は、NP-困難であることが示されている [5]。そのため、これらの意味を厳密に模擬するシミュレータは、現実的とはいえない。一般的に実用的なシミュレータでは、何らかの解釈に基づいた決定的なアルゴリズムを用いて、シミュレーションが行なわれていると考えられる。本システムでは、上記の非決定性に対する解釈を可変にすることを目的として、シミュレーションアルゴリズムを微妙な解釈に関係する部分と関係しない部分とに分離する手法を用いる。微妙な解釈に関係する部分は外部から定義されるため、回路の微妙な動作に関するアルゴリズムを任意に設定できる。逆に、アルゴリズムを記述しないシミュレータユーザの立場から見て、シミュレータの微妙な動きを知ることができる。また、回路の複雑さを限定すれば、非決定的な動作を完全にシミュレートするアルゴリズムを実現することも可能である。

Development of a UDL/I Simulation System based on Semantics Definition
Koji TANKAI, Fumitsugu OHTAKA, Hiroto YASUURA, Keikichi TAMARU
Kyoto University

4 UDL/I コンパイラ

4.1 内部処理

コンパイラの処理の各フェーズは以下のようになる。

1. 字句解析
   字句解析部は処理系への入力である UDL/I の記述を文法上の最小単位であるトークンに分離する。しかし、構文解析を行う以前に、完全なトークンに分離する事は困難である。従って、完全なトークンへの分離は、構文解析と並行して行う。
2. 構文解析
   構文解析部は、字句解析部によって分離された記述を、UDL/I の文法規則に従って完全なトークンに分離しながら、解析木を生成する。
3. 意味解析
   意味解析部は、文法規則には表れない意味上の解析を行う。
4. 変換
   変換部は、解析された結果をもとに、UDL/I の記述を中間言語に変換する。変換された結果に「衝突のある記述」がある場合には、さらに「衝突のない記述」に変換される。以下に衝突の解決例を示す [1]。

```
R := IF S1 THEN AT RISE(CK1) DO D1 END_DO END_IF;
R := IF S2 THEN AT RISE(CK1) DO D2 END_DO END_IF;
```

という核言語の記述は、レジスタ R に対する代入に衝突の可能性を含んでいる。この記述は、CR 関数を用いて

```
R := BEGIN
     IF S1 ! S2 THEN AT RISE(CK1) DO
          CR(D,IF(S1,D1,D),IF(S2,D2,D))
     END_DO END_IF;
     END;
```

のように「衝突のない記述」に変換される。

4.2 Prolog によるコーディング

構文解析を行う有力な手法として論理プログラミングがある。Prolog はこのような論理プログラミングに適した言語である。また、DCG (Definite Clause Grammar) 記法を用いることにより、文法規則を極めて明確に記述することができる [4]。このことは、高度で複雑な UDL/I の文法をインプリメントする場合に大きな利益となる。同時に、デバッグの効率も向上する。以上のような観点より、本システムは試作であることを考慮し、処理速度よりも開発効率を優先し、Prolog を用いてコーディングを行なった。

5 シミュレータの構成

シミュレータの概略構成を図 1 に示す。シミュレータの入力はコンパイラより出力される中間言語記述、初期状態、および入力波形記述である。この他にシミュレーションアルゴリズムも入力として与えられる。シミュレーション結果としての出力波形記述が出力となる。明確にされたセマンティクスにユーザ定義のシミュレーションアルゴリズムを付加させることにより、UDL/I 記述の持つ意味を変えることなく、種々のシミュレーションが実行可能である。

シミュレータの中心的なデータ構造である、タイムホイル (イベントリストも含む)、ファシリティテーブルの 2 つのデータ構造とユーザ間のインタフェイスとして、アクセス関数群が用意されている。これらのデータ構造とアクセス関数群をカーネルプログラムとして、シミュレーションアルゴリズムを記述したプログラムとリンクすることにより、実行形式のシミュレーションモジュールが生成される (図中点線部)。

ユーザ定義アルゴリズムの記述は、タイムホイルとファシリティテーブルへのアクセス関数を用いて行なわれる。中間言語の構成要素である、ターミナルとレジスタの 2 種類のファシリティの動作に対する記述が中心となる。中間言語がシンブルな形で与えられているため、比較的容易にアルゴリズムを記述できる。

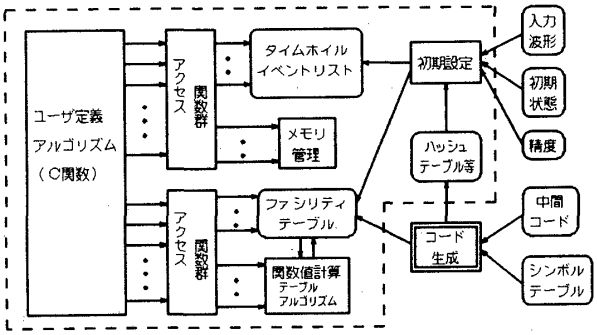

図 1. シミュレータの構成

6 あとがき

本稿では、UDL/I 処理系の試作に関して報告した。本処理系では、UDL/I のセマンティクス定義に基づいた手法を用いており、今後の標準化言語の処理系開発の一手法を示しているといえる。このシステムでは、UDL/I のセマンティクスにおいて処理系依存であるシミュレーションアルゴリズムの微妙な部分を分離し、外部から任意に与えることが可能なシミュレータを提案した。本シミュレータを用いてアルゴリズムを差し替えることにより、要求に応じた速度 / 精度を持つシミュレータが実現可能であると考えられる。

謝辞 日頃からご議論いただく本学石浦菜岐佐氏、松本裕治助教授、京都高度技術研究所神原弘之氏、横田吏司氏に感謝します。

参考文献

[1] LSI 設計用記述言語標準化委員会, "LSI 設計用記述言語仕様 第 1.0 版"(UDL/I マニュアル)

[2] Hiroto Yasuura and Nagisa Ishiura, "Formal Semantics of UDL/I and Its Applications to CAD/DA Tools", Proc.of ICCD, IEEE 1990

[3] 安浦寛人, "UDL/I のセマンティクス", VLSI 設計記述言語 UDL/I 講習会資料, 信学会, VLSI 設計技術研究専門委員会, 1991

[4] L.Sterling,E.Shapiro, "The Art of Prolog", MIT Press,1986

[5] 安浦寛人, 石浦菜岐佐, "ハザード検出問題の計算複雑さについて", 信学技報, COMP86-64(1986)