6E−5

論理型言語を用いた構造化分析法によるLSI仕様記述・検証法の検討*

長沼 次郎　松田 和浩　小倉 武†
NTT LSI研究所‡

1 はじめに

近年、LSIの大規模化、複雑化に伴い、その設計TATも増大している。設計TAT増大の一因は、LSI設計の流れの最上流に位置する仕様記述、理解の不完全性、困難性にある。このため、我々はLSI設計の短TAT化を図るため、構造化分析法に基づくLSI仕様記述、検証法の検討を進めている[1]。本稿では、抽象度の高い仕様を検証するため、動的検証用のプログラム言語として記号レベルのシミュレーションが可能な論理型言語Prologを用いることを検討した。仕様記述法にいくつかの制約を与えることにより、構造化分析手法による仕様記述からシミュレーション可能なProlog記述を生成し、これによる動的検証が可能であることを示す。

2 システム概要

2.1 システム概要

本システムの全体構成を図1に示す。本システムは構造化分析法を支援する既存の上流CASEツールであるSoft DA/SAツール[2]に動的検証ツールを付加した構成となっている。プロトタイプ記述用のプログラム言語として論理型言語Prolog[4]を用いる。

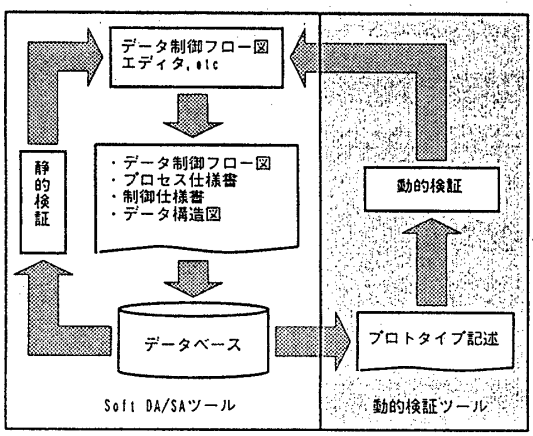


図1: 仕様記述・検証システムの全体構成

3 図形モデルの意味定義と仕様記述法

3.1 図形モデルの意味定義

構造化分析法ではシステムの振舞いを簡単なシンボルを使ったデータフロー図で階層的に記述する[3]。仕様記述に用いるプロセス、データフロー等の各図形モデルをProlog記述と対応させることにより、図形モデルの意味定義を行った。表1に、各図形モデルとProlog記述との対応を示す。

データフロー図のプロセスは述語に対応させる。プロセスの階層構造は節の呼び出し関係にマッピングされ、同一階層のプロセス群は、1階層うえの対応プロセスの述語をヘッド部にもつ節のボディ列を構成する。図2に概念図を示す。

データフローと制御フローは、各フローが接続しているプロセスに対応した述語の引数(変数)で表現する。ストアは、いわゆるレジスタとRAMに分離し、通常のストアの図形モデルを予約されたレジスタ述語に対応させ、RAMは予約されたRAM述語で表現する。プロセスの実行順序を規定する制御仕様書として、プロセス起動図あるいはプロセス起動表のいずれか一方を用いる。

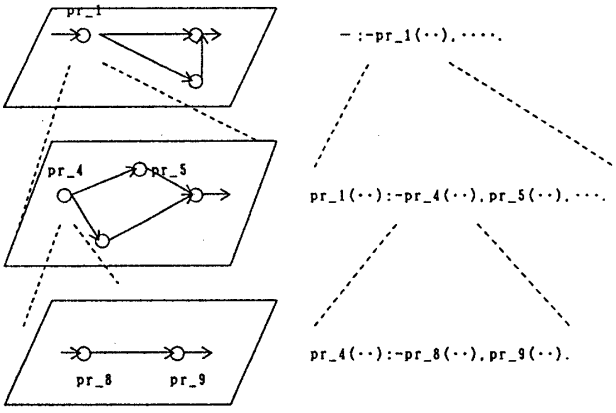


図2: プロセスと述語の階層構成

表1: 図形モデルとProlog記述の対応

図形モデル	対応するProlog記述
プロセス	述語に対応
データ、制御フロー	ボディ部の述語間の引数に対応
ストア	予約されたレジスタ述語に対応
制御仕様	ボディ部の述語の並びに対応

3.2 仕様記述法

基本的には構造化分析法に従い、図形モデルを用い

*High Level LSI Design Specification and Verification Based on Structured Analysis Method Using Logic Programming Language

†Jiro Naganuma, Kazuhiro Matsuda, Takeshi Ogura
‡NTT LSI Laboratories

て仕様をトップダウン的に記述するが、Prologでの動的検証を前提としているため、その記述法にいくつかの制約を与えている。

(1) プロセスの階層構成に関する制約

同一階層のプロセス群は、レジスタプロセス、RAMプロセスを含めてその実行順序がプロセス起動図あるいはプロセス起動表で記述可能なものに限る。

(2) 記憶要素に関する制約

記憶要素としては、レジスタ、RAMのみを用いる。データの入出力に伴うレジスタプロセス、RAMプロセスの起動を制御仕様書に陽に記述する。

(3) 制御データの生成に関する制約

内部で生成する制御データは、陽にプロセスで生成する。制御仕様書では制御データの加工を記述しない。

(4) プロセス仕様書に関する制約

ボトムプロセスのプロセス仕様書は、以下の形式でPrologで記述する。

ヘッド述語名	:当該プロセス名
ヘッド述語引数	:
第1引数	:動作前の内部状態変数
第2引数	:当該プロセスへの入力データ
第3引数	:当該プロセスからの出力データ
第4引数	:動作後の内部状態変数
ボディ述語	:動作の順序に従って記述する。

4 Prolog生成規則と生成実験

4.1 Prolog生成規則

仕様記述から動的検証のためのProlog記述を生成する規則を示す。Prolog記述の生成はボトムレベルから上位に向けて行う。

(1) レジスタプロセス、RAMプロセスの記述生成

レジスタプロセス、RAMプロセスに対しては予約されたボトムレベル記述を生成する。

(2) 階層内プロセス間接続の記述生成

同一階層内のすべてのプロセスのProlog記述が生成されたのち、当該階層のプロセス間接続のProlog記述を生成する。

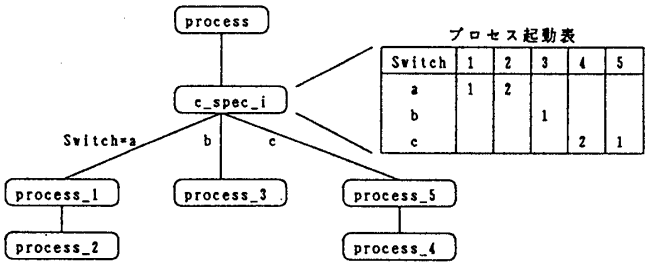
当該階層の制御仕様書がプロセス起動図で定義されている場合、最初に起動されるプロセスから分岐が存在するプロセスまでのプロセスに対応するProlog記述のヘッド述語をボディ述語として順次列記して引数の対応関係の整合性をとり、最後に分岐を表現する述語(c_spec)を生成しさらに、c_specをヘッド述語とするProlog記述を生成する.

当該階層の制御仕様書がプロセス起動表で定義されている場合は、ボディ述語として、分岐を表現する述語(c_spec)を生成し、さらに、当該階層のプロセスのProlog記述とプロセス起動表から、c_specをヘッド述語とするProlog記述を生成する。プロセス起動表の場合の記述生成例を図3に示す。

4.2 8ビットCPUを用いた生成実験

仕様記述・Prolog生成実験の対象として、命令セットとその動作が定義されている簡単な8ビットCPUをとりあげた[5]。本CPUの仕様をSoft DA/SAツールで記述し、先に示したProlog生成規則に従ってハンド・コーディングによりProlog記述に変換した。仕様は最大5階層で計57個のプロセスで記述できた。生成したProlog記述は約500行で、宣言節、規則節の数で約110個であった。

この記述を用いて、命令セットの記号レベルでのシミュレーションを行った。シミュレーション速度は、ワークステーション上のProlog処理系(100KLIPS程度)で、約500IPS(Instruction Per Second)程度である。



(a) プロセス起動表

```
process(···,Switch,···):-c_spec_i(···,Switch,···).
c_spec_i(···,a,···):-process_1(···),prpcess_2(···).
同値関係記述 .
c_spec_i(···,b,···):-process_3(···).
同値関係記述 .
c_spec_i(···,c,···):-process_5(···),prpcess_4(···).
同値関係記述 .
```

(b) 生成されるProlog記述

図3: プロセス起動表の場合の記述生成例

5 おわりに

LSI仕様を対象とした仕様記述・検証法に関し、抽象度の高い仕様を検証するため、動的検証用のプログラム言語として論理型言語Prologを用いることを検討した。仕様記述法にいくつかの制約を与えることにより、構造化分析法による仕様記述から記号シミュレーションが可能なProlog記述を生成し、これによる動的検証が可能であることを示した。今後、仕様記述法の検討を深めるとともに、Prolog記述の自動生成、並列動作を含む仕様の検証を行うための並列論理型言語[6]への展開等を検討する。

参考文献

[1] 松田, 小倉, "LSI仕様記述と仕様検証の一手法について," 情処学会DAシンポジウム'91, 論文集, pp.33-36, Aug. 1991.

[2] 磯田他, "設計情報とコードの一体管理方式に基づくソフトウェア開発支援システム(Soft DA/SA)," NTT R&D, Vol.38, No.11, 1989.

[3] D.J.Hatley and I.A.Pirbhai, "リアルタイムシステムの構造化分析," 日経BP社, 立田監訳, 1989.

[4] W.F.Clocksin and C.S.Mellish, "Programming in Prolog," Springer-Verlag, 1981.

[5] "論理設計CADに関する調査報告書," 日本電子工業振興協会, 1986.

[6] K.Ueda, "Guarded Horn Clauses," ICOT Tech. Rep. TR-103, July 1985.