5B-9

設計作用知識の記述レベルに関する評価

岩本雅彦　渡辺正信
日本電気(株) C&C システム研究所

1 はじめに

エキスパートシステム構築時の課題のひとつは「知識を記述する抽象化レベル、粒度をどう設定するか?」ということである。例えば、lisp 等の汎用言語を用いる場合もあれば、より抽象的なレベルとより大きな粒度の記述単位を設定して、その上で知識を記述する場合もある。この設定の判断基準として一般的には次の二つが重要である。(1) 記述能力が十分であること、(2) 知識の記述が容易で、理解しやすいものであること、である。本稿では、設計型システムにおける設計作用知識 (ルール) の最適な記述レベルについて述べる。特に設計型システムでは加えて、(3) 各設計作用記述において論理正当性 [1] を保存すること、が重要であることを示し、そのための適切なレベルについて考察する。最後にフルカスタム LSI 論理合成エキスパートシステム EXLOG[2] での知識記述レベルを以上の 3 点から評価する [3]。

2 設計作用ルール

設計とは機能仕様を与え、それを実現、実行可能なものに変換する過程である。このための知識としては、設計作用、その制御、部品等に関するものが必要となる。この中で特に設計作用知識は直接的な変換を実行するためのもので中心的な役割を果たす。以下、設計作用知識に絞りその記述レベルを検討する。

設計作用知識はある条件を満たす設計対象の一部をとらえ、そこに変換操作を加えるものであり、ルールとして表現されることが多い。ルールの LHS(条件部) では、要求仕様、中間状態に対するパターンマッチング、ルールの RHS(作用部) では設計対象に対する作用が主として記述される。

各ルールの粒度は適用、設計対象に対する作用が論理正当性を保存する程度に大きいことが望ましい。最終的な設計結果が満たすべき幾つかの条件の中で最も基本的なものは、機能仕様を満足していることである。これを保証するためにはルール適用前後での論理不整合がないのが望ましい。ルールが論理正当性を保存しない場合には、ルールの適用順を意識しなければならず、ルール間の独立性を保持できなくなる。

Evaluation on the Representation for Design Operators
Masahiko IWAMOTO, Masanobu WATANABE
NEC Corporation

表 1: 設計作用知識の記述レベル

抽象化レベル	例
設計操作	(decompose ...) (instantiate-macro ...)
設計構造	(new-node ...) (connect ...)
オブジェクト構造 (フレーム言語)	(create-object ...) (put-value ...)
リスト構造 (汎用言語)	(setf ...) (cons ...)

3 作用の記述レベル

ルールの RHS は作用節の並びで構成される。表 1は RHS の各節での作用記述の抽象化レベルを示したものである。設計分野固有の概念にまで抽象化されていることが、理解を容易にする上で必要である。また、過度に抽象化されている場合には、個々の記述単位が汎用性、柔軟性を失い、記述能力が不十分となる可能性がある。

また、各節の粒度も論理正当性が保存されるだけの大きさを持っていることが望ましい。ルールの適用順が不確定であるのと異なり、作用節の並びの途中で実行が中断されることはない。従って単に設計結果を保証するだけであれば作用節の並び全体として、つまりルール単位で整合性がとれていればよいということになる。しかし、各節の実行間においても論理整合性が保存されるならば、ルール記述の仕方によって不整合が生じる可能性がなくなる。従って、ルール記述者は整合を意識することが不要となると共に、ルール作成者間での記述の差がなくなり、ルールを互いに理解しやすくなる。

4 フルカスタム LSI 論理合成のための変換モデル

フルカスタム設計ではセミカスタム設計と比較し、複雑な処理を行なう。例えば、設計の過程の中でマクロを定義したり、トランジスタの特性を生かした設計を行なったりする。EXLOG では以上のフルカスタム固有の設計手法に対応するものも含め 16 種の記述単位 (作用素) を用意した。これらは変換モデルという観点からは仕様記述変換 (rewrite 等)、併合 (merge 等)、分解 (decompose 等)、置換 (instantiate-macro 等)、追加 / 削除 (insert-node 等) に分類される (図 1)。

このモデルを記述能力、記述容易性、論理正当性保存変換の観点から評価する。

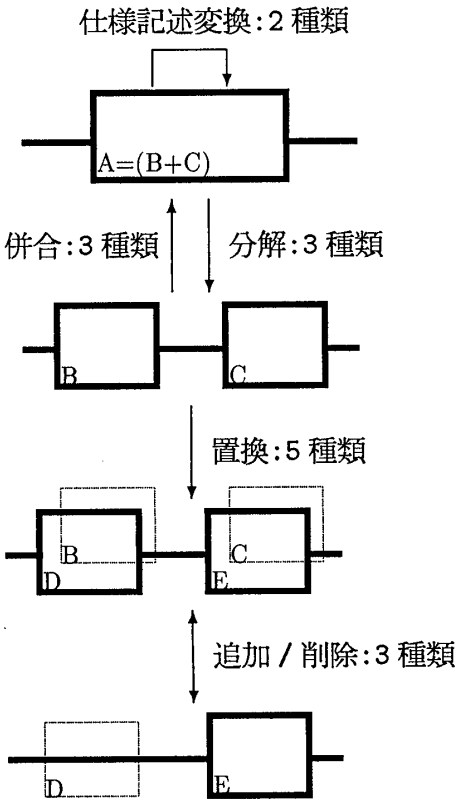

図 1: フルカスタム LSI 論理合成の変換モデル

(1) 記述能力: 16 種の作用素とルール変数の束縛 (bind)、タスク制御 (register-task) に関するもので、ほぼ全ルールをカバーし、十分なセットであることが示された。一部対応できなかったルールはより低レベル (構造操作レベル) の関数を使って記述している。合成される回路も回路規模の点では人手と同等の回路となっている。

(2) 記述容易性: 構造操作レベルで記述したルール (EXLOG のプロトタイプのルール) の RHS の平均行数が 16.8 行であったのに対して、設計操作レベルで記述した EXLOG 実用版でのものは 4.2 行と 1/4 になっている。概念的にも抽象化され、よりわかりやすくなっている。これ以上に抽象化した場合 (例えば ALU を合成する等) には、汎用性が失われ記述能力が損なわれると思われる。図 2は加算演算部分を加算器として取り出すルールを比較したものである。

(3) 論理正当性の保存: 併合、分解を行なうものは論理正当性を保存する。また、仕様記述変換、置換を行なうものに関しては、変換前後の仕様、モジュール機能の同一性が証明されれば、操作自体も正当なものとなる。追加 / 削除を行なうものは通常、論理正当性保存変換を行なわない。しかし、バッファとしてのインバータの挿入は必ずペアでなされること等からルール全体として保証することは難しくない。

```
(defrule r-decompose-adder
  (there-exists ?reg (i* register)
    (eq (the-status-of ^reg) 'refining)
    (match
      (specifications ^reg)
      {(@has (@and (add ? ?) ?add-op))}))
  ->
  (bind ?adder (new-node adder ^reg))
  (bind ?adder-out
        (new-path (width ^add-op)
                  ^adder ^reg 'data))
  (put-spec ^adder ^adder-out ^add-op)
  (dolist (path (in-paths ^adder))
  (connect path ^adder 'data))
  (bind ?before-in-paths (in-paths ^reg))
  (subst-spec ^adder-out ^add-op ^reg)
  (dolist (path (path-diff ^before-in-path
                           (in-paths ^reg)))
  (disconnect path ^reg)))
```

(a) 構造操作レベルの記述

```
(defrule r-decompose-adder
  <上記ルールの条件部に同じ>
  ->
  ; reg ノードの add-op 仕様部分を adder として
  ; 取り出す
  (decompose ^reg ^add-op adder))
```

(b) 設計操作レベルの記述

?...: 変数　　^...: 変数の参照

図 2: 記述レベルによるルール比較

5 おわりに

設計作用知識を記述するレベル (抽象度と粒度) を適切に設定することが、記述能力、記述容易性を増し、論理正当性保存変換を保証する上で重要であることを示し、さらに EXLOG での記述レベルがこの 3 点を満たしていることを示した。今後、より高品質の回路を合成するために制御知識の記述法の検討を進める予定である。

謝辞

最後に, 貴重な御助言を頂いた超 LSICAD 技術本部晴山部長, 黒部課長, 前田主任、河原林氏、及び C&C システム研究所大野部長, 小池部長代理に深謝します.

参考文献

[1] Mostow, J.: *Toward Better Models Of The Design Process*, AI Magazine 6, Spring, pp.44-57 (1985).

[2] Watanabe, M. et al.: *EXLOG: An Expert System for Logic Synthesis in Full-Custom VLSI Design*, in Knowledge Based Expert Systems in Engineering: Planning and Design, pp.315-329, Computational Mechanics (1987).

[3] 岩本, 渡辺, 河原林, 前田: フルカスタム *LSI* 論理合成エキスパートシステム *EXLOG* における設計過程モデル, 信学論, Vol.J72-A, No.8 掲載予定 (1989).