6M－2　設計支援ツールの統合化環境ＤＡＴＥにおける設計知識構造

新井　浩志[*1]　長谷川　拓己[*2]　深澤　良彰[*1]　門倉　敏夫[*1]
[*1]早稲田大学理工学部　[*2]日本電気㈱

１．はじめに

ハードウェア設計者の知的作業負荷を軽減するための設計支援環境として、設計知識を環境に取り入れ、それに基づいて設計を支援する手法を提案する。ここでは、その設計知識構造を中心に述べる。

本環境ＤＡＴＥ(Design Automation Tools integration Environment)では、ハードウェア設計者に要求される設計知識の量を減らすことにより、複雑化・多様化するハードウェア設計作業を効率化する。本環境では、設計知識を、一般のハードウェア設計者に共通の知識と、各設計支援ツールを利用するための知識などに分類して保持する。ハードウェア設計者は、共通の設計知識の概念を用いて設計要求を指示する。これに応じて本環境は、各設計支援ツールを統括してハードウェア設計作業を進める。

２．統合化設計支援環境とドメイン間の知識変換

設計支援ツールの複雑化・多様化にともない、適切な順序で、適切な設計ツールに、適切なデータを入力するための知的作業負荷が増大してきている。そこで、複数の設計支援ツールを使ってハードウェアを効率良く設計するための、設計支援環境が多く開発されている(1)-(3)。そこでは設計手法、設計支援ツールの機能、設計データ、およびそれらの相互関係をルールまたはスクリプトで表わし、エキスパートシステムによって処理するという形態をとっているものが多い。すなわち、ハードウェア設計に必要な知識をまとめて知識ベースとして表現し、設計支援ツールの実行と設計データの管理を自動化することを目的としている。

しかし、これらの設計支援環境では、ユーザがドメイン間の設計知識の変換をしなければならない。ここでドメインとは、ある範囲で閉じた設計知識の集合である。ハードウェア設計者は、それぞれ固有のドメインの設計知識を持っている。また、設計支援ツールの設計者も、特定のドメインの設計知識をもとに設計支援ツールを開発している。すなわち、同一の論理回路を表現するときにも、ゲートのタイプ設定、論理値の種別、遅延などのモデル化の方法がドメインによって異なっている。このため、ハードウェア設計者は常時ドメインごとの知識の対応を考慮しながら、設計作業を進める必要がある。すなわち、従来の設計支援環境では、設計作業の自動化に重点が置かれており、ハードウェア設計者とのインターフェースが十分考慮されていなかった。

これに対し設計支援環境ＤＡＴＥでは、ハードウェア設計作業に必要な知識を、共通のドメインの知識と、各設計支援ツールのドメインの知識に分離して記述し、その対応を定義する。ＤＡＴＥでは、このドメイン間の設計知識変換を自動化することにより、ハードウェア設計者が、設計支援ツールを利用するための知識の量と、設計支援ツールや環境と対話するための作業の量を軽減している。

３．ＤＡＴＥのアーキテクチャ

図１にＤＡＴＥのアーキテクチャを示す。

本環境の利用に当っては、環境設計者が関連する技術者の知識を統括して、共通ドメインの知識を構築する。設計支援ツールの設計者は、共通ドメインの知識との対応を考慮し、各設計支援ツールの機能、仕様を記述する。ハードウェア設計者は、共通ドメインの知識を用いて、設計要求の記述、環境・設計支援ツールとの対話をおこなう。

知識ベースは共通ドメイン、設計支援ツールドメイン、および設計戦略セグメントから構成され、知識マネージャを通して入力・管理される。設計戦略プランナは、設計要求に対して、設計戦略セグメントをもとに設計プランを構築し、ツール・マネージャに対して設計指示を与える。ツール・

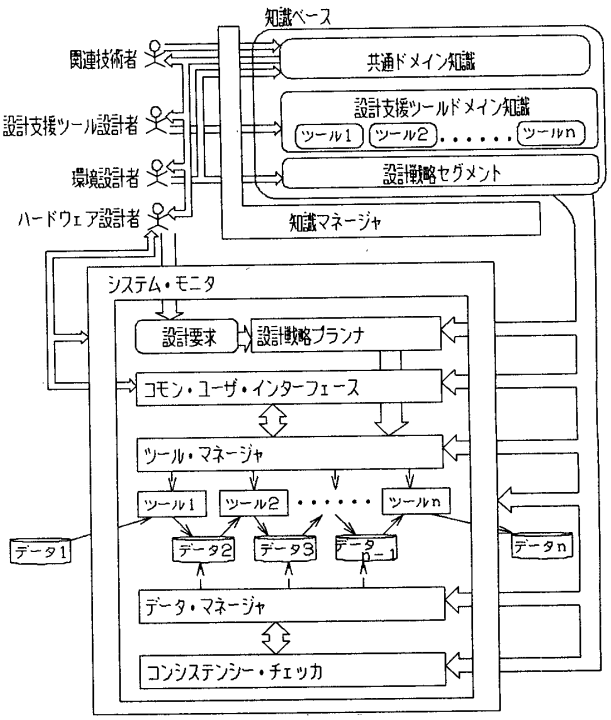

図１：設計支援環境ＤＡＴＥ

Design Knowledge Structure in the Design Environment "DATE" to Integrate Design Automation Tools

Hiroshi ARAI[*1], Takumi HASEGAWA[*2], Yoshiaki FUKAZAWA[*1], Toshio KADOKURA[*1]
[*1]Waseda University, [*2]NEC Corporation

マネージャは、コモン・ユーザ・インターフェースを通してハードウェア設計者と対話しながら、各設計支援ツールの実行を制御する。データ・マネージャは、各設計支援ツールの入出力データの仕様をもとに設計データのバージョン管理、設計空間管理をおこなう。コンシステンシー・チェッカは設計データのドメインの対応関係をチェックする。システム・モニタは以上の機能を統括して設計作業を進める。

4．知識ベース

知識ベースは以下の項目から構成される。

(1) 共通ドメイン
　設計対象ハードウェアの機能と構造に関する知識を保持する。また、関連する技術の知識なども含める。ハードウェア設計者は本知識だけを用いて環境と対話し、設計作業を進める。図2は、論理ゲートで構成される論理回路を記述している例である。

(2) 設計支援ツールドメイン
　各設計支援ツールのドメインの知識を保持し、共通ドメインの知識との対応を記述する。ここには設計支援ツールの機能とその入出力データのシンタックスが記述される。図3では、ある論理シミュレータの入力となる回路記述ファイルの仕様が記述されている。

(3) 設計戦略セグメント
　設計戦略プランナが各設計支援ツールの実行制御プランを構築するための、部分的設計手順を記述する。また、ある設計ツールの実行が失敗した場合の再設計手順を記述する。

ここで問題となるのは共通ドメインに記述される知識の質と量であり、これによって環境としての機能が決定される。たとえば、入出力設計データの内部知識まで記述されていれば、設計データのコンシステンシーをチェックすることが可能である。また、詳細で柔軟なモデル化が行なわれていれば、設計支援ツールドメインの知識の記述は容易であり、設計支援ツールの追加、機能変更へも容易に対応できる。しかし、一般には、ハードウェア設計に関する知識をすべて共通ドメインに記述することは困難である。したがって、環境は、部分的な知識でもそれだけで閉じていれば、記述された内容に応じて機能することが要求される。

```
$Logic_Circuit  :=  has_a $Circuit_Function ,
                    has_some $Element ,
                    has_some $Connection ,
                    has_some $Terminal ;
    ⋮                    ⋮
$Element        :=  is_a $Gates ;
$Gates          :=  is_a ($Or_Gate or_a $And_Gate
                                  or_a $Not_Gate) ;
$Or_Gate        :=  has_a $Logic_Function $Logic_Or ;
    ⋮                    ⋮
```

図2：共通ドメインの知識記述例

```
tool_name       :=  logsim ;
execution       :=  'logsim' Circuit Pattern ;
input           :=  Circuit_File ,
                    is_a $Logic_Circuit ;
    ⋮                    ⋮
Circuit_File    :=  is_a Text_File ,
                    has_a File_Name := Circuit '.CCT',
                    has_a Circuit_File_Syntax ;
Circuit         :=  is_a string ,
                    has_a format [A-Za-z][A-Za-z0-9]₁'¹⁵ ,
                    is_a $Circuit_Name ;
Circuit_File_Syntax
                :=  Node_Def_Blk Input_Def_Blk
                                 Output_Def_Blk Net_Def_Blk ;
Node_Def_Blk    :=  'NODE' ';' [Node_Def]₁^ ;
Node_Def        :=  Node_Name Node_Type ';' ,
                    is_a $Element ;
Node_Name       :=  [A-Za-z]₀'¹⁶ ,
                    is_a name_of $Element ;
Node_Type       :=  is_a  (NOT or_a AND or_a OR ) ;
OR              :=  is_a $Or_Gate ;
    ⋮                    ⋮
```

図3：設計支援ツールドメインの知識記述例

5．知識記述量と評価

現在、設計知識の記述、表現方法を検討している。具体的には、論理シミュレータと論理最適化の2つの設計支援ツールで構成される環境の記述を試み、表1の記述量で記述可能であることを確認している。

表1：設計知識記述例

		記述A	記述B
記述量	共通ﾄﾞﾒｲﾝ	7ｽﾃｯﾌﾟ	80ｽﾃｯﾌﾟ
	設計ﾂｰﾙﾄﾞﾒｲﾝ	55ｽﾃｯﾌﾟ	120ｽﾃｯﾌﾟ
記述最小単位		論理回路	論理ｹﾞｰﾄ
環境の機能	自動実行	可	可
	ｺﾝｼｽﾃﾝｼｰﾁｪｯｸ	不可	可

6．まとめ

共通ドメインの知識と設計支援ツールドメインの知識を分割して記述することにより、設計者が必要とする知識量を減らす手法を提案した。ハードウェア設計者は、共通ドメインの知識だけで環境とのインターフェースをとることが可能である。

今後は、論理設計支援ツールを当面の対象として考え、本環境の各機能の詳細な機能設計・開発をおこない、実際に論理設計支援環境を構築してゆく予定である。特に対話型設計支援ツールをコモン・ユーザ・インターフェースでいかにサポートするかが今後の研究課題である。

将来的には、各設計支援ツールドメインの知識をもとに、データ・コンバータの自動生成への応用も可能であると考えている。

［参考文献］

(1) "VLSI CAD Tool Integration Using the Ulysses Environment", M.L.Bushnell, S.W. Director, IEEE 23rd D.A.Conf., 1986

(2) "A Monitor for Complex CAD Systems", A.D. Janni, IEEE 23rd D.A.Conf., 1986

(3) "A Design Utility Manager: The ADAM Planning Engine", D.W.Knapp, A.C.Parker,IEEE 23rd D.A.Conf., 1986