設計自動化 55－15
(1990.12.13)

設計支援環境ＤＡＴＥにおける ツール更新作業負荷の軽減機構

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㈱

あらまし　設計支援環境において，設計支援ツールを組込む作業，更新する作業の負荷を軽減する機構を提案する。本設計支援環境 DATE では，環境全体で統合化された一般的設計知識と，個々の設計支援ツールに関する設計知識とを分離して記述する。そして，それら２種類の設計知識の間の変換機構を設計支援環境に付加している。これにより，個々のツールに固有な設計知識の管理を一元化し，設計支援環境の各種機能を，この設計知識から切り離している。このため，設計支援ツールの更新作業は，そのツールに固有な設計知識の更新作業と，環境全体で統合化された設計知識に対する更新作業に分離され，ツール更新時の環境への影響を最小限におさえることが可能である。本報告では，設計支援環境の機能の例として，ツールに共通なユーザインターフェースを提供する機能について報告するとともに，本機構により環境の更新作業負荷が軽減されることを示す。

Modifiability Improvement in a Design Environment DATE

Hiroshi ARAI*1, Takumi HASEGAWA*2, Yoshiaki FUKAZAWA*1, Toshio KADOKURA*1

*1 Waseda University, *2 NEC Corporation

Abstract　We propose a mechanism to reduce operations, which are necessary to update or add tools in a design environment. In our design environment DATE, design knowledge is categorized into two types. One is the integrated design knowledge common to all design tools. The other is the design knowledge peculiar to each design tool. A knowledge translation mechanism between these two types is incorporated into the environment. By this mechanism, the knowledge management of each design tool is unified, and each function of the environment is independent of latter type of knowledge. Hence, the influence of tool modification on the environment is minimized and the modifiability is improved. In this paper, user interface integration mechanism of DATE is mainly described and evaluated.

1．はじめに

1．1　設計支援環境の必要性

計算機によるディジタルシステムの設計支援への要求の1つとして，ハードウェア設計者にとって使い易い設計支援機能を構築することがあげられる。そこでは，設計者の，純粋な設計作業以外の負荷が少ないことが要求される。

この様な設計支援をおこなうための手法の1つとして，総てのハードウェア設計作業を支援する，単一の大規模高機能ツールを開発することが考えられる。しかし，ディジタルシステムの開発，製造技術の進歩の速さに対応するための保守作業は，部分的な設計機能をサポートする小規模なツールに比べて困難である。また，ディジタルシステムの設計・開発部門毎にその設計・開発形態が異なり，その形態も変化し続けるため，あらゆるユーザの設計作業に適合する大規模ツールを開発することは困難である。

このため，比較的単機能な複数の設計支援ツールを組合せて設計作業を行なうための，設計支援環境を構築する手法が必要となる。そこでの目的は，以下の2種類の負荷を同時に軽減することである。

(1) ディジタルシステムの設計作業負荷
　ハードウェア設計者が，その環境においてディジタルシステムを設計する作業に要する負荷である。すなわち，ツールの実行，設計結果の判断，設計データの管理などに要する労力のことを指す。

(2) 設計支援環境の更新作業負荷
　ユーザ部門においてツール群を管理する人の作業負荷である。これには，新しいツールを環境に組込む作業に要する負荷と，従来からあるツールを更新する作業に要する負荷が含まれる。

一般に，設計作業負荷を軽くするためには，設計データモデルの共通化，設計操作モデルの共通化を図り，ツール間の結合密度を高めなければならない。しかしこれは，ツール間，ツールと環境の間の相互依存性を増すことにつながるため，更新作業負荷を増大させる。そこで，ツールの独立性を保ったまま，設計モデルを環境レベルで統合し，ツール間，ツールと環境間の意味的なつながりを密にする必要がある。

1．2　従来の設計支援環境

近年，複数のツールの実行制御とデータ管理を自動化し，設計作業負荷を軽減しようとする設計支援環境が研究されている[1]-[4]。これらの環境では，各ツールの機能に関する知識をスクリプトやルールで記述する。そして，エキスパートシステムにより，ツールの実行とデータの管理に関する設計作業負荷を最小限にしようとするものである。しかし，各ツールに関する設計知識の記述と，一般の設計知識の記述が分離されていないため，ツール更新作業において，設計知識のどの部分を更新すればよいかが明確ではない。

他方では，ツールと設計データをオブジェクトとして定義し，環境に組込む手法が提案されている[5]-[6]。しかし，そこでは，設計対象に関する統合的な枠組みがないため，ハードウェア設計者はオブジェクト化された各ツールの仕様を理解しなければならず，設計作業負荷が軽減されるとは限らない。

また，各種ユーザに対して，複数のベンダーが開発したツールを有効に組合せた環境を構築するための，フレームワークが実用化されつつある[7]。しかし，その基礎となるのはフレームワークとして固定された，共通のデータモデルである。そして，この共通データモデルと異なるデータモデルを持つツールを組込む場合，それらのデータモデル間の意味上の対応は，そのデータの変換系に依存する。すなわち，データモデル間の対応が明確でない。このため，ツールの変更時，他のツールの新規組込み時の影響範囲が明確でない。また，基礎となるデータモデルの範囲を越えるツール更新への対応は不可能である。

1．3　DATEの基本機構

本環境 DATE(Design Automation Tools integration Environment)では，環境全体で統合化

された一般的設計知識と，個々のツールに関する設計知識とを分離して定義し，知識ベースに保持する。これにより，個々のツールに固有な設計知識の管理を一元化し，環境の各種機能を，個々のツールに固有な設計知識から切り離している。このため，設計支援ツールの更新作業は，それぞれの設計知識に対する更新作業に分離され，ツール更新時の環境への影響を最小限におさえることが可能である。また，この２種類の設計知識の間の変換機能を介して，ツール間の意味レベルでの結合を実現することにより，設計作業負荷を軽減させている。

本報告では，まず本環境の機能を述べ，次に設計知識の表現方法と設計知識の変換機構について説明する。そして，環境の基本的機能への適用の例として，ツールに共通なユーザインターフェースの構築について解説する。最後に，具体的な環境の更新作業負荷の評価を行い，ツール固有の更新に関しては更新作業負荷が軽減されることを示す。

２．ＤＡＴＥの機能と設計知識変換

２．１　ＤＡＴＥの設計知識構造

本環境の構成を図１に示す。本環境では，ある範囲で閉じた設計知識の集合をドメインとして定義している。設計知識ベースは共通ドメイン，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。知識ベース内には以下の設計知識が含まれる。

(1) 共通ドメイン知識

環境設計者によって定義され，設計対象の機能と構造に関する知識を保持する。これは，環境全体で統合された設計モデルを定義することに相当する。

(2) ツールドメイン知識

ツール設計者によって定義され，各ツールに依存した知識を保持する。また，この知識と，共通ドメインの知識との対応を保持する。ここでは，ツールの機能と，その入出力データの構文と意味を記述することにより，ツールの設計データモデルと設計操作モデルを定義する。

(3) 設計戦略セグメント

各設計フェーズ毎の設計手順を保持する。設計戦略プランナは，この記述を用いて，各ツールの実行手順を決定する。ツールの実行が失敗した場合の再設計手順も保持する。

２．２　環境の各機能と更新作業負荷の軽減

本環境の各サブシステムは，ツール間，ツールと環境との間の意味レベルでの結合を，共通ドメインの設計知識とツールドメインの設計知識の間の変換機構を介して行なっている。これにより，ツール間の設計モデルの差異を吸収し，設計作業負荷を軽減している。また，この共通ドメインとツールドメインの分離記述により，環境としての更新作業負荷が軽減される。以下では，本環境の各サブシステムの機能と知識変換について概説する。

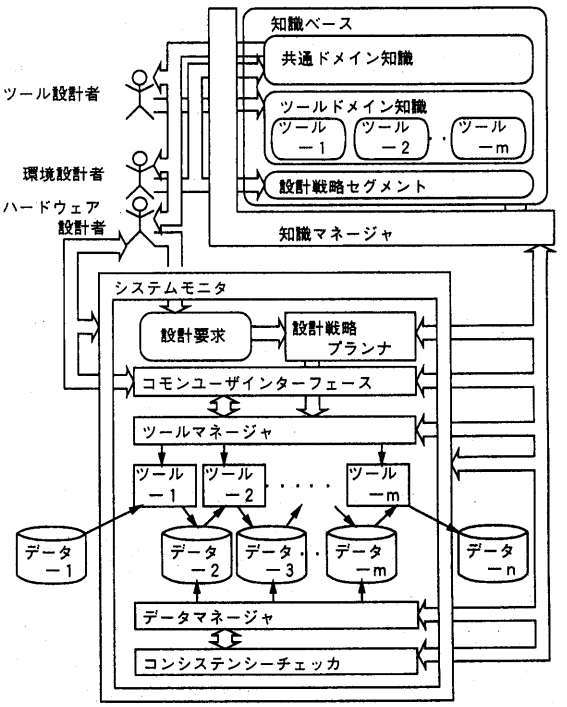


図１：ＤＡＴＥの構成

(1) データマネージャ

設計途中の設計データを管理するバージョン管理と，設計空間の異なる設計データを管理するレビジョン管理を行なう。本環境では，共通ドメインの設計知識を用いてデータの管理手法の記述をおこない，これを，ツールドメインの知識による，実際のデータを管理する記述へ変換する。これにより，データ管理手法の記述を，個々のツールのデータ構造から分離している。たとえば，「論理回路を変更した場合には，その回路データと，シミュレーション用のデータをバックアップする」というデータ管理手法の記述を共通ドメインで行なうことにより，特定のツールの論理回路のデータ構造が変更されたとしても，管理手法の記述への変更は不要となる。

(2) コンシステンシーチェッカ

このサブシステムでは，設計データの正当性を検証し，データの管理に利用する。この正当性には，以下の２種類が存在する。本環境では，設計データの意味上の正当性を扱うことにより，この双方に対してデータの内部まで考慮した検証・データ管理を行なうことが可能である。

① 複数の設計データ間の正当性検証

相互依存性のある複数の設計データ間で，片方が変更された場合に，その修正内容に応じて他方を更新し，データの正当性を管理する機能を提供する。例えば，図面データが変更された場合には，その図面に基づいたシミュレーションをやり直す必要がある。これに対し本サブシステムは，シミュレーションに影響しない図面情報が変更された場合には，シミュレーションをやり直す必要はないことを判断する。本来，あるデータの変更内容と，その変更内容に応じて影響が生じる全てのデータ群の対応を記述して，環境に与えておく必要がある。これに対し本環境では，共通ドメインを通して複数の設計データ間の関連を管理するため，ツールの追加／変更が生じた場合にも，データ間の依存関係を自動的に判断できる。

② ツール間の共有データの正当性検証

複数のツールに渡って，設計データの意味上の正当性を検証する機能を提供する。例えば，「ディジタルシミュレーションに用いられる図面データは，その中にアナログシンボルを含んではならない」など，実際にそのデータを処理するツールに応じて，データの意味上の正当性を検証することが可能である。これは，あるデータファイルに対してツールドメインでの認識を行い，これを共通ドメインを介して別のツールドメインに変換することにより，行なわれる。本来は，一連の設計作業において用いられるツール群ごとにこのような機能を作成する必要があるが，本環境では，これを自動的に判断することが可能である。

(3) ツールマネージャ

ディジタルシステムの設計作業に必要なツールを起動し，実行過程を管理し，設計結果を判断し，次の設計作業を決定する。このためには，個々のツールの具体的な実行制御方法に関する知識が必要となる。本環境においては，ツールの機能と設計データに関する知識を，ツールドメインとして一括定義することにより，ツールマネージャと各ツールとの独立性を確保し，ツールの追加／更新に対するツールマネージャへの影響を最小限に抑えている。また，あるツールの出力結果に応じて別のツールの実行を制御する場合には，ツールの制御出力データを，共通ドメインを介して他のツールの制御入力データに変換することにより，制御データ間の意味上の対応を決定することが可能である。

(4) 設計戦略プランナ

設計戦略セグメント内の部分的設計手順と，ツールドメイン知識内の各ツールの機能に基づいて，ツールの実行手順を決定する。設計戦略セグメントは，共通ドメインの設計知識を用いて記述されているため，特定のツールに関する設計知識から分離される。すなわち，ツールの変更が設計戦略に影響を与えない場合には，環境設計者は，設計戦略セグメントを変更する必要がない。

(5) コモンユーザインターフェース

各ツールは異なったユーザインターフェース（ＵＩ）を持っている。本サブシステムは，ハードウェア設計者がこれらのツールを共通のＵＩで利用できるようにする。フレームワークなどにおいても，アイコンと，アイコンに対する操作を共通化するなどの低いレベルの共通ＵＩが実現されている。しかし，ここではさらに，統合された設計モデルにより，論理ゲートなどの設計対象の共通化と，それに対する操作の共通化を含む共通ＵＩを実現する。すなわち，ハードウェア設計者からの設計要求の入力と，ハードウェア設計者への設計結果の提示を共通ドメインの知識でおこなうことにより，ツールに依存しない共通モデルでのＵＩを実現する。この共通モデルに影響しないツールの追加／変更は，そのツールドメインの知識だけを追加／変更することにより実現される。

３．知識変換機構

３．１　設計知識の表現

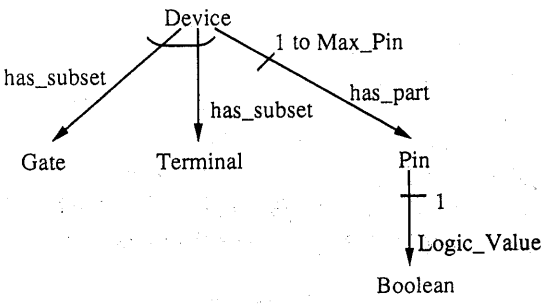
本環境では，設計知識は，記述言語 DOCK[10] (DOmain Concerning Knowledge description language) を用いて記述される。これは，意味ネットワークSEND (SEmantic Network for Design knowledge) に変換されて知識ベース中に保持される。この意味ネットワークは K-NET[8] を基礎とし，設計対象の表現機構，設計操作の表現機構などの拡張を行なっている。

SENDでは，設計対象を集合やその集合の要素としてノードで表現し，そしてそれらの間の関係をアークで表現する。図２の例では，「デバイス 'Device' はゲート 'Gate' と端子 'Terminal' に分類される」こと，また「デバイス 'Device' は１個以上，Max_Pin個以下のピン 'Pin' を持ち，そのピン 'Pin' には論理値 'Logic_Value' が与えられている」ことを表現している。

また，本意味ネットワークは，複数のノードとアークの集合を部分意味ネットワークとして定義する機能を持ち，同値，包含，動作などの対象を，部分意味ネットワークとして定義する。そこでは各ドメインの設計知識もまた，部分意味ネットワークとして管理される。図３の例では，「スケマティックエディタのドメインにおいて，電源シンボル 'Sym_Vcc' が要素 '(SV)' を持つならば，共通ドメインにおいても，それに対応する，電源端子 'Vcc_Terminal' の要素 '(VT)' が存在する」ことを部分意味ネットワークで表現している。

```
Device :=  has_subset                          Gate;
           and                                 Terminal;
       :=  has_part          1 to Max_Pin      Pin;
Pin    :=  has Logic_Value   1                 Boolean;
```

(A) DOCKによる表現

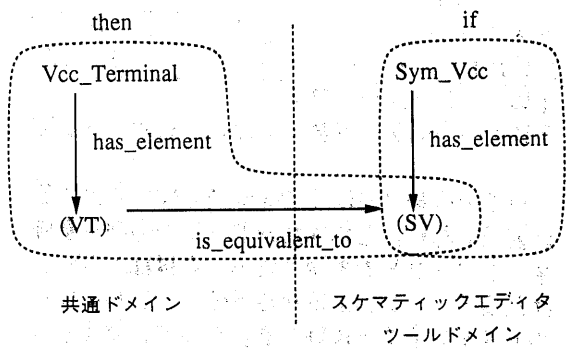


(B) SENDによる表現

図２：設計知識の表現

```
if   [Sym_Vcc          := has_element      (SV)]
then [$Vcc_Terminal    := has_element      (VT);
      (VT)             := is_equivalent_to (SV);]
```

(A) DOCKによる表現



(B) SENDによる表現

図３：ドメイン間の対応の定義

３．２　推論エンジン

本システムの推論エンジン DOT(DOmain concerning knowledge Translation engine) は，知識ベースマネージャ内に組込まれ，設計知識の変換を行なう。そのために，共通ドメインで与えられた意味ネットワークを，推論規則に基づいて，特定のツールドメインの意味ネットワークに変換している。この推論規則には，意味ネットワークの構造を定義するための規則，設計操作を表わす規則，ドメインの間の対応関係を表現する規則，そして，意味ネットワーク全体で共通な組込み規則が含まれる。

４．具体的ユーザインターフェースの改善

ここでは，シミュレータとスケマティックエディタで構成される論理設計環境について，その設計知識の記述と，共通ドメインからツールドメインへの知識変換過程を説明する。各設計知識の記述量は，表１に示す通りである。

表１：知識記述量

	共通ドメイン	ツールドメイン	
		スケマティックエディタ	シミュレータ
設計対象定義	45 19 | 19	23 12 | 12	21 11 | 11
設計対象ルール	16 20 | 30	0 0 | 0	11 18 | 20
動作定義	- - | -	52 33 | 47	19 20 | 20
ドメイン対応定義	- - | -	24 35 | 21	16 28 | 23

上段……DOCKステップ数
下段左…SENDノード数　下段右…SENDアーク数

ＵＩの改善における知識変換では，入力は設計要求であり，これは共通ドメインの部分意味ネットワークとで表現される。本推論エンジンは共通ドメインの上位から下位へ，さらにツールドメイン下位へ推論する。そして，与えられた設計要求を，目的とするツールドメインの最下位の動作，すなわち，個々の実行コマンドだけで構成できた時，推論を終了する。この過程は以下の３段階となる。この様子を図４に示す。この図において，アークのラベル 'he' は集合－要素関係を，'hp' は全体－部分関係を表わす。

(1) 共通ドメイン内の推論

ユーザは，目標とする状態を表わす部分意味ネットワークを共通ドメインで表現する。これが設計要求に相当する。本推論エンジンは，これに対して，共通ドメインでの推論規

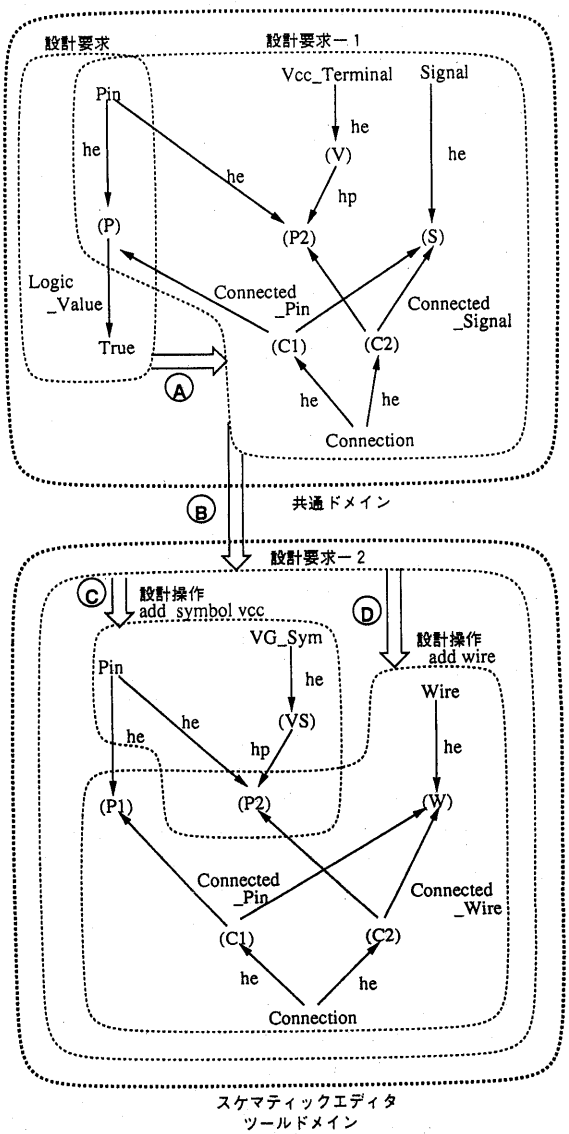


図４：設計知識の変換過程

則を適用することにより，共通ドメインの知識による，別の部分意味ネットワークに変換する。この推論は当該ツールのドメインへの変換が可能になるまで行なう。この結果，ツールとの対応が明確に定義された，共通ドメインの部分意味ネットワークを得る。図４においては，「ピンの集合'Pin'の要素'(P)'の論理値'Logic_Value'を真'True'にする」という共通ドメインでの設計要求を考える。これに対し，スケマティックエディタのドメインでは，ピンの論理値という概念が存在しない。そこで，知識変換Ａにより，この要求を，「電源端子'Vcc_Terminal'のピン'(P2)'に接続する'Connection'」という設計要求－１に変換している。

(2) ツールドメインへの変換

同値，包含，等価などによる，共通ドメインとツールドメインの対応関係を表わす推論規則を用いて，共通ドメインの部分意味ネットワークを特定のツールドメインの部分意味ネットワークに変換する。図４では，知識変換Ｂにより，スケマティックエディタのツールドメインにおける「電源シンボル'VG_Sym'のピン'Pin'に接続する'Connection'」という設計要求－２に変換している。

(3) ツールドメインでの動作への変換

ツールドメインの設計要求を入力として，ツールドメインの基本的な動作だけで構成される部分意味ネットワークを求める。この動作は各ツールの個々の実行コマンドに直接対応する。図４の例では，知識変換Ｃ，Ｄにより，設計要求－２が，「'add symbol vcc'コマンドにより電源シンボルを追加し，'add wire'コマンドによりそのピンとワイヤで接続する」という設計操作に変換されている。

以上により，共通ドメインでの設計要求が，スケマティックエディタのドメインでの設計操作に変換された。この知識変換により，特定のツールに依存しない共通モデルでのＵＩを提供することが可能であり，共通モデルに影響を与えない範囲でのツールの更新作業は，ツールドメイン知識の更新だけで実現される。

５．評価

５．１　ユーザインターフェースの改善量

４章における設計知識変換の例では，シミュレータとスケマティックエディタの知識量の和が，DOCKソースで１６６ステップであるのに対して，共通ドメインの知識量は６１ステップとなっており，約３／５に減少している。この様に，複数のツールが類似した設計データモデルを持ち，それらが共通ドメインで単一化できる場合には，かなりの設計知識が削減される。但し，この例では，設計データの詳細な内部構造まで定義していないので，コンシステンシーチェックなどを行なうには不十分である。

５．２ 環境の更新作業負荷

従来の環境と本環境における環境の更新作業

表２：設計支援環境の更新作業項目

ツール変更内容			環境	DAT E 設計戦略	DAT E ツールドメイン			DAT E 共通ドメイン		フレームワーク	自動設計環境
				ツール実行手順	データ管理手順	データ	アクション	対応	データ	アクション		
部分的変更	設計データ	追加		－	－	○	－	○	△	－	○	－
		削除		－	－	○	－	○	△	－	○	－
		変更		－	－	○	－	○	△	－	○	－
	制御データ	追加		△	－	－	○	－	－	△	○	○
		削除		△	－	－	○	－	－	△	○	○
		変更		△	－	－	○	－	－	△	○	○
	性能改善			△	－	－	－	－	－	－	△	△
新規作成	ツール変更			△	△	○	○	○	△	△	○	○
	ツール追加			△	－	○	○	○	△	△	○	○
	データ追加			－	△	○	○	○	△	△	○	○

○…修正必要　　－…修正不要
△…修正必要な場合あり

項目を，ツールの変更内容に応じて分類したものが表２である。この表の○で示された部分より，ツールの変更・追加に対する本環境の修正箇所は，ツールドメインのデータ部分またはアクション部分と，ツールドメインと共通ドメインとの対応定義部分に限られる。これにより，ツールの追加，更新時の環境の更新作業は，当該ツールドメインに限られることがわかる。ここで，△で表現される部分は，共通ドメインでの記述内容を逸脱して，ツールの変更が生じた場合であり，この時には共通ドメインと，設計戦略セグメントの変更も必要になる可能性がある。しかし，将来的には，この場合にも，ツールドメインの変更内容に対して，共通ドメインの変更が必要であるか否かを環境設計者に提示する機能を開発することにより，環境の更新作業負荷を軽減できると考えている。

６．まとめと今後の課題

知識変換機構を用いて，複数のツールを意味レベルで統合化し，設計作業負荷と更新作業負荷の双方を軽減するための設計支援環境について報告した。またその様な設計支援環境のＵＩを改善するための具体的知識変換過程を示した。

今後は，ツール間の意味レベルでの統合化を用いて，さらに設計作業を効率化する手法を模索すると同時に，更新作業負荷を軽減するための機能として，データコンバータの自動生成，ツール間の対応関係の自動生成などを検討している。

また，今回は，比較的小規模な設計知識記述を行ったが，実務レベルのシステムでは，ハードウェア設計データに関する，より詳細な知識を定義する必要があり，また，その記述量も膨大になる。そのような場面では，環境に組込まれている全ツールを理解・認識して共通ドメインの知識を定義することは非常に困難である。これを容易にするためには，インクリメンタルな知識の定義とその矛盾検出機構が必要であろう。また，その様な複雑な環境においては，同一の作業を行なう複数のツールが存在し，その設計知識が共通ドメインでは矛盾を生じる可能性がある。このように矛盾の存在する設計知識を管理するための機能として，一部のツールドメインに関連する共通ドメインの知識だけを用いた設計作業を可能とする機構が必要となろう。

［参考文献］

[1] A. D. Janni, "A Monitor for Complex CAD Systems", 23rd D. A. Conference, 1986

[2] M. L. Bushnell, S. W. Director, "Automated Design Tool Execution in the Ulysses Design Environment", IEEE Trans. on CAD, Vol. 8, No. 3, Mar. 1989

[3] D. W. Knapp, A. C. Parker, "A Design Utility Manager: the ADAM Planning Engine", 23rd D. A. Conference, 1986

[4] Z. Mehmood, A. Singhal, N. C. Srinivas, S. L. Taylor, K. Wu, "IDEAS: An Integrated Design Automation System", IEEE Trans. on CAD, 1987

[5] L. P. Deners, P. Jacques, S. Fauvel, E. Cerny, "CHESHIRE : An Object-Oriented Integration of VLSI CAD Tools", 24th D. A. Conference, 1987

[6] J. Miller, K. Groming, G. Shulz, C. White, "The Object-Oriented Integration Methodology of the Cadlab Work Station Design Environment", 26th D. A. Conference, 1989

[7] D. S. Harrison, A. R. Newtom, R. L. Spickelmier, T. J. Barnes, "Electric CAD Frameworks", Proc. of the IEEE, Vol. 78, No. 2, Feb. 1990

[8] R. Fikes, G. Hendrix, "A Network-Based Knowledge Representation and its Natural Deduction System", 5th IJCAI, 1977

[9] 新井浩志, 長谷川拓己, 深沢良彰, 門倉敏夫, "統合化設計支援環境DATEにおける設計知識構造", 情報処理学会第40回(平成２年度前期)全国大会, 6M-2, 1990年3月

[10] 新井浩志, 深澤良彰, 門倉敏夫, "設計支援ツールの統合化環境DATEにおける知識変換について", 1990年度人工知能学会全国大会(第４回), 16-35, 1990年7月