解　説

ディジタルハードウェアのための設計支援環境†

新井浩志††　深澤良彰†††

1. はじめに

VLSI などのディジタルハードウェアの設計には，その動作を検証するためのシミュレータや，チップ内の物理的な配置，配線を決定するためのツールが不可欠である．これらの設計支援ツールが出現した当初は，単一の，単純な設計作業に限定した支援をしていたため，大きな問題は生じていなかった．しかし，近年は，設計作業全体を支援するために，多数のツールが利用され，それにともなって多種・多数のデータを扱わなければならなくなってきている．このような設計作業では，複数のツールとデータが存在することによって，さまざまな問題が生じる．たとえば，複数のツールのうちでどのツールを使用すればよいのか，その具体的な起動方法は，そして，複数のデータのうちでどのデータを用いればよいのか，などの問題である．これらの問題に対処することが設計支援環境の目的である．

設計支援環境は，複数の独立した設計支援機能と，それらを効率的に組み合わせて設計作業を進めるための機能から構成される．特に，この後者の機能だけを抽出し，さらに，個々の設計支援ツールを自由に変更できるようにするための特別な配慮をしたものが，広くフレームワーク[1]と呼ばれているものである．

ネットワーク技術の進歩にともない，複数の計算機システムにまたがってツールとデータとを分散配置することが可能になった．その結果，どの計算機上にどのデータが存在し，どの計算機上のツールを実行すればよいのかが不明確であるという問題が生じている．また，複数の人間が設計作業に関与するために，複数の設計者間で設計作業を調停しなければならないという問題などが存在する．広義の設計支援環境としては，これらの問題に対処するための機能も含めることが多いが，オペレーティングシステムが扱うべき範囲と重複する部分が多いため，本稿では **4.3** で概説するにとどめる．また，本解説の内容の多くは，アナログハードウェアに対しても当てはまる．しかしアナログハードウェアは，そのデータモデルの複雑さ，多様性のために，複数のツールを統合化した設計支援環境が，解説の対象となるほど多数は実現されていないのが現状である．このため本稿では，ディジタルハードウェアのための設計支援環境に限定して解説する．

以下では，**2.** においてソフトウェア開発環境との差異，**3.** において従来の設計支援環境の経緯と問題点を述べる．**4.** では設計支援環境に要求される機能を概説し，**5.** でそれに対する研究開発状況について紹介する．

2. CASE 環境との差異

ソフトウェア開発作業を支援するための環境として，CASE環境が広く論じられている[2]．CASE環境が解決しようとしている本質的な問題点は，ハードウェア設計支援環境と同等である．しかし，設計対象がソフトウェアであるか，ハードウェアであるかによって，いくつかの差異が生じている．

一つはディジタルハードウェアの設計・製造技術の進歩の速さである．ハードウェア設計支援環境の設計対象の製造技術は日々進歩し続けており，その設計支援ツールも常に変化している．こ

† Supporting Environments for Designing Digital Hardware by Hiroshi ARAI (Department of Electrical Engineering, School of Science and Engineering, Waseda University) and Yoshiaki FUKAZAWA (Department of Information and Computer Science, School of Science and Engineering, Waseda University).

†† 早稲田大学理工学部電気工学科
††† 早稲田大学理工学部情報工学科

のため，ハードウェア設計支援環境は，ツールやデータの変更に対して容易に対応するための機能・構造が必要となる．二つ目の差異は，ユーザごとに設計の対象や範囲が異なるため，設計に必要なツールやデータが異なる点である．たとえば，VLSI の論理設計だけをおこなうユーザにとって，その実装情報に関するデータと，それを扱うツールは不用である．このため，ハードウェアの設計支援環境には，ユーザが必要とする任意の設計支援機能を統合化する機能が必要である．最後の差異は，各ツールが設計対象をどのように表現しているかという，データモデル[3]の多様性である．設計対象はツールによってさまざまなモデルを用いて表現されている．複数のツールをもつ環境は，ツールごとのモデルの差異を許すための機能をもつ必要がある．

以上のように，ハードウェア設計支援環境は，その柔軟性に対する要求が非常に強いと言える．

3. 初期の設計支援環境

ディジタルハードウェアの設計作業に設計支援ツールが利用されるようになった当初は，論理回路の検証のための論理シミュレーションや配置・配線など，特定の設計作業を局所的に支援するツールが存在するだけであった．これらを用いた設計作業は，単一の計算機上にある単一のデータに対して，単一のツールを実行するだけであり，設計支援環境は必要ではなかった．しかし，設計作業のさまざまな局面を支援するツールが出現するにつれて，複数のデータを扱う必要が生じてきた．そこでの最大の目的は，設計データの再入力を避けるためのデータの流用であり，必要な分だけのデータコンバータを用いてデータを共有する手法が用いられていた．しかしこの手法では，多数のデータコンバータが必要となり，複数のデータの管理も複雑になる．そこで，各ツールが共通の設計データベースにアクセスする手法が広く用いられるようになった[4]．しかし，環境の柔軟性の問題が解決されていない点と，ツールの管理が考慮されていない点から，より統合化され，かつ柔軟な設計支援環境に対する要求が強まっている．

4. 機能的要求

ハードウェア設計支援環境に必要な機能は，複数の設計者が，複数の計算機上で，複数のツールを用いて，複数のデータに対して設計操作を施す作業を支援する機能である．本章では，**4.1** で複数のデータの管理機能，**4.2** で複数のツールの管理機能について解説する．また，**4.3** では，その他の管理機能として，複数の設計者の管理，複数の計算機の管理について述べる．

4.1 データの管理機能

単一のハードウェアを設計する場合に，多種のデータが存在すると，どのデータに対して設計操作を施せばよいのかを常に注意しながら作業を進める必要がある．これはハードウェア設計者に対する負担となる．この複数のデータの間の関係としては，以下の3とおりが存在する．

①ある設計対象が，物理的な部分回路ごとに別別のデータで表現され，また，トップダウン設計の上位階層と下位階層を表わす複数のデータが存在する場合

②機能的側面と物理的側面などを別々に設計する場合：このときは，同一の設計対象に対して視点を変えた複数のデータが存在する．

③設計の進捗とともに複数のデータが生成される場合

これらの関係を管理するために，従来からのデータベース管理システムの概念と，ソフトウェア開発におけるソース管理システムの考え方などが広く用いられている．しかし，前述の③には適しているが，部分ごと，視点ごとのデータの管理には十分ではない．これらデータ間の複雑な関係を保持するためには，複数のデータを統合した上位のデータ概念を表わす必要がある．これは高水準データモデル[5]などとして実現されている．

（1） 設計の進捗に応じたデータ管理

設計の進捗に応じて生成される複数のデータを管理する機能である．一般にバージョン管理，オルタナティブ管理，コンフィギュレーション管理がおこなわれている．バージョン管理では，設計対象が詳細化されるたびに，または問題点が修正されるたびに別の版として保持する．これはソフトウェアのソース管理システムと同じ考え方である．これに加え，ハードウェアの設計作業では，

異なった複数の詳細化を並列におこなうことによって，設計のトレードオフを評価しながら，試行錯誤的に設計を進めるという作業が頻繁におこなわれる．そこでオルタナティブ管理では，異なった手法によって詳細化したデータを別の版として管理する．これら2種類の管理は，単純なデータ間の関係を管理することを目的としている．これに対して，コンフィギュレーション管理では，先の①や②を含めて，複数のデータのバージョンを組み合わせて考えることによって，設計対象全体のバージョンを管理する．

以上のデータ管理の例を図-1に示す．ここで，設計対象はAとBの二つの部分から構成され，それぞれ個別に設計作業が進展している．Aのバージョンは詳細化に応じて $A1$〜$A5$ まで作成されており，バージョン $A2$ からの詳細化では，相異なった手法によって詳細化した二つのデータ $A3$ と $A3'$ が存在する．設計対象全体のコンフィギュレーションは，$A2$ と $B2$ を合わせた $C1$ や，$A3$ と $B4$ を合わせた $C2$ として表現される．

（2） データの整合性の管理

ハードウェア設計では，設計対象の各部分を表現した複数のデータや，同一の設計対象に対して複数の視点から見た相異なるデータが存在し，それらのデータごとの設計作業が独立して進行する．このため，先に①〜③で述べた複数のデータ間の関係をもとに，それらのデータ間の整合性を保つための機能が必要である．また，データのフォーマット変換や，部分的な情報の抽出をおこなった場合には，それらのデータ間の依存関係を考え，整合性を管理する機能が必要である．

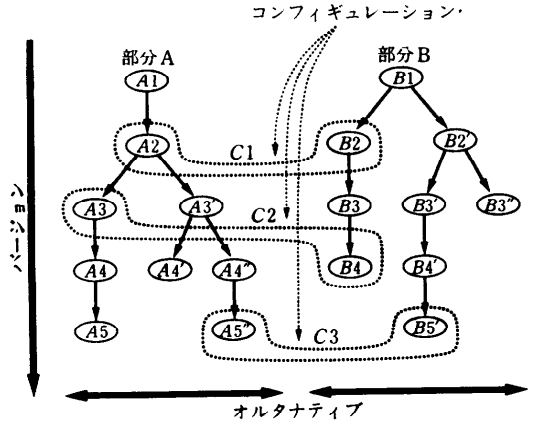

図-1　設計の進展に応じたデータ管理

（3） データモデルの管理

設計支援環境は，個々のツールが，ハードウェアをどのようにモデル化しているかによって，同一のデータモデルを扱うツールを統合化するものと，異なったデータモデルを扱うツールを統合化するものに分けられる．異なったデータモデルをもつツールを統合化する環境においては，先の①〜③までのデータ間の関係の管理に加え，それらが異なったモデルで表現されることによる問題を解決しなければならない．従来のデータベース管理システムにおいては，実世界を自然かつ抽象的に表現するための意味データモデル[5]に関する研究がおこなわれており，データの構造を表わすスキーマ記述を，別のスキーマ記述へ変換する機能[6]が提案されている．ハードウェア設計支援環境にも，このような機能が必要である．

4.2 ツールの管理機能

ハードウェア設計者は，複数のツールの中で次にどのツールを実行すればよいのか，そして，それをどのように起動すればよいのかを判断しなければならない．また，その実行過程や実行結果を観察し，次の設計作業を決定しなければならない．単純な，少数のツールが存在するだけであれば，これらの作業は容易であるが，複雑なツールが多数存在する場合には，ハードウェア設計者の負担は非常に大きくなる．

バッチ形式のツールの管理をおこなうためには，個々のツールに固有な詳細情報を隠蔽したうえで，必要に応じてツールの起動を自動化する必要がある．これはツールのカプセル化の問題として扱われる．また対話形式のツールでは，個々のツールごとのユーザインタフェースの差異をなくすための，ユーザインタフェースの標準化の問題が存在する．

（1） ツールのカプセル化

個々のツールは，複雑な複数の機能をもち，その機能を実行するための操作手順も，さまざまである．この差異をなくすためには，個々のツールを利用するために必要な最低限の外部仕様を環境内に保持し，ハードウェア設計者が，設計支援ツールをブラックボックスとして利用できるようにしなければならない．ここで，その外部仕様をどこまで，どのように記述するかという問題が生じる．ツールの仕様を完全に記述するためには，

仕様記述言語によるソフトウェアツールの仕様記述と同等の記述量が必要になる．これに対して，一般には，個々のツールの特性だけを表形式で記述している．この特性としては，ツールの名称，起動方法，オプションの記述形式とその機能，入出力ファイルの種別などが用いられる．環境は，この情報を利用してツールを起動する．

（2） ユーザインタフェースの共通化

操作方法が異なった対話型のツールが多数存在すると，ハードウェア設計者は困惑してしまう．この問題はハードウェアの設計支援ツールに固有な問題ではないため，Motif[7]などによって，任意のアプリケーションソフトウェアのグラフィカルユーザインタフェースの共通化がおこなわれている．

さらに，ツールごとのデータのモデルが異なる場合には，各データがどのような意味をもつのか，各操作によってどのようにデータが変更されるのかをハードウェア設計者が把握していなければならない．たとえば，論理回路図を入力するためのツールであるスケマティックエディタにおいて，空きピンを自主的にグランドに落とすべきなのか，単に空きピンとして宣言すればよいのかは，ツールごとのデータのモデルに依存している．この場合も，データの管理と同様に，設計支援環境がツールごとのデータモデルの差異を管理する必要がある．

（3） 設計手順の自動化

一般の設計作業は複数のツールを繰り返し実行することによって進められるため，ハードウェア設計者は，どのツールをどの順序で実行すればよいのかを判断しながら設計作業を進める必要がある．

設計手順を部分的に自動化するためには，複数のツールを組み合わせた設計操作を定義・実行する機能が要求される．最も単純な方法としては，複数のツールの起動命令列を定義しておき，この定義に従ってツールを起動する手法があげられる．しかし，複雑な設計操作を自動化するためには，設計結果を評価・判断し，次の設計手順を自動的に決定するための条件判断や，設計要求が満たされるまで設計操作を繰り返すための反復作業を記述する機能などが必要である．また，複数の設計作業を並行に進めることが多いため，並行プロセスの記述機能も要求される．このように並行に動作するツールを許す場合には，それらのツール間のリアルタイムの情報交換機能が必要になる場合がある．

最終的には，現在の設計状況と，各ツールの外部仕様から，どのツールを実行すればよいのかをある程度自動的に判断して，設計作業を進める機能が要求される．ソフトウェアの開発環境においても，このような問題に対する研究がなされている[8]．

4.3 その他の管理機能

ここでは，複数の設計者を管理するための機能と，複数の計算機を管理するための機能について解説する．

（1） 複数の設計者の管理

複数の設計者が存在する場合には，セキュリティの問題として，アクセス権の管理や，ツールの実行権の管理などが必要になる．ファイル単位，ツール単位での管理は，オペレーティングシステムレベルでも可能である．これに対して，上位のデータ概念に対するアクセス権や，上位の抽象的な設計操作に対する実行権は，設計支援環境が管理しなければならない．

また，ソフトウェアの開発環境では，複数の設計者の間の情報交換や，設計作業全体の進行状況の管理などを目的としたグループウェアの研究がおこなわれている．ハードウェア設計支援環境においても，このようなグループウェアの機能が必要である．

（2） 複数の計算機の管理

ワークステーションを用いた一般的な設計支援環境では，複数の設計支援ツールと設計データが，複数のマシンにまたがって存在する．ハードウェア設計者にとって，これらのツールとデータが存在する計算機を意識することは，大きな負担になる．設計支援環境としては，これらを管理し，ハードウェア設計者に，ツールとデータの物理的な存在場所を意識させないための機能が必要である．ただしこの機能は，分散オペレーティングシステムの機能と重複する部分が多い．

5. 設計支援環境の現状

本章では，設計支援環境に関する研究の現状を，特定の設計対象と設計作業を想定した設計支援環境の研究，データの管理を中心としたフレー

ムワークの研究，そして，ツールの管理を中心としたフレームワークの研究に分類して概説する．最後に，フレームワークの標準化動向についてふれる．

5.1 特定目的の設計支援環境

VLSI の実装設計を目的とした環境として，CHIPBUSTER[9]，STEM[10]などがある．CHIPBUSTER は，複数の設計データのデータベース管理モジュールと，ユーザインタフェースモジュールを基礎として，複数のツールを組み込んだ環境である．設計手順の管理，上位のデータ概念の管理などの機能は存在しない．STEM は，Smalltalk のオブジェクト指向の概念を用いて設計データベースを構築した環境である．個々のデータの単位であるセルは，単一のクラスとして定義され，それに対する操作などをカプセル化して保持している．これによって，複数の視点から見たデータの管理を一元化している．

また，機能設計を目的とした環境として，中村の動作記述による環境[11]があげられる．そこでは，コマンドインタプリタにより，ユーザインタフェースの共通化が図られている．

5.2 データ管理を主体としたフレームワーク

データ管理の中心となる機能は，さまざまな版の管理機能である．UCB のバージョンサーバなど[12)~14)]では，バージョン，オルタナティブ，コンフィギュレーションの3種類の基本的な版の管理を実現している．さらに，等価な対象を表わす設計データを管理する機能[12]，ユーザ別のデータ環境を，まとまった作業中のデータの集合体であるワークスペースとして表現する機能[13),14)]などがある．

これに対して，設計データの版の管理，整合性の管理などを十分におこなうためには，データの間の複雑な関係を詳細に表現するデータベースマネージメントシステムが必要となる．ADAM 設計環境のデータマネージメントシステム EVE[15)]では，データフロー情報，構造情報，物理情報，タイミング情報などの間の関係と，個々のデータの階層間の関係を保持している．Cbase[16)]や NELSIS[17)]では，オブジェクト指向データベースによって上位のデータ概念を含めたデータ管理を実現している．図-2 に NELSIS の構成を示す．個々の設計データオブジェクト間の関係はメタデータによって表現され，ツールからデータへのアクセスはデータ管理インタフェースが管理している．

個々の設計データのモデルが異なっている場合には，その差異を吸収するための機構が要求される．ABB CADE 社のシステム[18)]では，共通の

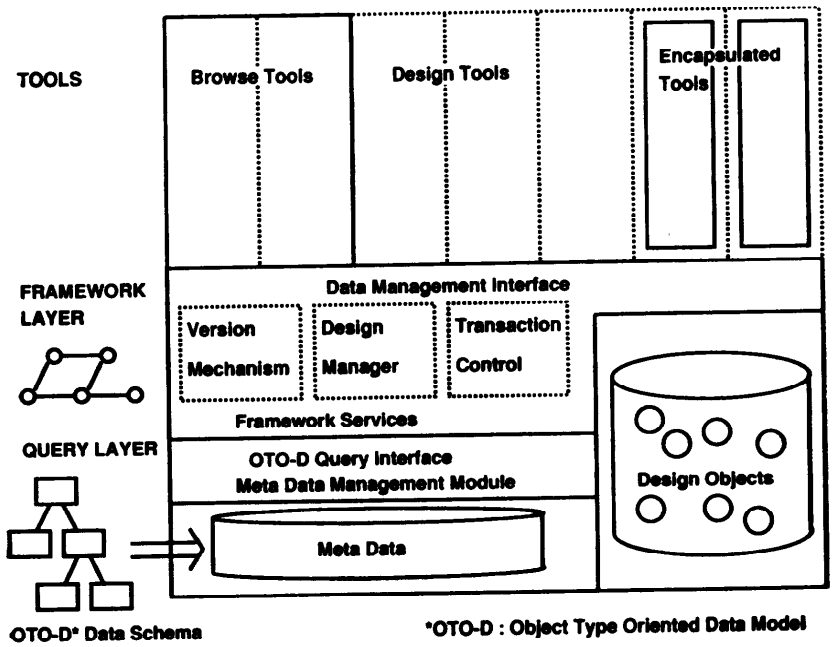

図-2 NELSIS 設計支援環境の構成

データベースに対して，個々のツールがアクセスするためのデータスキーマを用意し，ツールに応じたデータアクセスを可能にしている．DEC の D-BUS アーキテクチャに基づく BRIDGE システム[19]では，データバスと呼ばれる共通のデータ交換スキーマをとおして設計データの交換をするアーキテクチャを提案している．またわれわれは，データとツールの双方の管理において生じる，モデルの差異の問題を解決するための手法として，複数のデータモデルと，その間の対応関係の定義に基づいて動作する設計支援環境 DATE[20]を提案している．

5.3 ツール管理を主体としたフレームワーク

Cadence 社のシステムでは，SKILL[21] という手続き的なカスタマイズ言語によって個々のツールを環境に組み込んでいる．これに対して，ツールのカプセル化は，特定の記述形式で宣言的におこなう場合が多い．MCC 社のフレームワーク[22]では，Lisp ふうの言語を用いてツールの特性を記述し，カプセル化をおこなっている．さらにその言語によって，条件判断や，反復操作を含む設計手順も定義している．NELSIS[23]では，設計手順をフローマップと呼ばれる有向グラフで表現することによって定義している．フローマップでは，並行プロセス，プロセス間の階層的構造の定義なども可能になっている．

Ulysses[24]，ADAM[25]，Cadweld[26] などは，知識工学によるアプローチによって，ある程度の自動設計を実現している．ADAM のプランニングエンジンでは，個々の設計操作を実現するための前提条件と終了条件，そして，各ツールの特性に関する知識から，設計要求を満足させるための実行手順を生成し，それに応じてツールを実行する機能をもっている．Ulysses では，ツールの仕様と個々の設計タスクの内容を，スクリプトと呼ばれる言語で記述しておき，黒板モデルによるプロダクションシステムによって自動的に設計操作を実行している．Cadweld では，オブジェクト指向の概念によってツールをオブジェクトとして定義し，黒板モデルによって自動的に設計作業を進めている．

5.4 CFI による標準化

CFI (CAD Framework Initiative) は，CAD ベンダが中心となって 1988 年に設立された団体であり，フレームワークの規格化を目的として，さまざまな活動をおこなっている．このようなフレームワークの規格化をおこなうことによって，ツールとデータの互換性がさらに向上することになる．CFI の内部には，アーキテクチャ，設計データ管理，設計表現，システム環境，設計手法管理，ユーザインタフェース，ツール間通信などの小委員会が，存在し，それぞれのテーマ別に規格化をおこなっている．図-3 に CFI におけるフレームワークのアーキテクチャを示す[27]．フレームワークは大きくカーネルによる基本機能，データ表現機能，データ保持機能，ユーザインタフェースを提供するためのツールキット，そして，それらを基礎とした設計手法管理機能によって構成される．現在 CFI では OS，ウィンドウシステム，プレゼンテーションマネージャなどを規定しており，さらにツールのカプセル化，階層ネットリストレベルでのマルチベンダデータ交換，そしてツール間通信を可能にしている．しかし，任意

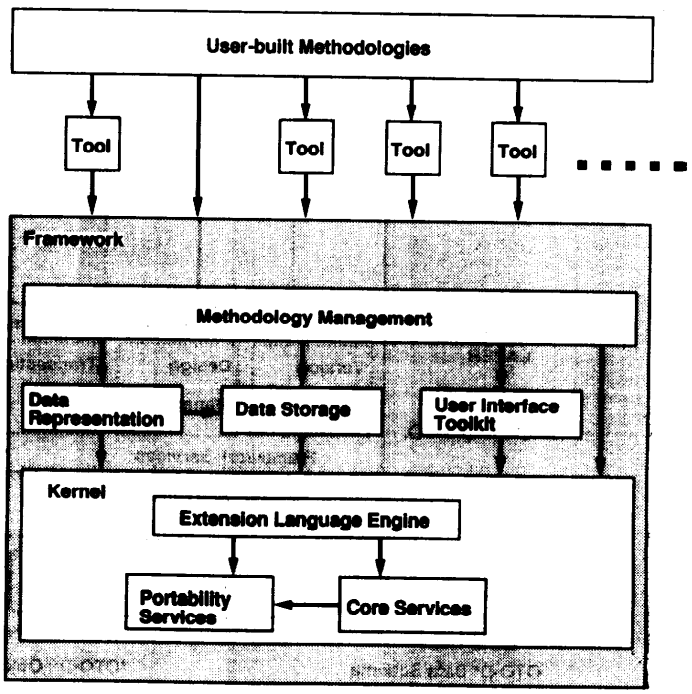

図-3 CFI におけるフレームワークの概念モデル

のツールの組込みを許すことがフレームワークの目的であるのに対して，その一部の機能を規格化することは，ツールを限定することに相当してしまう．この自己矛盾の妥協点をどこに見出すかが問題となるようである．

6. おわりに

以上，ディジタルハードウェアの設計支援環境に要求される機能について，データの管理とツールの管理の二つの側面から解説し，現在の研究開発状況について紹介した．設計支援環境の機能は，複数の設計者による，複数の計算機上での，複数の設計データに対する，複数のツールによる設計作業を支援することであった．

今後は，環境の各機能として以下のようなことが重要となるであろう．

（1） データの管理

各ツールのデータの内部表現まで管理するデータベース管理システムが望まれる．現状のデータベース管理システムを用いてこのような管理をすることは可能であるが，十分な速度が得られないため実用化はされていない．この問題に対しては，実用面からの妥協案[28]が提案されている．

また，整合性の管理においては，データの変更の影響範囲などを詳細に判断する必要がある．これに対して，演繹データベース[29]のような機構を取り入れることによって，データ間の制約条件などを明確に管理することが可能になると思われる．

（2） ツールの管理

現状，ツール間のリアルタイムな情報交換は特定の情報に限定されており，十分ではない．これはデータモデルの差異に依存する部分でもあり，また，その情報の変換速度にも依存している．たとえば，複数のシミュレータ間では，シミュレーション結果の情報がリアルタイムに交換されることによってミックスレベルシミュレーションが実現されることが望まれるが，現状では十分な速度は得られない．

また，将来的には，各ツールが独自に並行に設計作業を進め，必要に応じてリアルタイムに情報交換をおこなう分散協調型の環境[30]も重要になってくるであろう．これを実現することによって，設計自動化の最終的な目標である自動設計が可能となる．このための研究として，設計問題のモデル化・自動化に対する研究[31]と，設計支援環境との融合が重要になってくると考えられる．

参考文献

1) Harrison, D. S., Newton, A. R., Spickelmier, R. L. and Barnes, T. J.: Electric CAD Frameworks, Proc. IEEE, Vol. 78, No. 2, pp. 393-417 (1990).
2) 特集：CASE 環境，情報処理，Vol. 31, No. 8, pp. 1012-1094 (1990).
3) Batory, D. S. and Kim, W.: Modeling Concepts for VLSI CAD Objects, ACM Trans. Database Syst., Vol. 10, No. 3, pp. 322-346 (1985).
4) 須藤常太，唐津 修，永谷三義：設計データベース管理手法，情報処理，Vol. 25, No. 10, pp. 1137-1143 (1984).
5) 特集：高水準データモデルの最近の研究動向，情報処理，Vol. 32, No. 9, pp. 1007-1040 (1991).
6) Batini, C., Lenzerini, M. and Navathe, S. B.: A Comparative Analysis of Methodologies for Database Schema Integration, ACM Comput. Surv., Vol. 18, No. 4, pp. 323-364 (1986).
7) Open Software Foundation, OSF/Motif Style Guide, Prentice Hall (1990).
8) Bisiani, R., Lecouat, F. and Ambriola, V.: A Tool to Coordinate Tools, IEEE Expert Syst., pp. 17-25 (1988).
9) McCalla, B., Infante, B., Brzezinski, D. and Beyers, J.: CHIPBUSTER VLSI Design System, IEEE Int'l. Conf. on Comput. Aided Des., pp. 20-27 (1986).
10) Girczyc, E. F. and Ly, T.: STEM: An IC Design Environment Based on the Smalltalk Model-View-Controller Construct, 24th ACM/IEEE D. A. Conf., pp. 757-763 (1987).
11) Nakamura, Y.: An Integrated Logic Design Environment Based on Behavioral Description, IEEE Trans. on Comput, Aided Des., Vol. 6, No. 3, pp. 322-336 (1987).
12) Katz, R. H., Anwarrudin, M. and Chang, E.: A Version Server for Computer-Aided Design Data, ACM/IEEE 23rd D. A. Conf., pp. 27-33 (1986).
13) Banks, S., Bunting, C. and Edwards, R.: A Configuration Management System in a Data Management Framework, ACM/IEEE 28th D. A. Conf., pp. 699-703 (1991).
14) Silva, M., Gedye, D., Katz, R. and Newton, R.: Protection and Versioning for OCT, ACM/IEEE 26th D. A. Conf., pp. 264-269 (1989).
15) Afsarmanesh, H., Brotoatmodjo, E., Byeon, K. J. and Parker, A. C.: The EVE VLSI Information Management Environment, IEEE Int'l Conf. on Comput. Aided Des., pp. 384-387 (1989).
16) Breuer, M. A., Cheng, W. and Gupta, R.: Cbase 1.0: A CAD Database for VLSI Circuits Using Object Oriented Technology, IEEE Int'l Conf.

on Comput. Aided Des., pp. 392-395 (1988).
17) van der Wolf, P., Sloof, G. W., Bingley, P. and Dewilde, P.: Meta Data Management in the NELSIS CAD Framework, ACM/IEEE 27 th D. A. Conf., pp. 142-145 (1990).
18) Bartschi, M., Stamer, H. U. and Wunderlich, F.: An Integrated Tool Box for the Electrical Engineer, IEEE 2nd Int'l Conf. on Data & Knowledge Systems for Manufacturing & Engineering, pp. 144-147 (1989).
19) Bonneau, R. J.: DEC's Engineering to Manufacturing BRIDGE System Based on the D-BUS Architecture, IEEE Int'l Conf. on Comput. Aided Des., pp. 288-291 (1989).
20) 新井, 長谷川, 深澤, 門倉: 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, 情報処理学会設計自動化研究会, 90-DA-55 (1990).
21) Barnes, T. J.: SKILL: A CAD System Extension Language, ACM/IEEE 27th D. A. Conf. pp. 266-271 (1990).
22) Allen, W., Rosenthal, D. and Fiduk, K.: The MCC CAD Framework Methodology Management System, ACM/IEEE 28 th D. A. Conf., pp. 694-698 (1991).
23) Bosch, K. O., Bingley, P. and van der Wolf, P.: Design Flow Management in the NELSIS CAD Framework, ACM/IEEE 28th D. A. Conf., pp. 711-716 (1991).
24) Bushnell, M. L. and Director, S. W.: Automated Design Tool Execution in the Ulysses Design Environment, IEEE Trans. on CAD, Vol. 8, No. 3, pp. 279-287 (1989).
25) Knapp, D. W. and Parker, A. C.: A Design Utility Manager: the ADAM Planning Engine, ACM/IEEE 23 rd D. A. Conf., pp. 48-54 (1986).
26) Daniell, J. and Director, S. W.: An Object Oriented Approach to CAD Tool Control Within a Design Framework, ACM/IEEE 26th D. A. Conf., pp. 197-202 (1989).
27) CAD Framework Initiative, Draft Proposal: Framework Architecture Reference (1991).
28) Kaufman, G.: Pragmatic ECAD Data Integration, SIGDA Newsletter, Vol. 20, No. 1, pp. 60-75 (1990).
29) 特集: 演繹データベース, 情報処理, Vol. 31, No. 2, pp. 188-243 (1990).
30) 丸山, 角田, 松永他: 評価・再設計機構を備えた論理設計支援システム, 信学論, Vol. J-72-A, No. 8, pp. 1172-1180 (1989).
31) 武田, 河合, 冨山, 吉川: 設計過程の計算可能モデルに基づく設計シミュレーション, 1990年度人工知能学会全国大会 (第4回) 16-28, pp. 583-586 (1990).

(平成3年10月14日受付)

新井 浩志 (正会員)

昭和58年早稲田大学理工学部電気工学科卒業. 昭和60年同大学院修士課程修了. 同年(株)東芝入社. 平成元年早稲田大学大学院博士課程入学. 平成3年同大学理工学部助手. 現在に至る. CAD, DA システムの研究に従事. 人工知能学会会員.

深澤 良彰 (正会員)

昭和51年早稲田大学理工学部電気工学科卒業. 昭和58年同大学院博士課程中退. 同年相模工業大学(現湘南工科大学)工学部情報工学科専任講師. 昭和62年早稲田大学理工学部電気工学科助教授. 平成3年同大学情報学科新設にともない移籍. 工学博士. ソフトウェア工学, ハードウェア設計に対するソフトウェア支援などの研究に従事. 電子情報通信学会, IEEE, ACM などの会員.