知識工学と人工知能　44－5
（１９８６．１．２０）

設計支援システムにおける設計対象の階層的表現法とその応用

久野　巧＊，松尾健治＊＊，中島秀之＊，田村佳彦＊，諏訪　基＊
（＊電子技術総合研究所，＊＊安川電機）

１．はじめに

筆者らは設計を人間の典型的な知的活動として捉え，そのすべての段階を包括的に支援することが可能な設計支援システムの研究を進めている［1］［2］。設計対象はＶＬＳＩ（Very Large Scale Integrated circuits）である。この設計対象の適切な表現は設計者と設計支援システムとの円滑な対話を促し，設計者の設計活動を効果的に支援するための必要条件になる。ここではＶＬＳＩを設計支援システムの中に表現する方法について考察する。

ＶＬＳＩ設計の特徴は取り扱う情報量が多いことである。この性質に対して設計者は，伝統的な多重レベルでの抽象化やいわゆる階層化という手法をよく利用する。設計者はこのような手法を用いるとき設計対象に対してある種のイメージを思い浮かべる。このイメージと等価な（あるいはそれを含む）表現を設計支援システムがもつことは，人間と機械とのインターフェイスを考えるうえで極めて重要である。設計対象の表現を知識表現の一例と考えるならば，ここで述べる表現法は知識が大量でそれらの関係が複雑な場合の扱いについての知識表現形式に関する考察にもなると考えられる。

本稿では，まずＶＬＳＩの設計過程を概観し，その後で設計対象の表現としての概念モデルとその応用例について述べることにする。

２．ＶＬＳＩ設計の特徴

大規模システムを人手によって手際よく設計するためには階層化設計手法の導入が不可欠であり，何十年も前に先人の作り上げた伝統的階層が今でも重要な役割を果たしている。ユニット，マザーボード，プリント基板およびトランジスタという初期の階層は，トランジスタレベルが技術革新によってＭＳＩやＬＳＩに変化してきたが，実装上の境界を反映したこの階層そのものは今もなお存在している。

従来の設計は一般に方式設計，機能設計，回路設計および実装設計に分けられる。方式設計は実装技術を考慮しながらシステムの基本的な構成や性能を決定し，機能設計はそれをもとにしてデータパス部や制御部を詳細化する。回路設計は基本論理回路などの接続関係を定め，実装設計でプリント基板での配置あるいは集積回路内の配置を決める。このように従来までの設計は実装技術およびそれを反映した階層の影響を方式設計の段階から強く受けている。

このような状況は，設計の対象であるシステムがひとつのＶＬＳＩチップで実装可能になると一変する可能性がある［3］。まず，実装上の階層が一旦取り払われ，改めて設計者が階層を自由に設定できるようになる。そのため機能設計において実装上の制約を考慮せずに決定した機能ブロックごとの分割をそのまま実装レベルでの階層とすることが可能となる。次に，チップ製造後のバグの検出や修正が困難になるので，設計の正しさの保証が必要になってくる。検証やシミュレーションによる動作の確認により，設計段階で誤りを完全に除去しなければならなくなる。

３．設計対象の表現としての概念モデル

３．１　概念モデル

ある対象（システム）に関する人間の思考過程を考察するとき，'対象'そのもの，対象の'概念モデル'，人間の'思考モデル'およびその思考モデルを'概念化'したものの４つが重要である。Normanの定式化［4］に従えば，対象を t とすると，対象の概念モデルは C(t)，t に対する利用者の思考モデルはM(t)となる。利用者が実際どのようなM(t)を持つかは心理実験や観察などを通して推定する。そのようにして得られたモ

デルはM(t)ではなく，さらに概念化したC(M(t))である＊。また，概念モデルC(t)はｔの設計者によるモデルであるため，利用者がそれを使うことはない。利用者に対しては利用者の特性を考慮し，概念モデルC(t)に基づいて再構成されたシステムイメージが提示される。

設計過程の場合は上述の定式化を少し変更しなければならない。設計は設計対象ｔの実現のためにその概念モデルC(t)を構築することであるから，設計の途中ではｔが存在しない。設計過程を定義し直すならば次のようになる。

［設計過程の定式化］：設計の最終的な目標であってまだ存在しない仮想的な設計対象をＴとすると，Ｔに対する設計者の構想である思考モデルはM(T)，Ｔの計算機上の表現である概念モデルはC(T)，Ｔのドキュメントとしてのシステムイメージは$C_m(C(T))$＊＊となる。よって，設計とはM(T)に基づき，設計支援システムと断片的な$C_m(C(T))$をやりとりしながら，C(T)を構築することである。□

理想的には，M(T)と$C_m(C(T))$はC(T)に一致するが，通常は人間の情報処理の特性（忘却，錯誤，不完全な推論など）により必ずしも一致しない。概念モデルは設計支援システムの中にただひとつ存在する，設計対象の表現であり，設計作業はこのモデルの対する操作が主になる。

概念モデルC(T)はＴが大規模になればなるほど複雑になるので，設計者が直接それを操作し，その全体を把握することは難しい。設計支援システムは設計者にC(T)をそのまま示すのではなく，Ｔを理解しやすくするために適切な表現に変換する。その場合，C(T)は設計者（利用者）の目に直接触れることはないので，人間とのインターフェースを考慮せずに表現上最適な形式を採用することができる。設計支援システムの役割として，この概念モデルC(T)の保存と一貫性の管理，C(T)の$C_m(C(T))$への変換および$C_m(C(T))$からC(T)への変換が重要である。

概念モデルは１つの設計対象に関する複数の抽象レベルでの記述から構成される。重要な抽象レベルとして，アルゴリズム，レジスタトランスファ，ゲート，サーキット，レイアウトがある。各抽象レベルでの表現を忠実に概念モデルに反映させる方法［５］と代表的なある抽象レベルでの記述とそれに対する変換規則の集まりで概念モデルとする方法［６］がある。設計者からみた概念モデルとしてはこの２つは等価であるため，原理的にはこれらを区別する必要性はない。しかし，実際の設計支援システムにおける設計者との対話特性や概念モデルを操作する部分との関連などから，概念モデルの内部構造についても考慮しなければならないこともある。例えば，概念モデルに対するアクセスタイムを考慮する時には内部構造を上述の方法の前者をとるか後者をとるかは大きな問題となる［７］。

３．２　設計対象の全体部分階層と上位下位階層

１つの設計対象に対して複数の抽象レベルでの表現が存在する。従来は各抽象レベルごとに種々の形式で記述され，それぞれに対応する数多くのハードウェア記述言語が開発された［８］。また，独立に開発された抽象レベルごとの言語において，シンタックス上の対応を明確に規定することにより，それらを統合した例もある［９］。

これまでの設計対象の表現法に共通したものとして，特定の抽象レベルにおける記述をブラックボックスの

＊　思考モデルの概念化は思考モデルのモデルを構成することである。対象を計算機上にモデル化したものが概念モデルであるように，思考モデルを計算機で扱えるように表現することがその思考モデルの概念化である。概念化された思考モデルの存在は設計支援システムに重要であるが［１］，本稿では設計対象の表現としての概念モデルに注目して議論を進め，思考モデルおよびその概念化については扱わない。

＊＊　システムイメージＣｍ（Ｃ（Ｔ））は広義の仕様記述である。設計支援システムへ入力（あるいは出力）される，設計対象Ｔに関する情報のすべてを指す。Ｃｍ（・）は人間の思考過程を考慮した概念化を意味する。具体的なシステムイメージとしては従来のハードウェア記述言語によるＴの記述やディスプレイ端末に表示された回路図やレイアウト図などが考えられる。

境界での振舞という観点から階層化するという技法がある。この階層に関してほぼ自明で重要なことが3点ある［3］。

(1) 構成要素は，それを部分とする，より大きな構成要素と同じ種類のものである。

(2) 各構成要素は，それを合成(synthesis)あるいは分析(analysis)するときに使われる外部記述とより簡単な構成要素の集まりを示す内部記述がある。

(3) 設計対象は抽象レベルごとに異なる階層をもつ。

ここでは概念モデルの中に現われる上述の階層を全体部分階層と呼ぶ。全体部分階層は不要な詳細情報を隠蔽し，分割統治法(divide and conquer)を用いた設計の簡略化を可能にする。分割統治法は問題を解くためによく使う方法である。与えられた問題をより小さな部分に分割し，各部分の解を見いだし，その部分解を結合して全体の解を得る。この方法は部分問題がもとの問題の縮小版として現われるとき，すなわち部分問題のサイズの和を小さく保てるならば有効である。しかし部分問題に関連があって分割してもサイズが小さくならないときには計算量は膨大なものとなる。残念なことには，ほとんどの設計問題はその部分問題が独立でないことが多い。従って，できるだけ独立性の高い部分問題への分割という新たな問題が生じてくる。

これに対処するために設計者が用いる伝統的方法は，機能単位に分割することである。機能として各抽象レベルに共通なものを利用すれば，それが各レベルでの分割に対する指針になることは容易に想像できる。このようにして分割された，各抽象レベルでの部分問題（あるいはその解）もまたある階層構造をつくる。この階層は異なる抽象レベルにおける同一の機能に対する表現間の対応関係を表わしている。概念モデルの中に現われるこのような階層を上位下位階層と呼ぶことにする。

3．3　機能，動作および構造

機能あるいは動作という用語は設計のいろいろな場面で使われ，あまり明確にはその意味が定まっていない［10］［11］。ここでは混乱を避けるために機能，動作および構造という用語を定義し直し，以後この定義に基づいた意味で用いることにする。

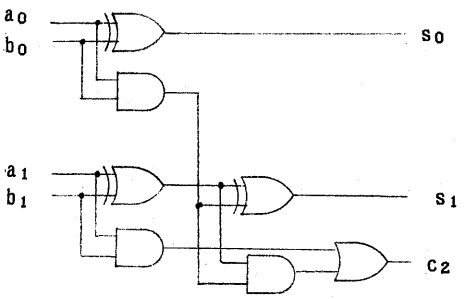
$$S = A + B$$

(a) レジスタトランスファレベルでの動作記述

a_1	b_1	a_0	b_0	c_2	s_1	s_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
		…			…	
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	0

あるいは，この真理値表と等価な論理式

(b) ゲートレベルでの動作記述



(c)ゲートレベルでの構造記述

図1　加算器のレジスタトランスファレベルとゲートレベルでの記述

機能（Function）とは，相互接続された部分からなる全体において，各部分の果たすべき役割および設計者が意図した全体の働きをいう。機能は次に定義する'動作'を抽象化し，'構造'を意味付けたものである。

動作（Behavior）とは，対象の入力に対する観測可能な出力の関係をいう。対象が'構造'をもつとき，その対象の動作は対象を構成する各部分の動作の組み合わせとなる。動作の本質的な属性に注目すれば（動作を抽象化すれば），それが'構造'の'機能'を表わす。

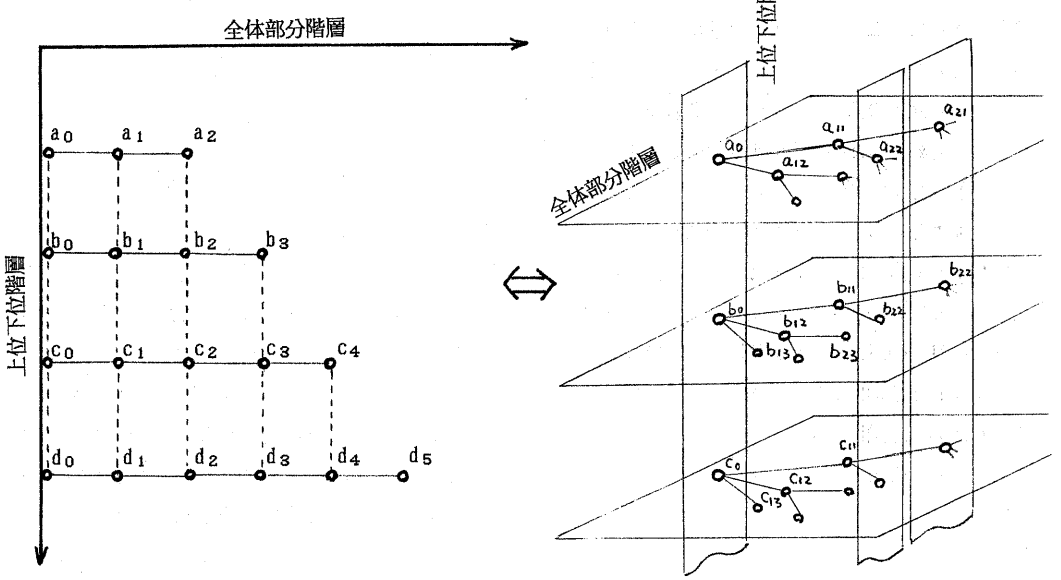
構造（Structure）とは，相互接続された部分からなる全体において，各部分およびそれらの相互接続をいう。構造は'機能'を実現したものであり，特定の入力に対する'動作'をもつ。

例えば，図1に示す記述を考える。(a)はレジスタトランスファレベルでの記述，(b)はゲートレベルでの動作記述，(c)はゲートレベルでの構造記述である。(c)はゲートレベルでのプリミティブ（Primitive）であるANDゲートやORゲートあるいは複合要素XORゲートとそれらの接続を表わしている。(b)は(c)の動作を表わしており，通常はプリミティブの動作とその接続情報を組み合わせて生成される。(a)は(b)の動作に対して入出力に意味を与え，ビット幅に関する情報を捨象した記述，すなわち加算器の機能を表わしている。

従来のハードウェア記述言語では，(a)と(b)の違いが明確でなく，2つをまとめて機能記述あるいは動作記述と呼んでいた。これは抽象レベル間の関係をあいまいにするものであり，抽象レベル間の変換である合成や分析を困難なものにする（5章の階層化設計の項を参照）。上で定義したように，構造に対する動作と機能を分離して抽象レベル間の対応を明確にすれば，ある設計対象に対する複数の抽象レベルでの記述を設計者の統一された概念モデルとして扱う基盤を与えたことになる。

3．4　概念モデルの上位下位階層と全体部分階層

概念モデルにおける上位下位階層と全体部分階層の関連を図2に示す。図2(a)の縦軸は上位下位階層，横軸は全体部分階層の深さを表わす。例えば，図2(a)の$a_0-a_1-a_2$や$b_0-b_1-b_2$は図2(b)のような全体部分階層を意味し，$a_0-b_0-c_0$や$a_1-b_1-c_1$は上位下位階層を意味する。ここで，添字は全体部分階層の深さを表わす。すなわち，$a_1-b_1-c_1$は$a_{11}-b_{11}-c_{11}$と$a_{12}-b_{12}-c_{12}$の2つの上位下位階層をまとめたものを示す。特に上位下位階層$a_0-b_0-c_0$は異なる抽象レベルでの表現が実は同じ設計対象であることを示している。ここで図2(b)のa_0，a_{11}，a_{12}，b_0，b



(a)概念モデルの模式図　　(b)概念モデルにおける上位下位階層と全体部分階層

図2　概念モデル

$_{11}$，b_{12}などをノードと呼ぶことにする。

ノードはある抽象レベルにおける設計対象の特定部分の動作と構造を表わしたものである。あるノードに対して，抽象レベルのより高いノードを上位ノード，より低いノードを下位ノードという。上位ノードは下位ノードの機能を表わすものであり，，下位ノードは上位ノードを具体化したものである。また，あるノードに対して，それを構成要素とするノードを拡大ノード，それの構成要素となるノードを部分ノードという。例えば，b_{12}に対して，a_{12}を上位ノード，c_{12}を下位ノード，b_0を拡大ノード，b_{23}を部分ノードという。

設計対象の概念モデルとして図2のように上位下位階層と全体部分階層を表わしたとき，図中のすべてのノードは原則として両階層に属する。すなわち，この概念モデルは各ノードに関して上位ノードと下位ノードの関係である上位下位階層および拡大ノードと部分ノードの関係である全体部分階層のすべてを保持していることになる。

次に概念モデルにおける各ノードの機能，動作および構造の関連を述べる（図3参照）。まず，各ノードは原則として動作記述と構造記述をもつ（例外はプリミティブと呼ばれるノードで構造記述がなく動作記述だけをもつ）。動作記述と構造記述は表裏一体の関係にあり，ノードの外部記述と内部記述にそれぞれ対応

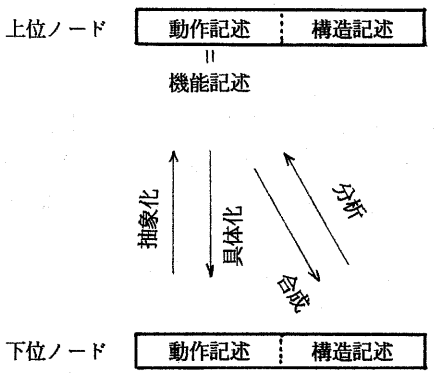


図3　上位ノードと下位ノードの機能，動作および構造の関係

する。また，あるノードの機能記述とは3．3の定義によりその上位ノードの動作記述をいう。つまり，機能記述は単独では存在するものではなく，上位ノードの動作記述の別名である。

各ノードは，(1)動作記述をもつ，(2)構造記述があってその動作を構成できる，(3)上位ノードの動作記述との対応がある（機能記述をもつ），のいずれかである。各ノードは何らかの動作記述と関連があるので，それに入力を与えて評価することにより，出力を得ることができる。このようなノードの特性を'動作可能'であるという。また，動作可能なノードからなる概念モデルを動作可能なモデルという。

概念モデルは設計の進行に伴って変化する。設計の初期には図2で示すノードのいくつかに断片的記述が設計者から仕様として与えられる。その後，合成や分析が繰り返され，徐々に上位下位階層と全体部分階層が出来上がる。概念モデルのすべてのノードが確定した状態を設計の終了という。

4．応用

設計対象の機能，動作および構造を上位下位階層と全体部分階層とで表現した概念モデルの目的はいうまでもなく設計支援システムへの応用である。そのときの概念モデルは設計支援システムの重要な構成要素であり，設計対象を表現する唯一のものとなる。この章では設計支援システムの中でどのように概念モデルが利用されるかを示す例として，複数の抽象レベルでモデルを動作させる方法（ミックスレベルシミュレーション，mixed-level simulation)を説明する *。

4．1　ミックスレベルシミュレーション

概念モデルにおけるミックスレベルシミュレーションとは，ある抽象レベルでの設計対象（全体あるいはその一部）の動作を観測するときに特定の部分につい

* 現在，概念モデル上でのレジスタトランスファレベルとゲートレベルのミックスレベルシミュレーションおよびレジスタトランスファレベルの動作記述から構造記述の生成，最適化，検証などがインプリメントされている［12］。

ては別の抽象レベルで扱うことをいう。例えば，図4の太線で示すようにノードの動作記述を利用するとき，ほとんどのノードはレジスタトランスファレベルで動作させるが，ノードr21はゲートレベルのノードg21，r22はサーキットレベルのノードc22で動作させることをいう。

ミックスレベルシミュレーションの目的は，設計者が注目する部分の動作確認，およびそのための動作環境の設定である。つまり設計対象が実際に稼動している状態を作り出し，その中で任意箇所の観測を任意の詳しさで行うことがねらいである。このようなシミュレーションが実行できる理由は概念モデルのすべてのノードが動作可能だからである。

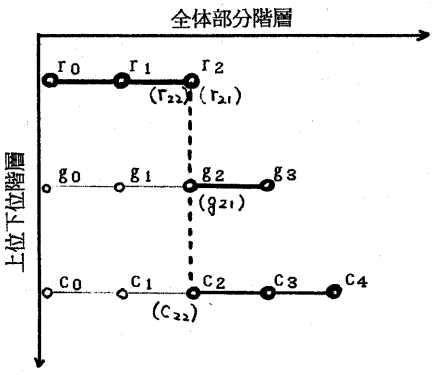


図4　ミックスモードシミュレーションの概念モデル

4．2　加算器のシミュレーション例

概念モデルのすべてのノードは動作可能なため，レジスタトランスファレベルだけ，あるいはゲートレベルだけでシミュレーションすることができる。ここでは，図1のような加算器を表わすノードはゲートレベルで扱い，他のノードはすべてレジスタトランスファレベルで扱う例について説明する。

全体部分階層における構成要素間の接続線は抽象レベルによって異なる信号を伝播する。レジスタトランスファレベルでは，接続線はデータパスや制御線を意味するのでその上を流れる信号は整数型データや論理型データである。また，ゲートレベルでは接続線上を伝わる信号は論理型データだけである。従って，ミックスレベルシミュレーションを行うためにはノードの対応だけでなく入出力信号も対応させる必要がある。このような対応関係は概念モデルの上位下位階層が保持する。例えば，図1の加算器に関しては次のような関係をもつ。

```
(BETWEEN ((RT adder)(GATE adder2))
         ((IN A)(IN (a1 a0))
          (IN B)(IN (b1 b0))
          (OUT S)(OUT (c2 s1 s0))))
```

この記述はレジスタトランスファレベルのノードadderとゲートレベルのノードadder2が対応し，adderの入力信号A（B）とadder2の入力信号a0，a1（b0，b1）およびadder出力信号Sとadder2出力信号s0，s1　c2がそれぞれ対応していることを表わしている。どのノードをどの抽象レベルでシミュレートするかは設計者／利用者が指示しなければならない。シミュレータはこの指示と上記の対応関係を使って，入出力信号の変換を行う。まず，adder のA，Bをadder2 のa0，a1，b0，b1に変換する。その後でゲートレベルシミュレーションを行い，その結果であるs0，s1　c2を逆変換してSを求める。このときゲートレベルの任意のノードに観測点を設ければ加算器の振舞をゲートレベルで観測することができる。

5．従来の設計過程あるいは設計手法と概念モデル

本稿で述べている概念モデルは，実装時の全体部分階層を設計者が自由に設定できるという特徴を生かして，抽象度の高い機能の持つ階層をそのまま自然に実装レベルまで反映させることをねらっている。その意味では新しい設計手法を提案していることにもなるが，従来のそれと相容れないものであっては目的のすべてを果たせない。すなわち，概念モデルは設計対象の完全な表現であるから，一般的設計者が採り得る設計手法をすべて包含する必要がある。次に従来の代表的な

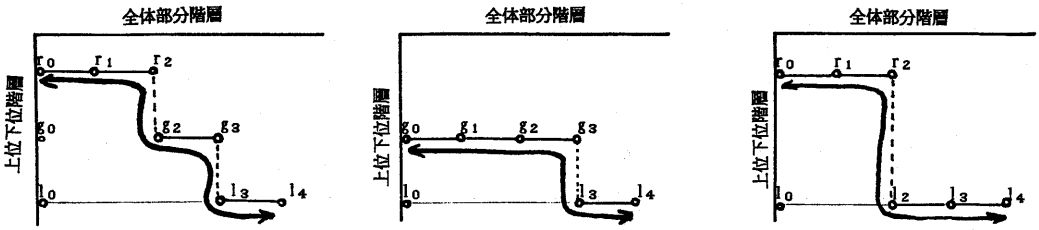


図5　従来の設計手法や設計過程と概念モデル

設計手法やそこでの設計過程が概念モデルでどのように表わされるかを見てみよう。

[階層化設計]

大規模かつ複雑な対象を設計する常套手段として従来からいわゆる'階層化設計'の手法が用いられてきた。この手法はまず仕様を高い抽象レベルで記述し，それを分割しながら同時に具体的レベルへ合成していく。例えば，レジスタトランスファレベルで仕様が与えられたとき，その仕様はまずレジスタやALUなどの機能ブロックに分割され，それらの接続を決めることで設計が進む。その後，レジスタやALUがANDゲートやNOTゲートの組み合わせとしてゲートレベルで実現される。

このような設計過程は概念モデルでは図5(a)のように表わされる。すなわち，(1)レジスタトランスファレベルで$r_0-r_1-r_2$を生成する（仕様を機能ブロックに分割），(2)r_2を具体化し，g_2を構成する（レジスタやＡＬＵの入出力インターフェースをゲートレベルで決定），(3)ゲートレベルをg_2-g_3と進み，g_3をANDやNOTに対応付ける（レジスタ，ALUのゲートレベルでの実現），という設計作業を行うことに相当する。

いわゆる階層化設計は本稿でいう概念モデルの全体部分階層と上位下位階層の一部をトレースする手法である。この手法の欠点は機能分割のときに分割されたブロックの明確なインターフェースを仮定しなければならないことである。このインターフェースはブロックの設計が終了しなければ決定できないことであるから矛盾である。設計者は過去の経験などをもとにして何らかの方法でインターフェースの適否を決定しなければならない。階層化設計は確かにＶＬＳＩのように複雑な対象の設計に有効だが，設計の最も重要な部分を見落としているように思われる。

[ゲートアレイ]

ゲートアレイによる設計の目的は，従来の論理設計と実装設計という区分けにおける，論理設計者の実装設計からの解放である。論理設計者はゲートレベルまでの合成を担当する。ゲートアレイによる設計を概念モデルを表わすと図5(b)のようになる。ゲートレベルの全体部分階層は一般には利用されず，ゲート回路とその接続関係という平坦な記述のままでレイアウトレベルに写像される（$g_0-g_1-g_2-g_3$とl_0-l_1）。

[シリコンパイラ]

シリコンパイラはレジスタトランスファレベルからレイアウトレベルへの直接変換を目指している。仕様を機能ブロックの集まりとして記述するとそれに対応するレイアウトレベルでのブロックを組み合わせることで設計を行う。概念モデルでは図5(c)のように表わされる。レジスタトランスファレベルでの全体部分階層$r_0-r_1-r_2$とr_2に1対1対応のl_2-l_3から，レイアウトレベルでの全体部分階層$l_0-l_1-l_2-l_3$を合成する。

6．おわりに

設計支援システムの中で重要な位置を占める設計対

象の表現について説明した。設計対象の完全かつ無矛盾な表現が概念モデルである。概念モデルは上位下位階層と全体部分階層からなり，任意の抽象レベルの設計対象を任意の細かさで表わすことができる。また，概念モデルにおける機能，動作および構造を明確に定義したので，各種の設計問題（合成，分析，検証，最適化など）が概念モデルを基礎にして議論できるものと思われる。一例として挙げたミックスレベルシミュレーションは動作可能な概念モデルの特性を利用している。

今後は設計過程に沿った概念モデルの変化に関する検討と全体的なインプリメンテーションが課題となる。

謝辞

研究の機会を与えていただいた当所電子計算機部柏木寛部長に感謝します。また，ご討論いただいた人間機械システム研究室の諸氏に感謝します。

参考文献

［1］諏訪基，他，"マン・マシンシステムへの思考モデルの導入"，情報処理学会第２８回全国大会予稿集4G-11(1984).

［2］田村佳彦，他，"設計支援システムにおける思考過程の構成"，情報処理学会第２８回全国大会予稿集4G-10(1984).

［3］ Allen, J., Penfield, P.JR.,"VLSI Design Automation Activities at M.I.T.",IEEE Trans. Circuits and Systems, vol.CAS-28, no.7, 645-653 (1981).

［4］Norman, D.A.,"Some Observations on Mental Models", MENTAL MODELS, ed. Gentner and Stevens, LAWRENCE ERLBAUM ASSOCIATES, 7-14(1983).

［5］久野巧，他，"設計支援システムにおける設計対象の階層的表現方法"，情報処理学会第２９回全国大会予稿集7N-7(1984).

［6］中島秀之，他，"多段階の抽象化を許す設計対象の記述"，情報処理学会第２９回全国大会予稿集7G-8(1984).

［7］Rosenbloom, P.S., Newell, A.,"The Chunking of Goal Hierarchies : A Generalized Model of Practice", Proc. of International Machine Learning Workshop, 183-197(1983).

［8］van Cleemput, W.M.,"Computer Hardware Description Languages and their Applications", Proc. 16th Design Automation Conf., 554-560(1979).

［9］Borrione, D., Paou, C.L.,"Overview of the CASCADE Multi-level Hardware Description Language and its Mixed-mode Simulation Mechanisms", Proc. 7th CHDL, 239-260(1985).

［10］Walker, R.A., Thomas, D.E.,"A Model of Representation and Synthesis", Proc. 22nd Design Automation Conf, 453-459(1985).

［11］Gajski, D.D.,"Silicon Compilation", VLSI SYSTEMS DESIGN, 48-64(1985).

［12］松尾健治，他，"論理設計における回路の自動合成および回路の最適化のための一手法"，設計自動化研究会報告86-DA-30(1986).