ＩＣチップ内部のグラフ表現による自動割付けシステム

新井 浩志　　門倉 敏夫　　深沢 良彰
（早稲田大学）　（相模工業大学）

１．はじめに

近年、高級言語によるハードウェアの記述から、直接VLSIのマスク・パターンを生成するシリコン・コンパイラ[1]が研究されてきている。これは、VLSIの大規模化と、多品種少量生産の２つの理由により、人手による設計が事実上不可能になりつつあるためである。

しかし、VLSIの開発能力を持たない場合や試作品を作る場合などは、ハードウェアの記述から合成された論理をもとに、既存のICチップを組み合わせて所望の論理を得る必要がある。そこで、ICチップの情報をライブラリに保存しておき、合成された論理に割付け、ハードウェアを作成するCMU-DAシステム[2]-[3]などが開発されている。CMU-DAシステムでは、TTLまたはCMOSセルを機能別に分類、登録しておき、合成された論理に割付けている。

これに対して本システムでは、ICチップの内部の機能を有向グラフとして表現してライブラリに保存しておき、合成したハードウェアを表わすグラフの部分グラフとの同型性を調べることにより自動割付けをおこなう。この方法によって、どのような機能のICチップをも登録することが可能である。

ここで、次の２点が問題となる。

(1) ある機能のグラフ表現は、一意的に決定されるとは限らない。

(2) グラフの同型性を調べる問題は、NP完全問題[4]であり、多くの計算時間を必要とする。

そこで本システムでは、ライブラリに登録するチップと合成された論理を、一定の規則に従って統一し、(1)の問題の解決をはかっている。また、(2)の問題に対しては多くのアプローチ[5]-[7]がおこなわれているが、いずれも決定的な方法とは言いがたい。そこで本システムでは、グラフを木に変換し、同じ深さにあるノードの間の距離にもとづいて同型性を調べるという方法を新たに開発して用いている。この方法の利点は、非同型である場合にそのことを早く発見できるということである。本システムでは、チップを表わす多くのグラフの中から、数少ない同型をしたグラフを選び出すことが要求されているので、非同型なグラフを早く発見できることは、全体の効率に大きく影響を与える。また、ICチップの機能の一部だけを用いる場合にも、木を単位として調べれば良い。現状では、複数組の回路が実装されているチップも多いので、その有効な活用は不可欠である。

２．システムの概要

図１に本システムの構成を示す。

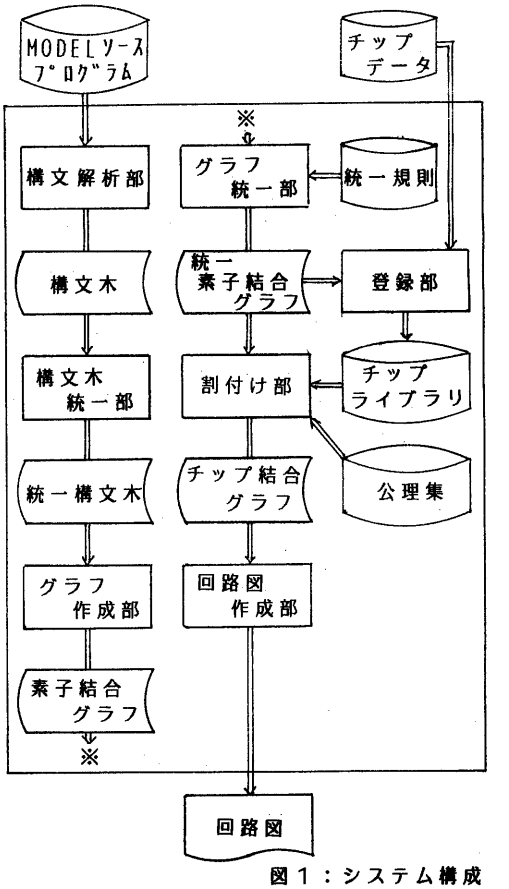


図１：システム構成

```
CPU: PROCEDURE ( ACC )  ;
     DECLARE  ACC(0:15)                                    REGISTER  OUT ;
     DECLARE (CAR(0:9),MAR(0:9),IR(0:15))                  REGISTER;
     DECLARE (OP(0:5)=IR(0:5),ADDR(0:9)=IR(7:16))  REGISTER;
     DECLARE  M(0:1023,0:15)       MEMORY ;
@ADS:      MAR=CAR ;
           CAR=CAR+1 ;
@IFT:      IR=M(MAR,*) ;
@LAD:      MAR=ADDR ;
@OP :      SELECT( OP ) ;
               WHEN(   4)  ACC=M(MAR,*) ;
               WHEN(   8)  M(MAR,*)=ACC ;
               WHEN(  16)  ACC=ACC+M(MAR,*) ;
               WHEN(  32)  CAR=ADDR ;
               WHEN(  33)  IF ACC(0)=0B THEN CAR=ADDR ;
           END ;
           GO TO @ADS ;
END CPU ;
```

図２：MODEL記述例

２－１．構文解析部

ハードウェア記述言語MODEL[8]によるハードウェアの記述を入力とし、構文解析をおこない、構文木を出力する。ハードウェア記述言語MODEL は、構造的プログラミングの概念を取り入れ、LSI やICを用いたハードウェアの設計を積極的に支援することを目的に設計された言語である。

２－２．構文木統一部

同じ機能は同じ構文木で表わされるように、構文木に部分的な変更を加える。大きく次の２つの処理をおこなう。

(1) 同意語の整理

(2) 論理式の積和形への展開

この(1) は、IF文のネストをSELECT文で置き換えるなど処理であり、(2) は、括弧をもちいれば多様な表現が可能な論理式の表現方法を一意的に決定するための処理である。

２－３．グラフ作成部

統一構文木をもとに論理合成をおこない、素子結合グラフを作成する。素子結合グラフは、ノードが素子や信号点を、アークがデータの流れを表わしている。このノードには、 AND、ORなどのゲートから、加算器、メモリなどの、機能の高いものまでを含む。アークは、ビット巾に関する情報を持ち、出力側から入力側への向きを持たせる。また、マルチプレクサなどのように、複数の性質の異なった入出力を表わすアークは、それらを区別する。

```
IF A = B THEN E = 0;
   ELSE E = C & D;        (a)
```

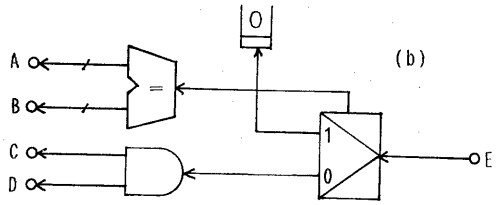


図３：素子結合グラフの作成
(a) MODELソース
(b) (a)に対応する素子結合グラフ

２－４．グラフ統一部

サブグラフの同型性を調べることによりICチップの割付けをおこなう場合、同じ機能であるにもかかわらず異なったグラフで表現してあると、割付けが不可能となる。そこでグラフ統一部では、構文木の段階では不可能だった統一化の処理をおこなう。すなわち、素子結合グラフの一部を、統一規則に従って変更し、割付けをおこなうハードウェアを表現したグラフとライブラリ中のICを表現したグラフにおいて、同じ機能は同じグラフで表現されるようにする。

統一規則は、変更前と変更後のグラフの対で構成されており、以下の３つに分類される。

(1) 低い機能のノードを高い機能のノードで置き換える。

(2) 高い機能のノードを低い機能のノードで置き換える。

(3) 同じ機能のノードで置き換える。

ただしここで、 (1)は、ノード数が増加しない場合、 (2),(3)はノード数が減

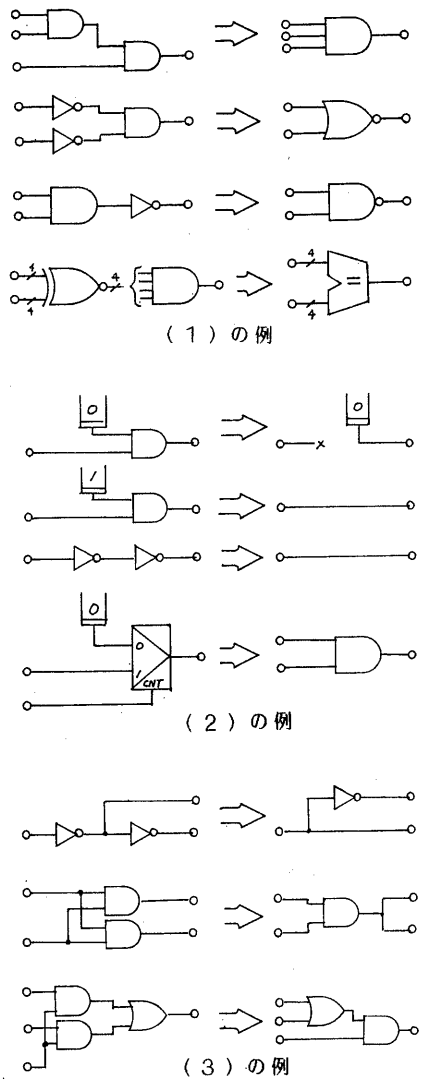


図４：統一規則

少する場合に限られる。

なお、(2) の例の１番目のように、どこからも参照されないノードが生じた場合には、このノードから入力側の参照されないノードをすべて削除する。統一規則は、ノード数の多いものから適用していく。

この統一化の処理は、一般の論理合成システムでおこなわれている最適化の処理とほぼ同一であるが、Fan-in、Fan-out などは考慮せず、ノード数の少なくなる方向におこなう。

２－５．登録部

グラフ統一部から出力された統一素子結合グラフが、新しく追加するICチップを表現したものである場合に、登録データと共にライブラリに登録する。登録データは、以下の２つから構成される。

(1) 割付け制約条件
- すべての入出力ピンの間の遅延時間
- 各入出力ピンの最大入出力電流

(2) 割付けコストデータ
- チップ面積
- 最大消費電力

この(1) は、ハードウェアをICチップに割付けたときに、遅延時間およびFan-Out 制限を満足させるために用いる。また、割付け部において、(2) の総和を最少にするような割付けをおこなう。

２－６．割付け部

グラフ統一部から出力された統一素子結合グラフG_H が、自動割付けするハードウェアを表現したものである場合に、チップライブラリの中の統一素子結合グラフG_L との、サブグラフの同型性を調べることによりICチップへの割付けをおこなう。この割付けのアルゴリズムは、次の３．で詳しく述べる。

３．割付けアルゴリズム

本システムで用いている割付けアルゴリズムは大きく次の２つの段階に分けられる。

Step1. G_H を構成するために最適なG_L の集合をノードの集合だけに注目して求める。

Step2. G_L の集合から一つずつ、G_H のサブグラフと同型であるかを調べる。

このStep1 で失敗した場合には、チップライブラリの中のG_L には含まれない特別な素子ノードがG_H に存在することになる。そこで、そのノードを公理集に基づいて展開する。

また、Step2 で失敗した時はG_L の最も入出力に近いノードで不一致が発見された場合に限りそのノードを展開して再び同型性を調べる。

公理集は、先の統一規則と同様に、統一素子結合グラフの対で構成されており、統一規則とは逆の方向への変換をおこなう。ただし、図４の(2) の例のようなものは含まない。さらに図５のように、データの巾を合わせるための公理を含んでいる。

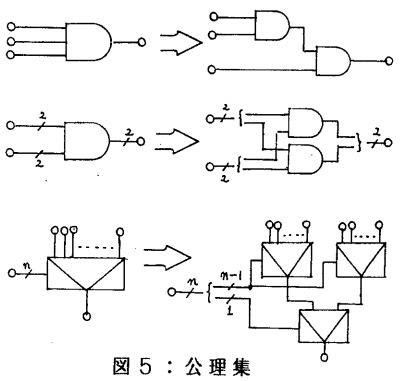


図５：公理集

３－１．Ｓｔｅｐ１

ライブラリ中の各G_Lに含まれるノードの型と数に注目して、線形計画法により、最適なチップの集合を求める。ノードの型は次のように分類し、各々連立不等式をたてる。

(1) 素子ノード

ゲート、マルチプレクサなどで、２入力と３入力のANDは別の型として扱う。ｉ番目のG_Lに含まれるα型の素子ノードの数をn_i、G_Hに含まれるα型の素子ノードの数をｎとすると、すべての型に対して

$$\sum_i n_i \geq n \qquad (1)$$

(2) 信号ノード

信号ノードは、入力、出力、内部の３つに分けて考える。ｉ番目のG_Lに含まれる入力信号ノード、出力信号ノード、内部信号ノードの数をI_i、O_i、S_iとする。また、G_Hに含まれる数をＩ、Ｏ、Ｓとすると、

$$\left.\begin{aligned} \sum_i I_i &\geq I \\ \sum_i O_i - \{\sum_i I_i - I\} &\geq O \\ \sum_i S_i + \{\sum_i I_i - I\} &\geq S \end{aligned}\right\} \qquad (2)$$

以上の式を拘束条件として、登録部で入力したコストデータの総和を最小にするようなチップの集合を求める。

３－２．Ｓｔｅｐ２

G_LがG_Hのサブグラフと同型であるかを調べるために、まず、各々のグラフを横型探索を用いて木に変換する。

グラフG_Hは、すべての出力端子から同時に横型探索をおこない、すでに到達済みのノードに達した時には、このノードを今通過したアークに対して分離する。図６の例では、ノードＤをアークｘに対して、ノードＧをアークｙに対して分離している。

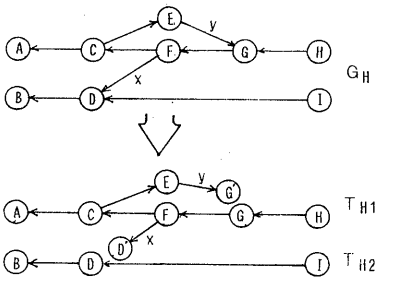


図６：G_HのT_Hへの変換

グラフG_Lは、各々の出力端子から１回ずつ横型探索をおこない、入力アークが２本以上あるノードに達した場合には、今通ったアーク以外のすべての入力アークに対してそのノードを分離する。

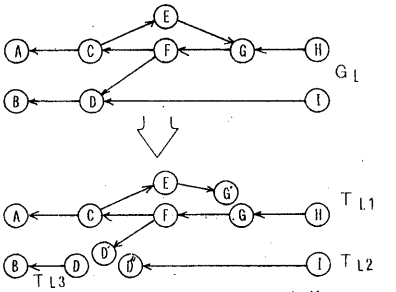


図７：G_LのT_Lへの変換

このように、グラフを木に変換することにより、ICチップの一部だけを割付けに用いる場合にも木を単位として調べれば良いという利点が得られる。また、図７のT_{L1}だけを割付けに用いるということは、ノードＦの出力が確定できないためあり得ない。一般に木T_{Li}を割付けに

用いるならば、グラフG_LにおいてTLiの根から到達可能なノードを根に持つ木TLjはすべて割付けに用いなければならないという制約条件を得られる。

そこで木T_Lの関係をグラフで表わし、入力に近い木から順にT_Hとの同型性を調べることにより、計算時間を短縮することができる。図7のT_Lに対しては、図8のようなグラフが得られる。

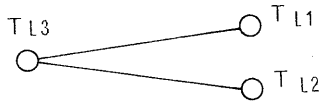


図8：T_Lの間の関係

次に木T_LがT_Hのサブグラフと同型であるかを調べるために、行列D_ℓとEを次式(3),(4)にしたがって求める。

$$D_\ell(i,j) = d(i,j)/2 \qquad (3)$$

$$E(i,n) = \sum_{j,\ell} \delta(D_\ell(i,j),n) \quad n=0,1,\cdots,\ell max \qquad (4)$$

ここで、d(i,j)は、ノードυ_iとυ_jの間の距離を、ℓmaxは深さℓの最大値を表わす。また、

$$\delta(x,y)=\begin{cases}1 & (x=y)\\ 0 & (x\neq y)\end{cases}$$

である。

そして、T_Lのノードのうち、同じ深さにあるノードの数が最も多いものから順に、これに対応し得るノードがT_Hに存在するかどうかを行列D_ℓ、Eをもちいて調べる。ここで、ノードυ_{Li}がノードυ_{Hj}に対応するためには、

$$E_L(i,n) \leqq E_H(j,n) \qquad (5)$$

が成立しなければならないことを利用する。

また、ノードの対応が1つでも決定したのちは、さらに行列Dをもちいて探索の範囲を少なくする。すなわち、ノードυ_{Li}をノードυ_{Hj}に対応させたとすると、ノードυ_{Lk}はT_Hのノードのうちで、次式を満足するノード$\upsilon_{H\ell}$に対応しなければならない。

$$D(\upsilon_{Li},\upsilon_{Lk}) = D(\upsilon_{Hj},\upsilon_{H\ell}) \qquad (6)$$

以上により基準深さのノードの対応がすべて決定したのちは、一般的な木T_Lにおいて、図10のBの部分のノードの対応は一意的に決定される。そこで次に図のAの部分に本アルゴリズムを再帰的に適用して同型性を調べる。またCの部分については、基準深さのノードと同様に行列DとEを用いて順次決定していく。

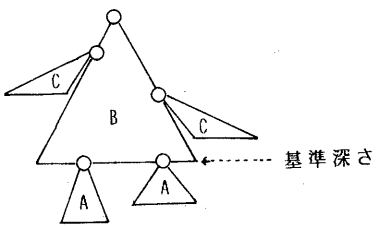
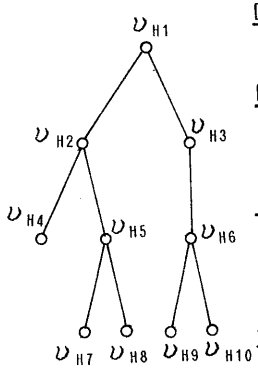


図10：一般的な木の探索

以上のアルゴリズムの効率は、どのくらい探索の枝を限定できるかに依存する。すなわち、行列E_Lの各要素が行列E_Hの各要素に近い値を持ち、行列E_L,E_Hの各行の要素が異なった値を持つ場合に式(5)による探索の限定がおこなわれる。



D_{H0}

	1
1	0

D_{H1}

	2	3
2	0	1
3	1	0

D_{H2}

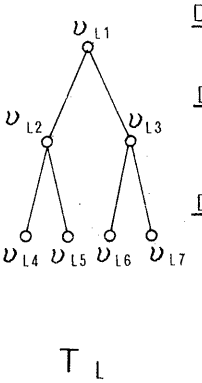
	4	5	6
4	0	1	2
5	1	0	2
6	2	2	0

D_{H3}

	7	8	9	10
7	0	1	3	3
8	1	0	3	3
9	3	3	0	1
10	3	3	1	0

E_H

	0	1	2	3
1	1	0	0	0
2	1	1	0	0
3	1	1	0	0
4	1	1	1	0
5	1	1	1	0
6	1	0	2	0
7	1	1	0	2
8	1	1	0	2
9	1	1	0	2
10	1	1	0	2



D_{L0}

	1
1	0

D_{L1}

	2	3
2	0	1
3	1	0

D_{L2}

	4	5	6	7
4	0	1	2	2
5	1	0	2	2
6	2	2	0	1
7	2	2	1	0

E_L

	0	1	2
1	1	0	0
2	1	1	0
3	1	1	0
4	1	1	2
5	1	1	2
6	1	1	2
7	1	1	2

図9：木T_H，T_Lと対応する行列D_ℓ，E

そしてこれは、木T_Lが、木T_Hに近い大きさを持ち各ノードの出次数が一定でないことを意味している。

特に、式(5)を満足しないノードがT_Lの基準深さに存在するときの計算量は、

（T_Hのノード数）
×（T_Lの基準深さ）
×（T_Lの基準深さのノード数）

であり、グラフ全体では、この不一致が発見される木が図８の根に存在するならば、

（G_Hのノード数）
×（T_Lの基準深さ）
×（T_Lの基準深さのノード数）

の計算量でG_Lを割付けに用い得ないことを決定できる。

４．計算機実験結果

図11に、木T_LがT_Hの部分木と同型であるかを調べるプログラムの実行時間を示す。n_H、n_Lは各々木T_H、T_Lのノード数であり、出次数の最大が３の木を乱数により作り入力とした。図の黒丸は、同型性を発見した場合である。図より、同型性の存在およびn_Lにはほとんど関係なくn_Hの 1.2乗に比例する計算時間がかかることがわかる。なお、使用計算機は、IBM 4341プロセッサシステムであり、オペレーティングシステムとしてはCMS を用いた。また、図11の処理時間にはタイムシェアリングによるＯＳのオーバー・ヘッド等を含む。

５．今後の課題

今後の課題を以下にあげる。

(1) 本システムでもちいている統一化の処理は、局所的なものであり、全体として最適な割付けがなされるとは限らない。そこで、構文木の段階でデータフロー解析をおこない、機能ブロックを時分割共有できるようにすることが考えられる。また、素子結合グラフのノードをすべてゲートに展開して、チップライブラリの中を参照しながら徐々に割付けをおこなうなどの方法が考えられる。

(2) 現状では、ICチップの入力の一部をHIまたはLOW に固定して割付けに用いることは考えていない。このため、

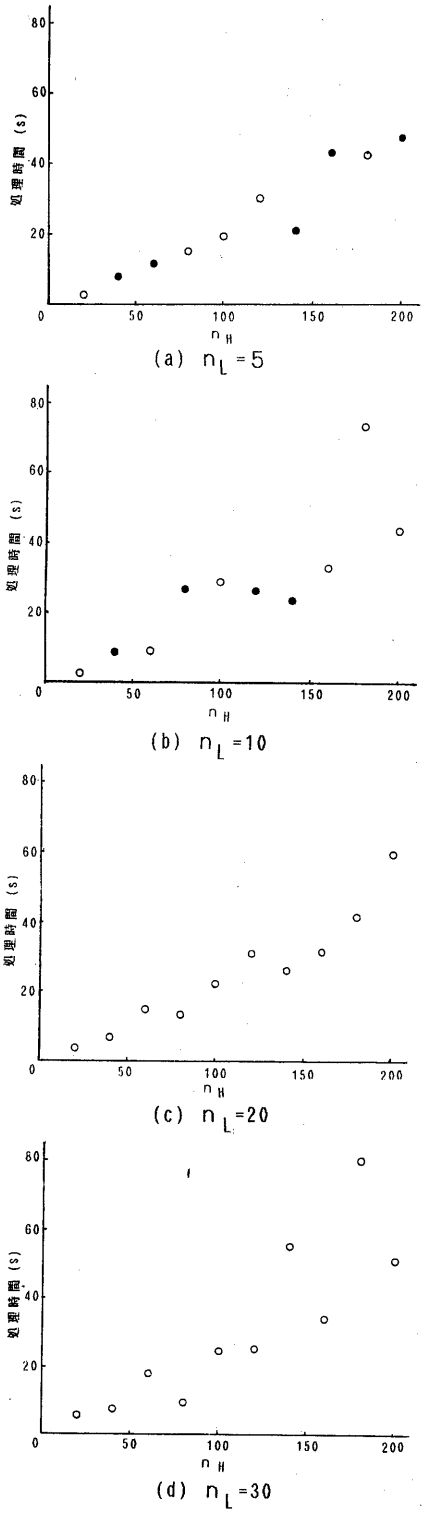


図11：計算機実験結果

ALU チップを加算器として用いることはできない。そこで、図7のT_{L3}だけがT_Hの一部と一致したときにも、ノードAの入力を固定することにより、ノードDへの入力をICの出力端子に伝達できないかを判定する必要がある。

6．むすび

ICチップとハードウェアを共にグラフで表わし、サブグラフの同型性を調べることにより自動割付をおこなうシステムについて述べた。また、割付けアルゴリズムとその有効性を示した。

最後に、本システムの作成にあたって多くの助言をいただいた日本電気の長谷川拓己氏と、計算機の使用に御協力いただいた早稲田大学情報科学センターの方々に感謝いたします。

[参考文献]

[1] J.D.Ullman : Computational Aspects of VLSI, pp.369-378, Computer Science Press, 1984.

[2] L.J.Hafer and A.C.Parker : Automated Synthesis of Digital Hardware, IEEE Transactions on Computers, vol.C-31, NO.2, 1982.

[3] G.W.Leive and D.E.Thomas, A Technology Relative Logic Synthesis and Module Selection System, 18th D.A.Conference, pp.479-485, 1981.

[4] M.R.Garey and D.S.Johnson : Computers and Intractability, A Guide to the Theory of NP-Completeness, W.H.Freeman and Company, pp.202, 1978.

[5] J.R.Ullman : An Algorithm for Subgraph Isomorphism, J.ACM, vol.23, No.1, January 1976, pp.31-42.

[6] D.G.Corneil and C.C.Gotlieb : An Efficient Algorithm for Graph Isomorphism, J.ACM, vol.17, No.1, January 1970, pp.51-64.

[7] A.T.Berztiss : A Backtrack Procedure for Isomorphism of Directed Graphs, J.ACM, vol.20, No.3, July 1973, pp.365-337.

[8] 鈴木、門倉、深沢：システム記述言語MODELについて，第22回情報処理学会全国大会、4G-12, pp.963-964, 昭和56年 3月．