設計自動化 34－5
（1986. 10. 21）

マイクロコンピュータの編集設計システムの試作

平野収三† 橋爪正樹†† 為貞建臣†
† 徳島大学工学部 †† 徳島大学工業短期大学部

試作したシステムは編集設計手法によりマイクロコンピュータを設計支援するシステムである．編集設計手法とは過去に作られた回路から利用可能な基本回路を選び出し，それに修正を加え，仕様を満たす回路を作成する手法である．本報告では，本編集設計を計算機に支援させるための回路表現法と，その表現法に基づき表現された回路を基にマイクロコンピュータを編集設計するための設計方法について述べる．

A development of CAD system for designing microcomputers

Shuzo HIRANO† Masaki HASHIZUME†† Takeomi TAMESADA†

† :Faculty of Engineering, Tokushima Univ.
†† :Techinical College of Tokushima Univ.
2-1 Minamijyousanjima-cho Tokushima-shi,770,Japan

We made a prototype of a knowledge-based system for providing interactive aid in the design of microcomputers. The design process depends on selecting and modifying the basic circuits suitable for the system specifications.

Basic circuits needed for the design are redesigned at each functional block along with the design hierarchy and connected with templates.

This report presents the representations of microcomputer circuits and the design method of microcomputers using this circuit representations. As a simple example, we show a microcomputer designed with this method.

１．まえがき

回路のVLSI化の普及に伴い設計回路が複雑化しているにもかかわらず，短期間で設計誤りのない回路を設計することが要求されている．したがって，計算機による設計支援は必要不可欠となりつつある．過去にもさまざまな設計支援システムが開発されてきたが，本稿で述べるシステムは過去に開発されたCADシステムと異なり，「編集設計」を支援するシステムである．

編集設計とは過去に作られた回路に一部分修正を加え，与えられた仕様を満たす回路を設計する方法である．実際，今まで人手で行われてきた設計は多かれ少なかれこの設計方法に基づいており，この設計手法の有用性は高い．

編集設計は必ずしも最適な回路設計を保証しない．しかしこの設計手法で論理回路設計ができれば，設計の生産性を上げることが期待できる．また，設計回路はその動作が確認済みの回路から設計されるため，実装時に十分実用に耐える回路を設計できる．さらにこの手法は回路設計時の労力を減少させることができ，最適な回路の導出が不可能な規模の大きい回路設計には有効な設計手法である．

計算機に編集設計を支援させるためには，あらかじめ回路を計算機内に格納しておく必要がある．その際，回路を系統だてて整理し格納しなければ，回路に対する編集操作が困難となる．そのため，この設計手法は回路構成がほぼ一定の回路設計に適している．マイクロプロセッサをベースとする回路，つまりマイクロコンピュータはそれ以外の回路に比べ容易に整理できる．そこで，マイクロコンピュータを設計対象とする編集設計支援システムの試作を行った．

２．では編集設計手法に基づいたマイクロコンピュータの設計方法を，３．では試作した設計システムを述べる．

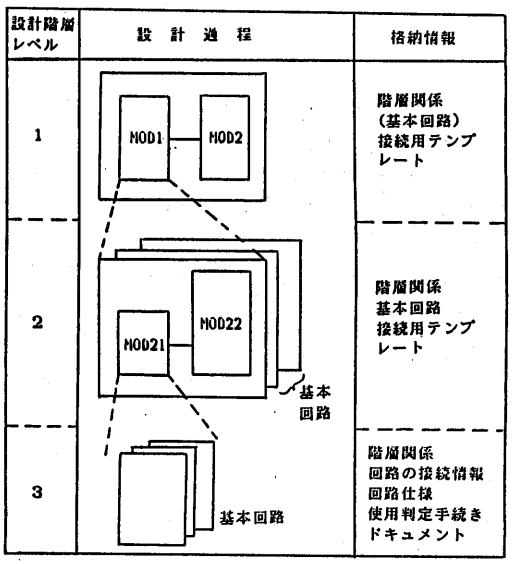
２．編集設計によるマイクロコンピュータの設計

編集設計とは過去に作られた回路から利用可能な回路（ここでは「基本回路」とよぶ）を選び出し，それに修正を加え，仕様を満たす回路を作成する手法である．本稿で述べる設計システムは編集設計を支援するためのものであり，マイクロコンピュータを設計できるよう回路構造をあらかじめ整理・登録し，また設計過程の単純化を行ったものである．なお，基本回路は編集設計の基になる回路であり，基本回路に編集操作を行い設計する手法を編集設計という．

編集設計を行うには基本回路だけでは不十分で，回路の階層関係に関する情報を必要とする．この情報をここでは「設計階層」とよぶ．

基本回路は機能ブロックごとに登録する．そのため，表１に示すように回路を機能ブロックごとに分割する．さらにそれを小さな機能ブロックに分割してゆく．これ以上分割できない回路レベルには基本回路を登録しておく．表１は３段階の回路レベルに分割される回路例を示している．

表１　本システムの編集設計手法



設計は計算機内に格納した設計階層に基づきトップダウン設計とボトムアップ設計を繰り返しながら進める．たとえば，表１の設計レベル１でMOD1の設計をするためレベル２の設計に移り，さらにMOD21の設計をするためレベル３の設計に移る．最下位の設計レベル３ではそこに格納された基本回路から仕様を満たす回路を選び，設計レベル２に戻る．設計レベル２では他のMOD22を設計するため下位レベルの設計に移る．このように本システムでの設計は設計階層を上下しながら機能ブロックごとに進めて行く．

設計レベル３ではどの基本回路が仕様を満たすのか判定しなければならない．それを計算機で支援するため，設計レベル３では接続情報だけでなく，基本回路の回路仕様（以後「基本回路仕様」という）も格納しておく．設計時には計算機により設計仕様を満たす基本回路を基本回路仕様を基に検索する．本システムのように回路仕様を用いずに，回路をハードウエア記述言語で表現し，その回路動作が設計仕様を満足するかを調べる方法も考えられる．しかしそれにはシミュレータが必要となり，設計システム自身が複雑化してしまい，マイクロコンピュータのような規模の設計が困難になる可能性がある．そこで，本システムでは回路仕様を利用して設計を行う．しかし，回路仕様だけでは厳密な基本回路の選択が行えない場合が存在するため，使用できるかどうかを調べる手続きも登録できる．

それでも設計仕様を満たす基本回路が複数個存在することがある．その場合はその中から使用者に選択させる．使用者は単に回路図を示されるだけでは選択しにくい．そこでその判定資料として基本回路の設計者名，設計日時，設計者からのコメントなども回路登録時に格納しておき，その場合にそれらを出力する．

さらに設計レベル１，２で下位レベルの設計が完了し，各ブロックの回路が定まった時，それらを接続しなければならない．しかしそのための一般的な手続きを記述するのは困難である．そこで下位レベルが存在する設計レベルでは，予想される接続関係を記述した「テンプレート」を格納しておき，それを基に下位レベルで定まった基本回路を接続する．

設計は仕様を満足しなければならないため，なんらかの方法で仕様を与えなければならない．設計仕様は設計前に仕様記述言語で記述し与える方法も考えられる．しかしこの方法は厳密な仕様を与えるが，仕様は設計途中で明確になることが多く，設計前にすべての仕様を列挙することは困難である．したがって各設計レベルに決めなければならない仕様を明記しておき，その仕様を必要となった時点で使用者に指定させる．

設計が必ず最下位レベルまで行くのは，高速に設計を行う上で好ましくない．そこで，各階層レベルごとに基本回路を設けておき，それが使用できない場合のみ下位レベルの設計を行う．

編集設計は仕様を満たす回路を選ぶだけでなく，それに対する修正操作も含んでいる．それを実現する方法として，修正手続きを記述する方法や，「REDESIGN」のようにシミュレーションを行って回路の動作と設計仕様との矛盾を発見し，エキスパートのもつ設計手続きで修正を行う方法も考えられる[1],[2]．しかし，いずれの場合でも修正手続きに一般性をもたせることは困難で，実用的な設計は不可能である．そこで本システムでは，基本回路の修正により他の回路設計に影響を及ぼさない場合だけ修正を許し，それ以外は修正を行わない．そのかわり多種類の基本回路を

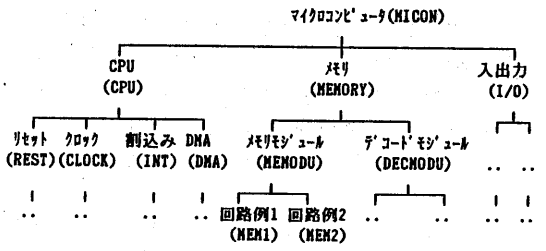


図１　マイクロコンピュータの階層構造

登録し，その中から仕様を満足する回路を取り出すことにした．

マイクロコンピュータは機能的には図１に示すように分類できる．つまり，マイクロコンピュータはCPU回路，メモリ回路，および入出力回路から構成され，さらにCPU回路はリセット回路，クロック発生回路，割り込み回路，DMA回路から構成される．そこであらかじめマイクロコンピュータ回路を図１のように分類し，基本回路として計算機内に格納しておく．そして上記の方法で与えられた仕様に基づきトップダウン設計とボトムアップ設計を行い，マイクロコンピュータ回路の編集設計を行う．

３．試作した設計システム

試作したシステムの構成を図２に示す．本システムは過去に作られた回路に関する情報を格納する「編集設計用データベース」と，設計仕様を格納する「仕様データベース」，設計結果を格納する「設計結果データベース」および設計支援を行う「設計エンジン」からなる．以下では各構成要素の機能について述べる．

3.1　編集設計用データベース

設計データを一元的に管理するのが好ましいため，編集設計用データはすべて「フレーム」[3]で表現した．フレームは人間のもつ知識を表現する１つの枠組みであり，スロット，ファセットとその値で知識を表現する方法である．フレームは(1)設計仕様を満足するように基本回路を修正する手続きなどの手続き的な知識と，回路の接続関係と回路仕様のような宣言的な知識の両方を格納することができる．(2)回路の階層関係のような知識を表現することができる．(3)共通した回路の情報は１箇所で管理できるという特徴をもつ．

設計用データベースには(1)基本回路に関する情報と(2)設計階層に関する情報の２種類の情報を格納しておく．

基本回路は，２．で述べたように，(1)回路

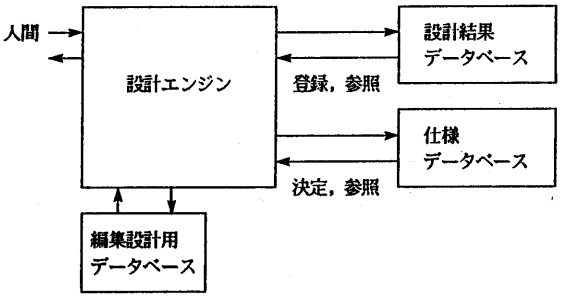


図2　システム構成

```
MEMODU  AKO         VALUE     MEMORY
        MODULE      DEFAULT   MEM1,MEM2
                    ..........
```

(a)ﾒﾓﾘﾓｼﾞｭｰﾙ(MEMODU)の回路表現

```
MEM1    AKO         VALUE     MEMODU
        CONNECTION  VALUE     (接続表現)
                    ELEMENT   (ROM1 2716),･･･
        DOC         VALUE     (ﾄﾞｷｭﾒﾝﾄ)
        SPEC        VALUE     (#ROMNAME 2716)
                              (#ROMNUM 3),･･･
                    IF-ADDED  #ROMTOP,#ROMBOTTOM,･･･
        CHECK       IF-NEEDED (
                      (LAMBDA (ROMTOP ROMBOTTOM ･･･)
                        (SETQ ROMTOP (FGET ･･･ ))
                              ..........))
```

(b)　回路ﾓｼﾞｭｰﾙ(MEM1)の回路表現

図3　メモリ素子モジュールの表現例

の接続関係だけでなく，(2)上位設計階層名，(3)回路仕様，(4)その回路を使用できるかどうかを調べるための手続き，および(5)その回路の設計者名，設計日時，設計目的，設計者からのコメントを格納する．

図３にメモリ回路のメモリ素子モジュールの表現例を示す．図３に示すように，AKOスロットには回路の上位階層を表すフレーム名MEMORYを格納する．そのレベルに基本回路MEM1,MEM2がある場合，MODULEファセットにその値を格納しておく．各回路モジュールは図３(b)に示すように１つのフレームで表現する．そこではAKOスロットに上位階層のフレーム名を格納しておく．

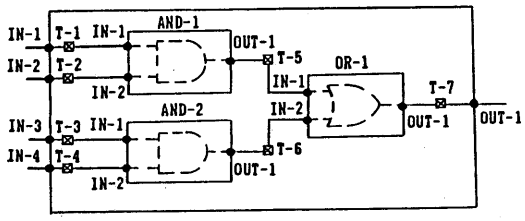
基本回路の接続関係は「モジュール」と「ポート」と「ターミナル」で表現し[4]，CONNECTIONスロットに格納する．回路の各構

成要素はブラックボックスとして考え，１つのモジュールで表す．各モジュールは２つ以上のポート（入力ポートと出力ポート）から構成され，回路の接続関係はターミナルとポート間の接続関係で表現する．図４に回路例とその表現結果を示す．図４(b)に示すように，(1)入力ポート，出力ポートと内部ターミナルとの接続関係，(2)各ターミナルと各内部モジュールとの接続関係，(3)各素子とターミナルとの接続関係を記述する．図４では存在しない抵抗やコンデンサ，ダイオードのような素子はその端子に１，２の番号をつけ区別する．トランジスタの場合は端子名の頭文字で区別する．使用するターミナル名，ポート名，素子名は回路登録時の労力を少なくするため，回路が異なっていれば重複を許している．さらにそれらには任意の名前をつけてもよい．設計完了時には各素子が具体的にどんな機能の素子かを調べる必要があるため，その情報をCONNECTIONスロットのELEMENTスロットに格納しておく．

基本回路を採用することで定まる仕様はSPECスロットのVALUEファセットに格納する．基本回路仕様により回路を使用できるかどうかを決定するが，さらに詳細なチェックを要する場合にはCHECKスロットにそれを調べる手続きを格納する．その手続きを実行するために定まっていなければならない仕様はSPECスロットのIF-ADDEDファセットに格納する．その仕様が定まっていなければ使用者に聞き，決定する．

仕様は回路規模が大きくなると増加するため，どの階層レベルの仕様か区別する必要がでてくる．そこで，その階層レベルを図３(a)に示すようにSPECFRAMEスロットに格納する．設計仕様を満足する基本回路が多数存在する場合に使用する設計者のドキュメントはDOCスロットに格納する．

設計階層は図５に示すようにそれぞれ上位階層はAKOスロットに，下位階層はHASスロットに記述しているフレーム名を格納する．またその階層での基本回路はMODULEスロットに格納しておく．さらに下位階層が２つ以上の



(a)回路例

```
ｽﾛｯﾄ    ﾌｧｾｯﾄ              値
CONNECTION VALUE ((INPORT  (IN-1 T-1) ･･･ (IN-4 T-4)
                  (OUTPORT (OUT-1 T-7))
                  (T-1     (AND-1 IN-1))
                  ･･･････････････････
                  (T-7     (OR-1 OUT-1) (OUTPORT OUT-1))
                  (AND-1 (T-1 IN-1)(T-2 IN-2)(T-5 OUT-1))
                  ･･･････････････････
                  (OR-1  (T-5 IN-1)(T-6 IN-2)(T-7 OUT-1)))
```

(b)回路の接続情報の表現

図４　フレームによる回路の接続関係の表現

ﾌﾚｰﾑ名	ｽﾛｯﾄ	ﾌｧｾｯﾄ	値
MEMORY	AKO	VALUE	MICON
	SPECFRAME	VALUE	MEMSPEC
	MODULE	DEFAULT	MEMORY1,MEMORY2
	HAS	*VALUE*	*MEMODU,DECMODU*
	CONNECTION	IF-ADDED	(接続用ﾃﾝﾌﾟﾚｰﾄ)
		･･･････	

図５　回路の階層関係

ﾌﾚｰﾑ名	ｽﾛｯﾄ	ﾌｧｾｯﾄ	値
MEMSPEC	ROMNAME	VALUE	使用ROM名 NIL
	RAMNAME	VALUE	使用RAM名 NIL
	ROMNUM	VALUE	ROMの個数 NIL
	ROMTOP	VALUE	ROMのスタート番地 &0000H
	ROMBOTTOM	VALUE	ROMのエンド番地 NIL
	･･･････	･･･････	･･･････

図６　仕様データベース

回路から構成される場合，それらを接続するための接続用テンプレートを格納する．テンプレートは下位階層の回路が決定された場合にそれらを接続するためのもので，前もって接続される可能性のある接続関係を図４の形式で記述したものである．

3.2　仕様データベース

仕様データベースは設計過程で定まる仕様を格納するためのものである．設計仕様はす

べて使用者が与える．この仕様データベースに格納されている仕様を満足する基本回路が設計エンジンにより選ばれる．

図６に仕様データベースの例を示す．このように初期状態では値はすべて格納されていなく，定まった時点で値を格納する．仕様は設計する回路により異なり，それを列挙すると相当量になる．したがって，基本回路を登録する時に仕様名が衝突しないよう，仕様名は基本回路の設計階層名と仕様名から構成している．

使用者に仕様の入力を要求する場合ROMTOPのように仕様を表す変数名を表示しても使用者はそれが何を表すかわからない．そこで，仕様値だけでなくそれが何を表すのかも格納する．

3.3 設計エンジン

設計は3.1で述べた形式で格納された基本回路を設計エンジンにより２．で述べた方法で行う．はじめにトップダウン設計を行い，下位階層の回路が決ってからボトムアップ設計を行う．

トップダウン設計時に設計エンジンはHASスロットを調べ，そこに格納されている機能モジュールを左から右に設計する．１つ設計レベルが下がるたびに設計エンジンはMODULEスロットに格納されている基本回路を，その記述順に設計仕様を満たしているかチェックする．その過程で仕様データベースを調べ，設計仕様が定まっていなければ使用者に仕様を要求し，仕様データベースに格納する．

基本回路が設計仕様を満たしている場合，各モジュールのDOCスロットに格納されている情報を表示し，使用するかどうか使用者に指定させる．どの基本回路も設計仕様を満たさない場合や使用しない場合は，下位レベルの設計に移る．

下位レベルの設計でも設計エンジンは上記と同様に設計を進める．下位レベルで回路が決まれば接続用テンプレートで接続する．その結果を設計結果データベースに格納し，上位レベルの設計にもどる．

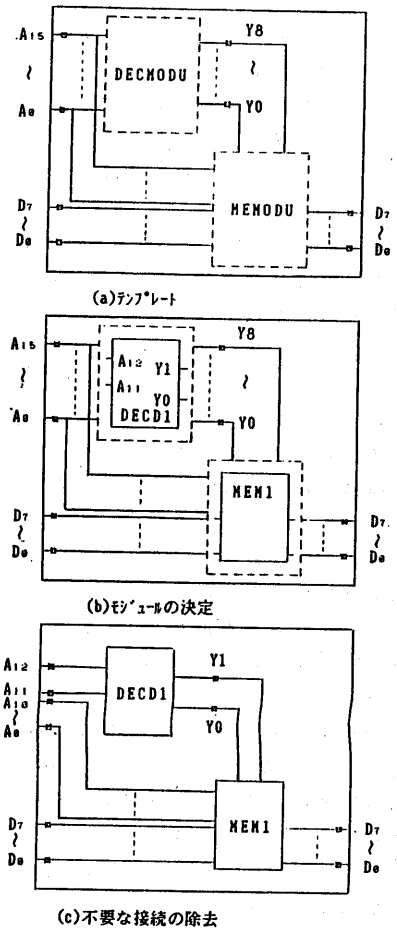


図７　メモリ回路の設計過程

上記過程を回路設計が完了するまで繰り返す．

図７にメモリ回路を設計する時の設計過程を示す．図７(a)ではデコードモジュール(DECMODU)とメモリ素子モジュール(MEMODU)から回路が構成されることを表している．そこで，HASスロットの内容を基にメモリ素子モジュールを設計し，次にデコードモジュールの設計に移る．メモリ素子モジュールでは基本回路MEM1を選択したとする．また，DECMODUではDEC1を選んだとする．その結果，図７(a)のレベルに戻り，図７(c)のようにテンプレート

を用いてそれらの接続を決定し，設計結果データベースに格納する．

3.4 設計結果データベース

設計結果データベースは設計エンジンによって作られた回路の接続関係を格納する部分である．回路は3.1で述べた形式で表現する．回路は各階層レベルごとに一時的に格納され，最終的には各回路が１つにまとめられる．その回路の仕様などは仕様データベースに格納されており，ここには接続関係だけを格納する．

3.5 本システムによる設計例

図８に本システムにより設計した回路例を示す．図８の回路はLEDを点滅するための回路である．図９に設計過程の使用者との対話を示す．図９のように計算機と対話しながら設計を進めて行く．

```
MEMODUを決定します

ROMTOP
この項目はすでに決っています
ROMのスタート番地は&0000Hです

ROMBOTTOM
ROMのエンド番地を決めて下さい
&FFFH

RAMTOP
RAMのスタート番地を決めて下さい
&FFFH

RAMBOTTOM
RAMのエンド番地を決めて下さい
&FFFH

MEM1モジュールを使用しますか (Y/N) Y
Y

DECMODEを決定します
```

図9 設計過程(一部)

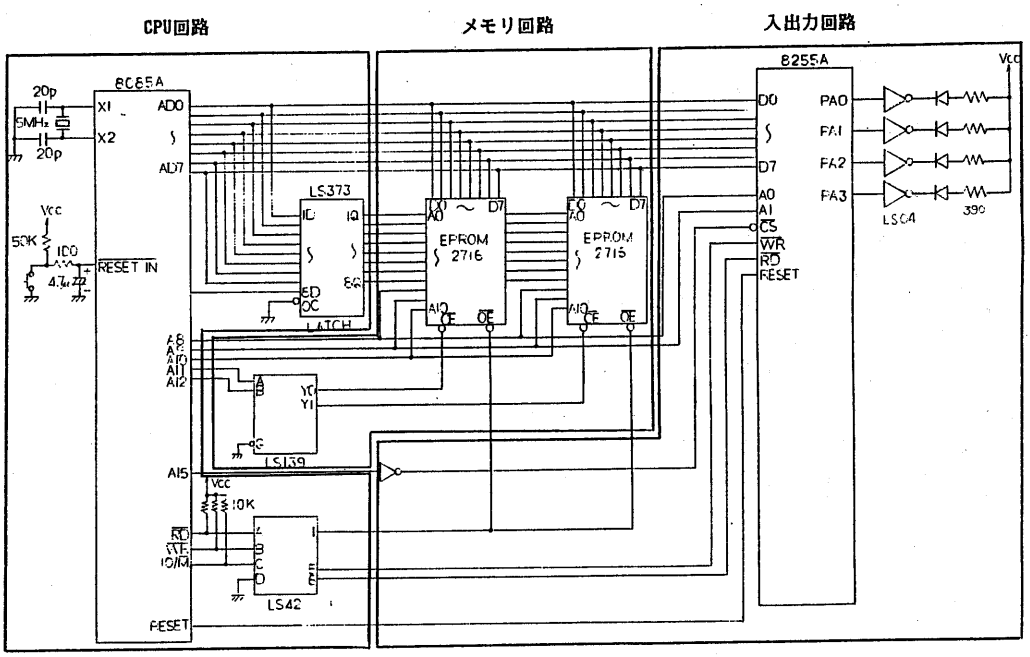


図8 設 計 例

４．おわりに

編集設計手法による設計可能なマイクロコンピュータの設計支援システムを試作した．その結果，本稿で述べた方法で回路を登録し設計を行えば設計可能であることを確認した．本システムは従来，人間が行ってきた設計手法を計算機の助力を得て行うもので，人間にとっては使いやすいシステムと考えられる．

現状はパソコン(NEC製PC-9800)上のLISPで本システムをインプリメントしており，マンマシンインターフェースも不完全である．今後は大型計算機への移植を予定しており，その際にマンマシンインターフェースを充実する予定である．また，現在登録されている基本回路は少なく，20個である．さらにこの数を増やし，本システムで柔軟な設計が可能か評価を進めて行く予定である．

編集設計手法はマイクロコンピュータだけでなく，一般的な論理回路設計にも利用できると期待できるが，有効かどうかは今後の課題である．

最後に本研究を進めるにあたり，熱心に研究討論に参加していただいた本研究室山本博資助教授に感謝します．

［参考文献］

(1)Steinberg,L.I. and Mitchell,T.M.:
"A Knowledge Based Approach to VLSI CAD the Redesign System", Proceedings of 21st Design Automation Conference, Paper 26.2, June 1984

(2)Kelly,V.E.:"The Critter System-Automated Critiquing of Digital Circuit Designs", Prc.fo 21st Design Automation Conference,papar 26.3, June 1984

(3)上野晴樹:"知識工学入門" ,オーム社, pp.84-108(1985)

(4)R.Davis,H.Shrobe :"Representing Structure and Behavior of Digital Hardware",COMPUTER,pp.75-82(1983)