人工知能と対話技法 26－2
（1982. 6. 11）

知的CADの構想

川戸信明　斉藤隆夫　広瀬貞樹　上原貴夫
（富士通研究所）

1. はじめに

大規模・複雑化するディジタル・システムの設計に対処するためには，階層設計が可能な論理設計支援システムが不可欠であり，更に論理設計の専門家レベルの知識を持つ高度な設計支援システムが望まれる。このためには，人工知能的手法の採用が必要であると考え，その第一歩として，フレームとデモンの概念に基づいたシステムを開発している[1]。フレームを利用することにより，ハードウェアの階層構造を容易に表現でき，さらにその機能に関する情報，設計上の制約条件を組込んで，デモンの機能によりそれらを活用することができた。このシステムは，階層的な論理設計を支援するグラフィック・エディタ，設計者の推論過程に適合する記号シミュレータを持ち，設計ミス発見のためのアサーションを与えることも容易である。しかし，設計の詳細化自体は設計者が行わなければならない。

一方，設計初期段階の検証を目的として，レジスタ・トランスファ・レベルの設計言語DDL（Digital system Design Language）に基づく設計支援システムを開発して効果を挙げている[2]～[4]。このシステムはDDLで与えられた機能仕様からゲート・レベルの設計を行うために必要な情報を抽出するトランスレータを持ち，詳細化過程の一部を自動化している。ただし，使用する半導体テクノロジに応じた論理分割や最適な論理回路合成は人手で行われている。ところが，このテクノロジに依存する設計は最も誤りを犯し易くかつ設計工数を最も必要とする部分であり，この設計を支援するシステムが強く望まれている。この部分の設計を自動化することを考えた場合，システムは論理設計の専門家の持つ設計手順，テクノロジや回路合成に関する種々の知識を持つことが必要である。したがって，これら各種の知識が容易に組みあわせ，さらにまたテクノロジ変化や異なるテクノロジに対しても知識の入れ換えによって対処できる柔軟な構造を持つシステムであることが望まれる。

本報告では，このような設計の詳細化過程を自動化する知的CADを実現するには，知識工学的手法の適用が必要であると考え，現在検討中のシステムについてその基本構想を述べる。

2. 構造設計と機能設計

ハードウェアの階層構造は，タイプとコンポーネントの概念を用いて表現することができる[5]。すなわち，サブシステムの集合からなるシステムはタイプと呼ばれ，サブシステムはコンポーネントと呼ばれる。サブシステムは下位レベルのタイプのインスタンスである（図1）。この点に関しては，本システムは文献[1]と同じである。しかし，このシステムでは，タイプの機能設計を行うことが許される。すなわち，設計は次のように行われる。図2に示したように，設計の開始時において，システムはいくつかのコンポーネントに分割され，ついで各タイプの詳細化が行われる。それぞれのタイプは構造的に更に

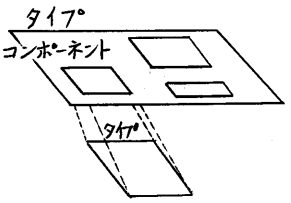

図1. タイプとコンポーネント

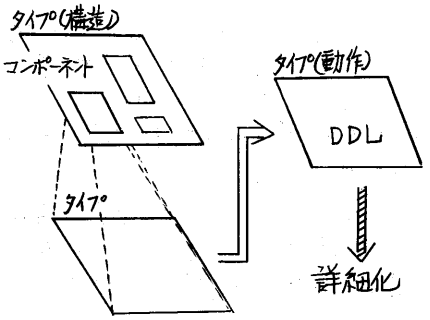

図2. 設計の詳細化

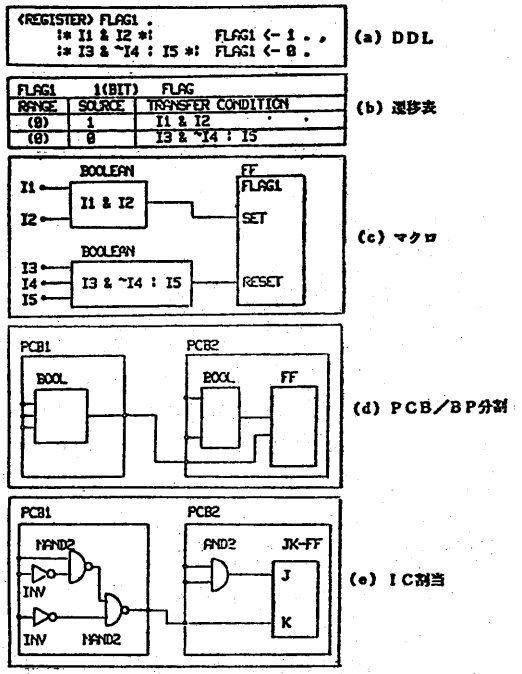

図3. 設計ステップ

詳細化される場合もあれば，またその機能仕様が与えられるともある。機能仕様はハードウェア設計言語DDLを用いて指定する。この場合は，その機能を実現するための構造的詳細化を行わなければならない。この詳細化は，全てのタイプの構造が決定し，展開された時の最下位レベルが特定のICに対応する迄続けられる（ここでは，TTL SSI/MSIテクノロジを用いる設計を考える）。この際，プリント板（PCB）やバックパネル（BP）へのシステムの分割も終了していなければならない。

この機能仕様から論理回路を合成する過程を支援することが本システムの目的である。このシステムの開発にあたっての基本方針は，従来論理設計の専門家が行っている設計手順を明らかにし，その手順をそのままシステムに組み込むことである。まず最初に，機能仕様から論理回路を得るまでの間に存在する設計者が意識する設計ステップについて調べ，図3のようなステップが存在することがわかった。すなわち，まず機能設計の結果から，各レジスタに対する転送条件や各ステートの遷移条件を収集し，これに基づき，テクノロジーとは独立のフリップ・フロップやマルチ・プレクサ等のマクロからなる構造的表現に変換する。ついで，マクロを実現するためのIC数やディレイの概算を行い，ピン数を考慮して，各マクロをプリント板に割り付け，システムの分割を行う。各マクロの特定ICを用いての合成は，この分割を考慮して行われる。以上の設計ステップのうち，機能設計から転送条件等を収集する部分は，既にDDLトランスレータが自動化しており，それ以降のステップ間の変換が現状では人手で行われている。したがって，更に自動化を可能とするためには，これらのステップ間の変換に使用される知識を明確にし，それをシステムに組み込まなければならない。論理設計の専門家はこの変換を数学的最適化というようなアプローチではなく，いろいろなヒューリスティクスを用いて，効率的に行っており，我々の目標はこのような知識をシステムに組み込むことである。

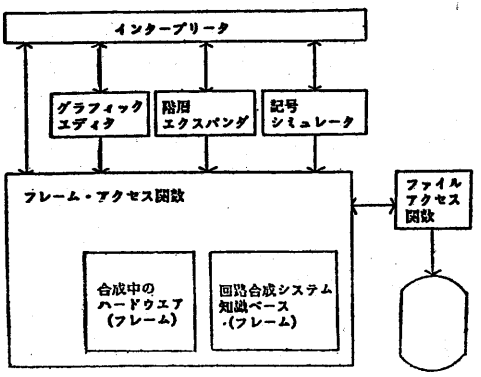

図4. システム構成

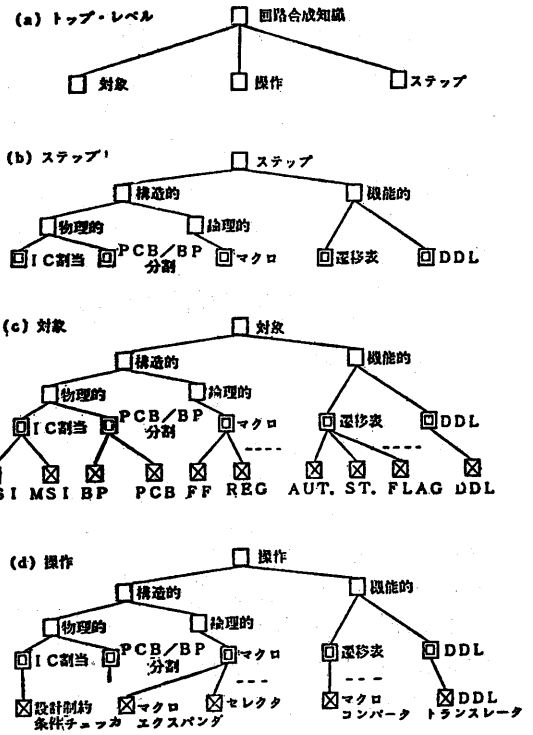

図5. 知識ベースの構造

3. システム構成

本システムの基本構成として図4に示した構成を考えている。インタープリータは，設計過程全般の管理を行うものと，変換過程に必要な設計タスクのプランニングを行う。システムの知識は合成中のハードウェアと論理回路合成用の知識ベースからなる。両者ともフレームを用いて表現する。グラフィック・エディタ，階層エクスパンダおよび記号シミュレータは既開発のものである[1]。

論理回路合成用の知識ベースは図5に示すように階層的に構成される。トップ・レベルの分類は，一般に，設計は何らかの「対象」を「操作」する「ステップ」からなることに対応している[6]。これ以下のレベルの分類は，前節の変換ステップを体系化することにより定義している。設計過程で動的に発生する設計タスクは「ステップ」の対応する部分に生成される。各設計ステップで取り扱う対象となるプロトタイプは「対象」の対応する部分に格納され，これらの「対象」の状態を変化させたり，各種の処理を行うオペレータは「操作」の対応する位置に置かれる。

設計は，インタラクティブ・グラフィック端末を用いて行う。グラフィックエディタの使用により，設計者は設計対象システムの構造あるいは機能仕様を与える。一般に単純な部分はマクロあるいは直接ICを用いて設計されるが，複雑な制御部は機能仕様がDDL心与えられることになる。インタープリータは設計中のハードウェアの各コンポーネントの状態から設計タスクを見出し，これを実行する。タスクを実行できるだけの知識が存在しなければ，設計者のインタラクションを要求することになる。このような設計のプランニングは，図6に示すアジェンダ制御構造を採用することによって実現できると考えられる[6]。すなわち，フォーカス戦略は，各オペレータに実行できるタスクを問い合わせ，

その中心最も優先順位の高いタスクを実行する。何らかの理由で実行できなくなったタスク（例えばfanoutチェックのタスクで出力先が全て合成済みでない時とかのようなことが起る）は延期される。レジューム戦略は，フォーカス戦略ではタスクが見つからなくなった時に起動され，延期されていたタスクのうち実行可能となったものを探し，これを再開する。オペレータは，タスクを見つける機能は必ず持たなければならないが，そのタスクを実行する機能は必ずしも持たなくてもよく，この場合は設計者に委ねることになる。タスクの実行を自動化できるだけの知識が明確になった時点でこれをオペレータに組み込めばよいと考える。

以下，例題を用いて本システムの動作を説明する。インプリメンテーションの詳細についてはまだ検討しなければならない点が残されているが，上に述べた知識の構造化や制御構造を用いれば実現は充分可能であると考えている。

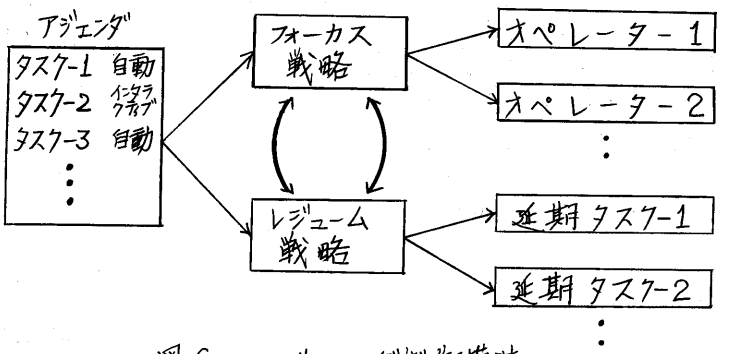

図6. アジェンダ制御構造

4. DDLトランスレーション

本システムへの入力はグラフィック・エディタ［1］により行う。設計者はDDLを用いてタイプの動作仕様を与えることができる。図7～9に例を示した。設計すべきシステム（CPU_INT）の構造を図7のように入力する。これは，3つのコンポーネントA_TO_S1, A_TO_S2, INITからなる。A_TO_S1, A_TO_S2は単純な回路でマクロレイ登録されているものを使用している。INITは複雑な制御回路であり，図8のように機能仕様をDDLで与える。図9はグラフィック・エディタが作り出したCPU_INTのフレームである。

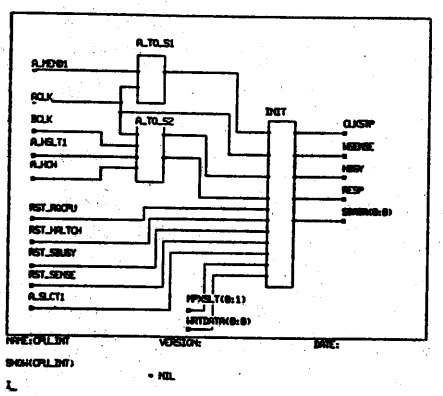

図7. グラフィック・エディタによる入力

```
(EXTERNAL) MEM1, HSLT1, HCH, ACLK, RST_RGCPU, RST_HALTCH,
           RST_SBUSY, RST_SENSE, A_SLCT1, MPSLT(2)
           MRTDATA(9), HBSY, RESP, SDATA(9), CLKSTP, MSENSE.
(TIME)     ACLK.
(REGISTER) MR01, RGCPU, HALTCH, SBUSY, RESP, SCLKSTP,
           MAITSENSE, SENSEBYTE4(9), SENSEBYTE5(9), .
           SENSEBYTE6(9), SENSEBYTE7(9).

(BOOLDRV)  HBSY=HSLT1 & "HCH & "RESP & SBUSY,
           CLKSTP=SCLKSTP , MSENSE=MAITSENSE .
(ASYNC)
           !e A_SLCT1 e!:      MR01   (<- 0 , .
           !e RST_RGCPU e!:    RGCPU  (<- 0 , .
           !e RST_HALTCH e!:   HALTCH (<- 0 , .
           !e RST_SENSE & "RGCPU e!:
                SENSEBYTE4 (<- B"1(9)" ,
                SENSEBYTE5 (<- B"1(9)" ,
                SENSEBYTE6 (<- B"1(9)" ,
                SENSEBYTE7 (<- B"1(9)" .

(AUTOMATON) START : ACLK :
   (STATES)
      IDLE : !e HSLT1 & HCH e!:
                HALTCH (<- 1 , RGCPU (<- 0 , SBUSY (<- 0 , -> RST . ,
             !e HSLT1 & "HCH & "SBUSY e!:
                MR01 (<- 1 , MAITSENSE (<- 1 , -> ERR_CHK . ,
             !e "(HSLT1 & HCH) & "(HSLT1 & "HCH & "SBUSY) e!:
                -> IDLE .
      RST  : "HALTCH : RESP (<- 1 , -> ATTENT .
      ERR_CHK : MEM01 :
                !e STATERR e!: SCLKSTP (<- 1
                ; RESP (<- 1 , RGCPU (<- 1 , SBUSY (<- 1 , MAITSENSE (<- 0 ,
                -> ATTENT . .
      ATTENT : "HSLT1 : RESP (<- 0 , -> IDLE .
(END) .
(*@) START .
```

図8. DDLによる機能仕様

インタープリータは，この時，設計タスクとして，A_TO_S1，A_TO_S2のマクロ展開とDDLトランスレーションを見出すが，DDLトランスレーションの方が優先度が高いので，図8のDDL記述を遷移表に変換する。図10は遷移表の内部フレーム表現である。これは，遷移表ステップの「対象」の集合とそれらの間の関係からなる。プロトタイプは既述のようにフレームとして格納されており，そのスロットはあらかじめ定義されていて，具体値が入れられてそのインスタンスが作り出される。たとえば，オートマトンのステートに対応したプロトタイプにはSET，RESETおよびINITIALというスロットがある。

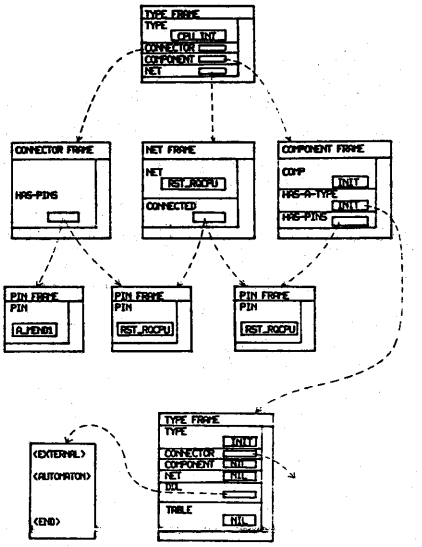

図9. ハードウェア構造のフレーム表現

SET(RESET)スロットは，このステートに入る（出る）条件を示す。INITIALはこのステートがオートマトンの初期状態か否かを示すために用いられる。この後，ブール式の簡単化やステート割当て等のタスクが発生し，これらが終了すると，遷移表はマクロに変換される。

5．論理分割

インタープリータは，CPU_INTの全てのコンポーネントがマクロまで詳細化されたことから，プリント板へのシステムの分割が可能であることを認識する。各マクロには，IC数やディレイの概算を行うオペレータを用意しておく。たとえば，TTLのSSI/MSIを用いる場合，ブール式を実現するためのIC数の概算は，全てNANDとINVERTゲートを用いて，fanout等の制約条件は無視して行うことが通常なので，このようなヒューリスティックな知識を組み込んでおく。インタープリータはこのような推定タスクを起動した後，分割のためのコンポーネントのトリー構造を作り出す（図11）。分割はコンポーネントをプリント板に割り付けることにより行える。中間レベルのコンポーネントを指定した場合は，これを親とする全てのマクロは同一プリント板に割り付けられる。この分割自体を自動的に行える見通しは現在のところなく，インタラクティブに行うことを考えている。したがって，システムはコンポーネントが割り付けられるたびに，プリント板のIC数やピン数を設計者に表示する程度の支援にならざるを得ない。

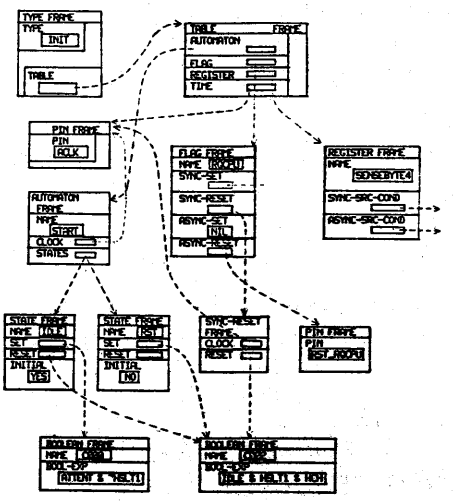

図10. 遷移表のフレーム表現

6. 回路合成

論理分割が終了すると，インタープリータは各マクロにICを割当てるためにマクロ・エキスパンダを起動する。本システムでは，マクロ・エキスパンダに，マクロが使用される環境を考慮して，最適なICを選んで展開するような機能を持たせることを考えている。

たとえば，マクロREGは図12に示したように2通りの展開が可能である。いづれを選択するかはREGの出力回路に依存する。すなわち，REGの出力がプリント板の出力ピンにのみ接続されているならQ出力を持つICを選び，複数の組合せ回路に接続されているならばQと$\overline{Q}$を持つICを選ぶというような知識が使用される。この種の知識は，常に正しいということをあらかじめ言うことは一般に困難であり，試行錯誤的にシステムに組み入れては確認するということが不可欠である。したがって，図13に示したようなルールの形で表現することが望ましい。このような表現を採用することにより，容易に知識の追加，修正，削除ができることになる。

マクロが展開されるとfanout等の制約条件をチェックするタスクが実行される。もし制約条件に違反すると，これを回避することが必要となる。したがって，違反のタイプ毎にこれを回避するタスクを実行できるオペレータを用意しなければならない。

7. おわりに

以上，機能仕様に基づき論理回路の自動合成を行うシステムについて考察し，知識工学的手法の適用が充分に可能であることを示した。現在，本稿で述べた基本構想にもとづき，より具体的なインプリメンテーション上の問題について検討を行っている。なお，ここでは，TTL SSI/MSIを対象としたが，同様の手法がVLSIに対しても適用可能であると考えられる。

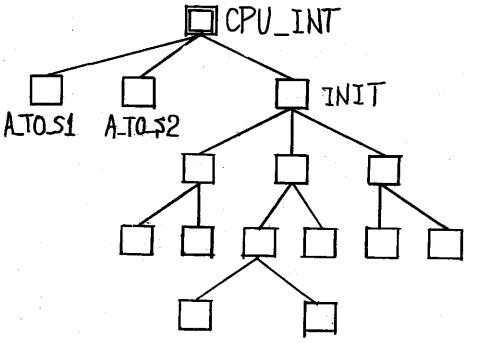

図11. プリント板分割のための
コンポーネント・トリーの発生

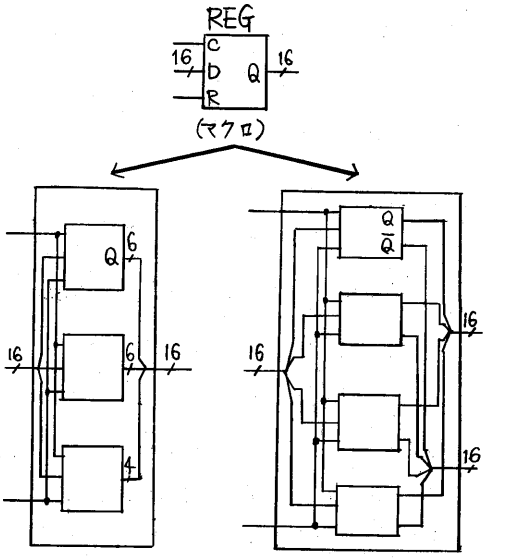

図12. マクロ・エキスパンダ

図13. マクロ・エキスパンダの知識

```
RULE-0:
  IF
    CONNECTED ONLY TO OUTPUT PIN
  THEN
    USE REGISTER WITH ONLY Q
RULE-1:
  IF
    CONNECTED TO COMBINATIONAL CIRCUITS
  THEN
    USE REGISTER WITH Q AND Q̄
```

参考文献

[1] Takao Saito, Takao Uehara, and Nobuaki Kawato, "A CAD System for Logic Design Based on Frames and Demons", Proc. of 18th DA. Conf., pp.451-456, July, 1981.

[2] Nobuaki Kawato, Takao Saito, Fumihiro Maruyama, and Takao Uehara, "Design and Verification of Large-Scale Computers by Using DDL", Proc. of 16th DA. Conf., pp. 360-366, June 1979.

[3] Fumihiro Maruyama, Takao Uehara, Nobuaki Kawato, and Takao Saito, "Hardware Verification and Design Error Diagnosis", Proc. of FTC-10, pp. 59-64, October 1980.

[4] Takao Uehara, Fumihiro Maruyama, Takao Saito, and Nobuaki Kawato, "DDL Verifier", Proc. of 5th International Conference on Computer hardware description languages, September 1981.

[5] W. M. van Cleemput, "An Hierarchical Language for the Structured Description of Digital Systems", Proc. of 14th DA. Conf., pp.377-385, June 1977.

[6] Mark Jeffrey Stefik, "Planning with Constraints", Stanford University, Computer Science Department, Report No. STAN-CS-80-784, 1980