フレームとデモンによるCADシステム（FIDDLE）

斉藤隆夫　川戸信明　上原貴夫
（富士通研究所）

1. はじめに

論理設計者の能力と経験を十分に生かすことのできるCADシステムを実現するためには，コンピュータを人工知能化することが必要であると考え，MINSKY［1］によって提案されたフレームを基礎としてシステムを構築することとした。フレームの利点は，ハードウェアの階層構造を容易に表現できる他に，その機能に関する情報，設計上の制約条件，さらには設計者の持っているノウハウ等を組込んで，デモンの機能によりそれらを活用できることである。

本文では，論理設計のための人工知能を実現するための第一歩として開発したCADシステムFIDDLE（FRAMES FOR IDEAL DESIGN WITH DEMONS OF LOGICAL EXAMINATION）について，データベースとしてのフレームの実現，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および設計ミス発見のためのアサーションの与え方等について述べる。

2. フレームとデモン

フレームは

（フレーム　（スロット　（ファセット　（バリュー））））

という枠組より成り，フレームを作ったり，その内容を検索する標準関数はLISPで簡単に用意できる［1］。例えば，

FGET（フレーム　スロット　ファセット）

は，バリューをとりだす関数であり，

FPUT（フレーム　スロット　ファセット　バリュー）

は，バリューをしまう関数である。

フレームの大きな特徴の一つは，データの一種としてデモンと呼ばれる関数を格納しておき，変更，追加あるいは削除されたデータの値に応じてそれを起動することができる点である。

FGET+（フレーム　スロット　ファセット）

は，バリューをとりだす関数であるが，バリューが存在しない場合にはIF_NEEDEDデモンが起動する。

FCHANGE+（フレーム　スロット　ファセット　バリュー）

は，バリューをおきかえる関数であるが，バリューに変化がある場合にはIF-CHANGEDデモンが起動する．
次の例では，SIA（SERIAL INTERFACE ADAPTER）のコンポーネントは，INTF（INTERFACE）とTRNS（TRANSMITTER）より成ること，INTFのICの数は40個であること等を表現しようとしている．

```
((SIA  (COMPONENT  (VALUE (INTF)(TRNS)))
       (HAS_ICS    (IF_NEEDED (TOTALIZE))))
 (INTF (HAS_ICS    (VALUE (40))))
 (TRNS (HAS_ICS    (VALUE (130)))))
```

TRNSのICの数を知りたければ，

```
FGET+(TRNS HAS_ICS VALUE)
```

と入力すると，

```
(130)
```

と答が返ってくる．SIAのICの数が知りたい場合にも

```
FGET+(SIA HAS_ICS VALUE)
```

と入力するが，この場合にはVALUEが存在しないので，IF_NEEDEDデモンがTOTALIZEという関数を起動して，SIAのコンポーネントであるINTFとTRNSのICの数を合計して

```
(170)
```

という答を返すようになっている．
FCHANGE+は，後に述べるグラフィックエディタにおける図面修正および記号シミュレータにおけるイベント伝達に関して重要な役割をはたす．

3．タイプとコンポーネント

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[2]．図1は，SIAの構造をこの概念にしたがって表現した例である．図1.1～1.3は，各タイプがどのようなコンポーネントから成るかを木で示し，また各コンポーネントのタイプを（）内に示している．図1.4～1.6は，コンポーネントの具体的な接続関係を示している．例えば，図1.2から，タイプTRNSがSFTR1，SFTR2等のコンポーネントより成り，SRTR1，SRTR2は共にタイプSFT4，すなわち図1.6に示された4ビットのシフトレジスタによる直並列変換回路，であることがわかる．

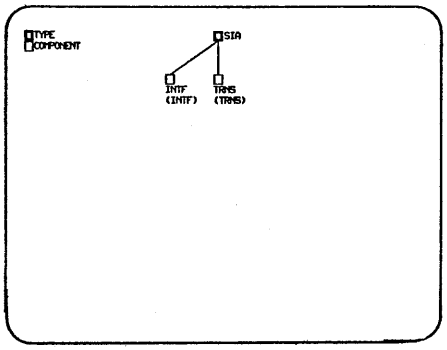


図1.1 SERIAL INTERFACE ADAPTERの
ハイアラーキ ダイヤグラム

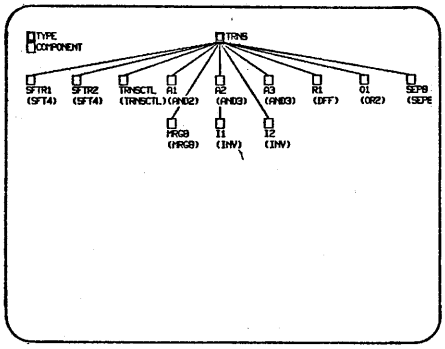


図1.2 TRANSMITTERの
ハイアラーキ ダイヤグラム

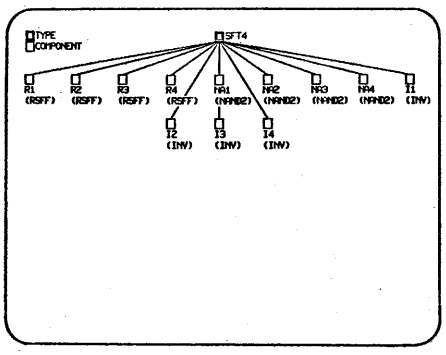


図1.3 4 BIT SHIFT REGISTERの
ハイアラーキ ダイヤグラム

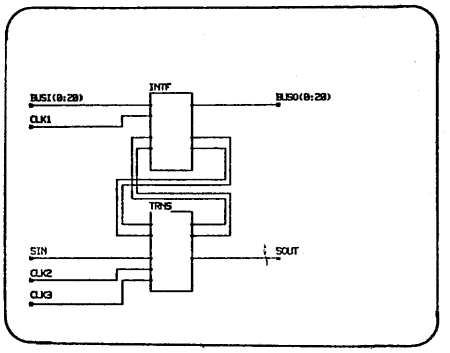


図1.4 SERIAL INTERFACE ADAPTERの
ブロック図

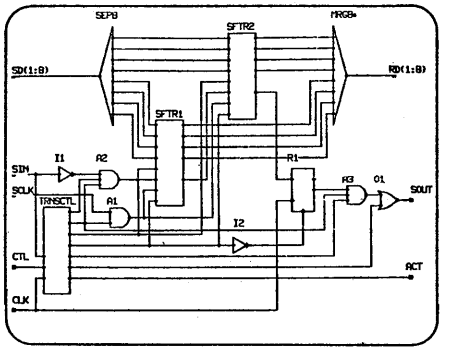


図1.5 TRANSMITTERの
ブロック図

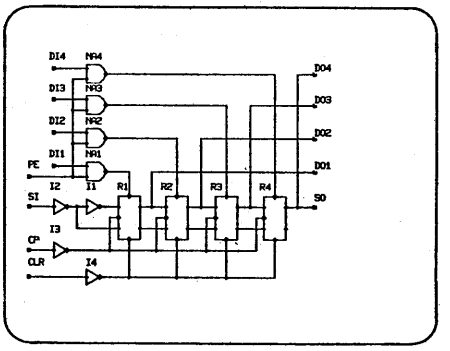


図1.6 4 BIT SHIFT REGISTERの
回路図

図1. グラフィックエディタにより階層的に表現された
SERIAL INTERFACE ADAPTERの設計

4. システムの概要

図2に，FIDDLEのシステム構成を示す．

フレームの作成，参照，修正および削除を行う基本関数，フレームをファイルに出入れする関数をMシリーズのコンピュータ上にLISPで実現した．グラフィックエディタは，設計者がフレームを意識せずに設計が進められるような図面によるインタフェースを提供する．記号シミュレータは，図面上で論理式のままのシミュレーションを進めるので，設計者は分析が行いやすい．マクロエキスパンダは，階層的データを必要なレベルまで展開するもので，シミュレーションのためのフレームを生成するのに使われている．

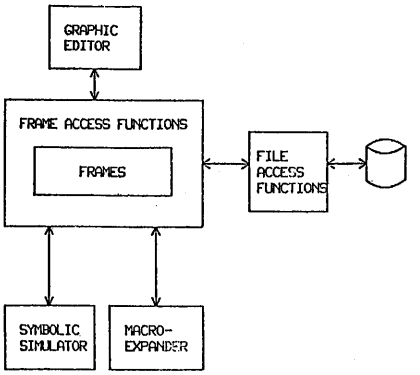


図2. FIDDLEのシステム構成

5. グラフィックエディタ

タイプとコンポーネントの概念に従ったハードウェアの構造表現は，図3のようなフレームで実現できる．グラフィックエディタによるタイプの定義はつぎのように行う．例えば，タイプSFT4のコンポーネントとしてNAND2というタイプをもつた2入力NANDゲートNA1, NA2, NA3およびNA4を加えるには，

ADD_C(SFT4 NAND2((NA1 NA2 NA3 NA4)))

と入力すると，カーソルにより各コンポーネントの左下位置を指定するようシステムからの要求が起る．外部端子やネットについても同様に，コマンドと位置指定を行って行けば自動的に図3のようなフレームができあがる．

デモンは，図の編集を行うための関数を実現することを容易にしている．例えば，使用者がコマンドとカーソルによりコンポーネントの移動を指示すると，それに接続するネットに関しても

FCHANGE+(ネット名 LOCATION VALUE 新座標)

という関数でデモンが起動され，次々と修正が加えられる．

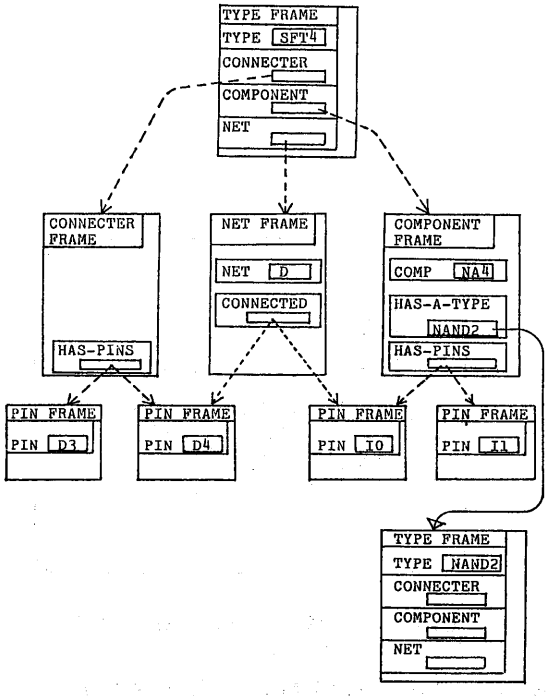


図3. フレーム表現されたタイプとコンポーネント

6. 記号シミュレータ

記号シミュレータは，図面上で論理式のままのシミュレーションを進めるもので，通常はデータを記号のままにして制御信号を0または1の論理値として用いると便利である。

本シミュレータは，フレームのネットワークを記述する機能とデモンのイベントを検知する機能およびLISPの記号処理機能を活用して作られている。その概要は次のようになっている。

（1）SIM-PROLOG(回路名) により，階層的に定義された回路を展開し，信号値を格納するためのフレームを作りだし，ピンの入出力タイプに応じて IF_CHANGED デモンを格納する。

（2）EXECUTE() によりシミュレーションが開始される。シミュレータは，INIT(条件 ピン名 値) により初期設定されたイベントやシミュレーション中に生じたイベントの内から，現時刻に起すべきイベントをとりだし，FCHANGE+(ピン名 SYMVAL VALUE 信号値) により，信号値をフレームに格納する。これによりフレーム内の信号値が変化するとIF_CHANGEDデモンが起動して新しいイベントが起る。

例えば，図4において，ピンD2に記号値Aが与えられると，出力のタイプをもつデモンOUTPが起動される。(もし，ピンD2の値がもともとAならばOUTPは起動されない。) このOUTPは，ピンD2につながるコンポーネントE3のピンP2に対して FCHANGE+(P2 SYMVAL VALUE A) を実行する。これにより値の変化があれば，P2に付加されている入力のタイプのIF_CHANGED デモンであるINPが起動される。INPは，図5のようなコンポーネントの機能定義を評価して出力値を求め，もし可能ならば簡単化を行い，イベントとして登録する。図5では，時刻T2で記号値 A@BをE3の出力ピンにしまうイベントが登録された。シミュレータは時刻T1のイベントがなくなれば時刻を更新し，上述の処理をくりかえす。

（3）CLOCK(ピン名 周期 幅 位相) によりクロックが定義されており，現時刻でその値が変化するならば，（2）と同様にして対応するピンに値を伝える。

（4）現時刻に関するイベントがなく，STOP(条件) によって与えられた停止条件が満たされていれば，シミュレーションを停止する。

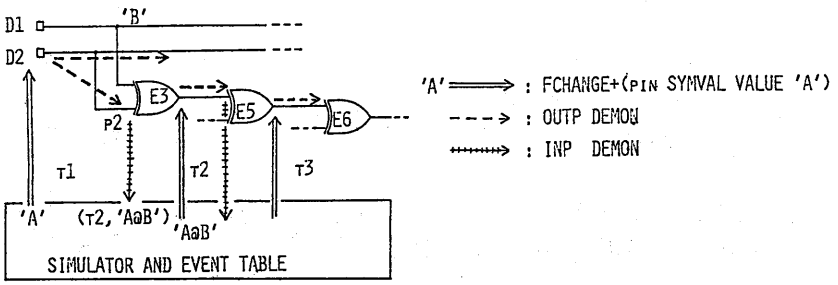
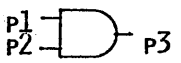


図4. 記号シミュレータの原理

P1	P2
0	1
1	0
A	~A

```
DEFINE(((NOTS (LAMBDA (ARG)
      (COND ((EQ ARG 0) 1)
            ((EQ ARG 1) 0)
            ((EQ ARG 'X) 'X)
            ((EQ (CAR ARG) '~) (CADR ARG))
            (T (LIST '~ ARG)) ) ) )))
```



P1 \ P2	0	1	A
0	0	0	0
1	0	1	A
B	0	B	A&B

```
DEFLIST(((ANDS (LAMBDA (ARGS A)
       (PROG (V VARIABLES)
   LOOP  (COND ((NULL ARGS) (GO PROLOG)))
         (SETQ V (EVAL (CAR ARGS) A))
         (COND ((EQ V 0) (RETURN 0))
               ((EQ V 1))
               ((MEMBER V VARIABLES))
               ((CMPLMNT_CHK V VARIABLES) (RETURN 0))
               (T (SETQ VARIABLES (NCONC VARIABLES (LIST V)))))
         (SETQ ARGS (CDR ARGS))
         (GO LOOP)
   PROLOG(COND ((EQUAL (LENGTH VARIABLES) 0) (RETURN 1))
               ((EQUAL (LENGTH VARIABLES) 1) (RETURN (CAR VARIABLES))
               (T (RETURN (CONS '& VARIABLES))))
      ) ) )) FEXPR)
```

P1 \ P2	0	1	A
0	0	1	A
1	1	0	~A
B	B	~B	A@B

```
DEFLIST(((EOR (LAMBDA (ARGS A)
   (PROG (V FLG VAL)
    (SETQ FLG 0)
    LOOP
    (COND ((NULL ARGS) (GO PROLOG)))
    (SETQ V (EVAL (CAR ARGS) A))
    (COND ((EQ V 0))
          ((EQ V 1)(SETQ FLG (ADD1 FLG)))
          ((MEMBER V VAL) (SETQ VAL (DEL-M V VAL)))
          ((CMPLMNT_CHK V VAL) (PROG (SETQ VAL (DEL-M2 V VAL))
                                     (SETQ FLG (ADD1 FLG))))
          (T (SETQ VAL (NCONC VAL (LIST V)))))
    (SETQ ARGS (CDR ARGS))
    (GO LOOP)
    PROLOG
    (SETQ FLG (REMAINDER FLG 2))
    (COND ((EQUAL (LENGTH VAL) 0) (RETURN FLG))
          ((EQUAL (LENGTH VAL) 1)
           (COND ((ZEROP FLG) (RETURN (CAR VAL)))
                 (T (RETURN (NOTS (CAR VAL))))) )
          (T (COND((ZEROP FLG) (RETURN (CONS '@ VAL)))
                   (T (RETURN (NOTS (CONS '@ VAL))))) ))
   )) )) FEXPR)
```

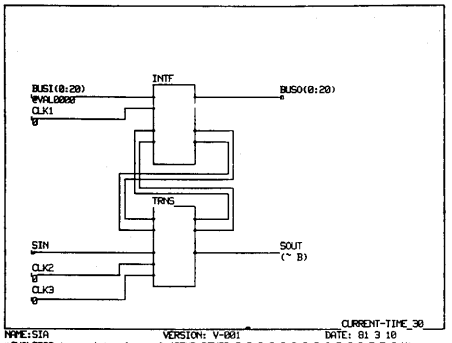
図5. 記号シミュレーションのための機能定義

本シミュレータでは，上述のデモンINP，OUTPの他に，*IF_CHANGED*デモンの一つとして*SHOW_VALUE*がある。これは図6に示すように，ディスプレイ画面上のブロック図（回路図）の対応する位置に値（記号値／論理式）を表示するものである。

設計者は，時々刻々変化するシミュレーションの状態を図面上で見ながら，論理記号を用いた抽象的な分析を進めることができる。さらに，階層のどのレベルを見るかは

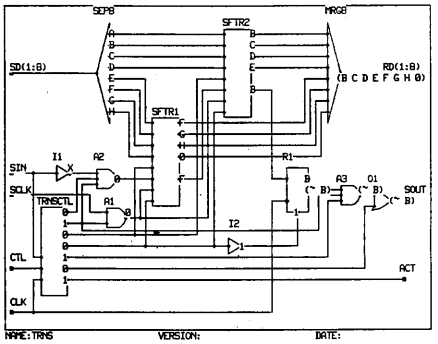
PRC_SHOW(DOWN コンポーネント名*)* あるいは *PRC_SHOW(UP NIL)*

等のコマンドを使って自由に選べる。



BUSI(0:20)にSEND命令とDEVICE IDENTIFIER及び転送すべきデータが入力され，コンポーネントINTFはデータの下位バイト（A B C D E F G H）と出力命令をコンポーネントTRNSに伝える。SOUTからは，A，B，C，D，…と順次反転出力されるはずである。現時刻では，第2番目のビットBが反転出力されている。

さて，TRNSの内容を見たければ，
PROC_SHOW(DOWN (TRNS))
とコマンドを与えれば，左の図が現れる。



左図で，A，B，C，D，… という記号値が 1，0，0，1，… というような論理値で与えられる通常の論理シミュレーションを思い起すと，記号シミュレーションが人間に与える情報がいかに多いか理解できる。

さらに，SFTR1の内容を見たければ，
PROC_SHOW(DOWN (SFTR1))
とコマンドを与える。

上のレベルにもどりたいときには，
PROC_SHOW(UP NIL)
とコマンドを与える。

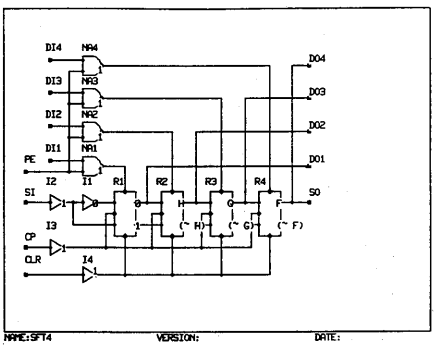


図6．記号シミュレーションの例

7. アサーション

最近，BACKWARD REASONINGにより，アサーションと設計の間の矛盾を見つけだす検証法が開発された[3]ここでは，FORWARD REASONINGすなわちシミュレーションの過程でアサーションに反する状況が発生したことを検知する方法について述べる。

設計者は，仕様の一部とか設計条件等をアサーションとしてフレームに付加する．このアサーションに対応するデモンは，シミュレーションの際中ずっとデータを監視しており，矛盾が起ると使用者に警告をだす．

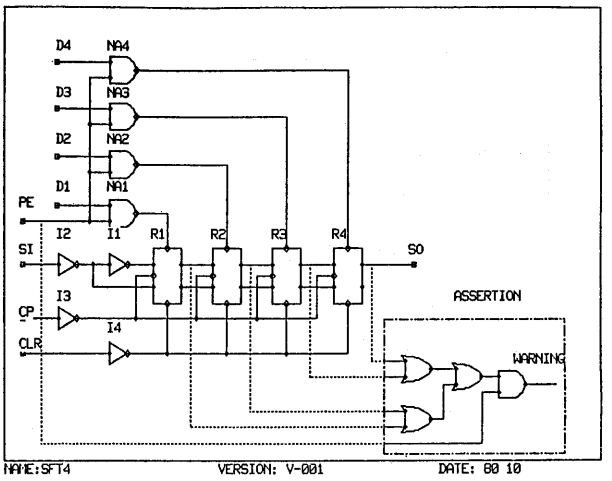


図7. アサーションの例

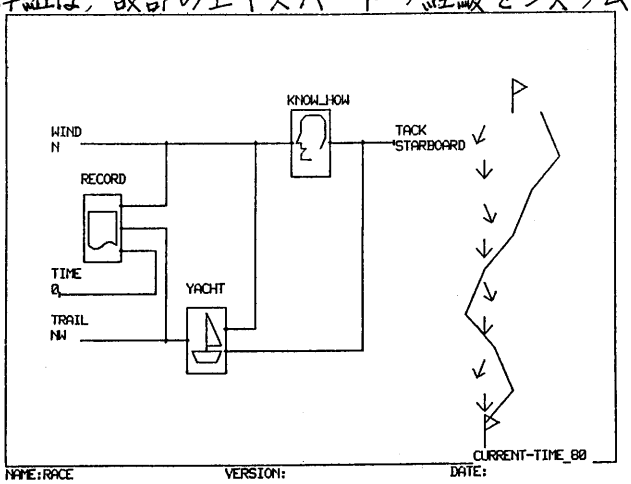
例えば，図7のSFT4のシフトレジスタにデータを並列にロードする場合，各フリップフロップの値が0でなければ正常に値をセットすることができない．設計者はこの知識をデモンとしてSFT4に与えることにより，シミュレーション時に，SFT4のタイプをもつコンポーネントSFTR1，SFTR2の状態を監視し，エラーメッセージの出力やシミュレーションの停止を行うことができる．

8. むすび

このCADシステムは，富士通の大型コンピュータ上にLISPで開発された．階層的な図面の上に表示される記号シミュレーションの結果は，理想的なマンマシンインタフェースを実現している．使用者が定義した関数を容易に付加できる枠組は，設計のエキスパートの経験をシステムに組込むことを可能とした．

なお，このシミュレータは論理回路にかぎらず一般的な因果関係をシミュレートすることができる．図8では，ヨットレースにおけるノウハウの効果をシミュレーションによって確めている．

最後に，日頃御指導いただく富川情報処理研究部長に感謝します。




```
* The condition (EQUAL %CRNTTIME 80) satisfied
  = NIL
I _
```

図8. ヨットレースのシミュレーション

参考文献

[1] WINSTON, P.H. "ARTIFICIAL INTELLIGENCE", ADDISON-WESLEY, 1977.

[2] VAN CLEEMPUT, W.M. "AN HIERARCHICAL LANGUAGE FOR STRUCTURED DISCRIPTION OF DIGITAL SYSTEMS", 14TH DAC, pp. 377-385, JUNE, 1977.

[3] MARUYAMA, F. et al. "HARDWARE VERIFICATION AND DESIGN ERROR DIAGNOSIS", FTC-10, pp. 59-66, OCT. 1980.