設計自動化 29－2
（1985. 11. 19）

機能シミュレータ MELFUNの実現方式

Simulation Techniques of
Functional Simulator MELFUN

榎本 清之　人見 高史　石川 淳士　松下 浩明　村井 真一
Kiyoshi Enomoto　Takashi Hitomi　Junji Ishikawa　Hiroaki Matsushita　Shinichi Murai
（三菱電機（株）情報電子研究所）
Information Systems and Electronics Laboratory
MITSUBISHI ELECTRIC CORP.

内容梗概

ここで紹介するのは，設計者にとって記述容易な状態遷移記述による任意の機能モジュールとゲートの混在を許し，かつそれらが階層的構造記述で表現された回路を模擬できる機能シミュレータ，MELFUNである．模擬回路は機能モジュールの動作と機能モジュール間の接続関係を表現するハードウェア記述言語，MDLで表現される．模擬アルゴリズムは，コンパイル方式とイベント・ドリブン方式の混合である．バッチ型と会話型の実行ができ，4値の信号値まで扱える．プログラムは移植性を考慮してFORTRAN77で書かれた．MELFUNは，これまでにVLSI設計およびプリント基板設計に使われている．

1 はじめに

一般に論理装置の設計は，方式，機能それに論理の各設計段階からなる．誤りの無い回路を設計するためには，各設計段階での設計検証を充分に行うことが重要である．

論理回路の設計検証を支援するツールとして論理シミュレータは有力な手段である．とくに，ゲートレベル・シミュレータは論理設計段階で広く使用されている．

しかしながら，このシミュレータを使用する場合，以下に記す問題点がある．

（1）汎用マイクロ・プロセッサ等のLSIを含む基板回路の模擬が難しい．これは，そのようなLSIのゲート回路図を知ることが一般に不可能なためである．

（2）VLSIの機能設計段階で使用できない．換言すると，設計がより詳細なレベルすなわち論理設計段階まで進まないと模擬できない．

（3）電子計算機のような大規模回路の模擬が難しい．これは，以下の二点に起因する．

・直接には関心のない回路，たとえば入出力装置でもゲートで表現することが要求される．

・かりに全体をゲートで表現できたとしても，全体の信号線数は数十万本以上となり，禁止的な模擬時間を要する．

このような問題点を解決する目的で機能レベル・シミュレータが開発されている〔1〕〔2〕．これらのシミュレータは，レジスタ・トランスファ・レベルで記述された回路を模擬するシミュレータである．さらにはレジスタ・トランスファ・レベルとゲートレベルの混在を許す混合レベル・シミュレータ〔3〕〔4〕も報告されている．

ここで紹介するのは，設計者にとって記述容易な状態遷移記述〔2〕による機能モジュールとゲートが混在し，かつそれらが階層的構造記述で表現された回路を模擬できる〔4〕シミュレータMELFUNである．本報告では，模擬回路を表現するためのハードウェア記述言語MDL，MELFUNの模擬方式，および適用例について述べる．

2 MELFUN 概略

2.1 特徴

構造化設計支援：大規模回路の設計では構造化設計が必要不可欠である．MELFUNでは構造記述言語MDL－Sでこれを支援している．

機能設計支援：機能設計では回路を構成するモジュールの動作を自由に表現できることが望ましい．MELFUNでは機能記述言語MDL－Fでこれを支援している．

模擬方式：一般に論理シミュレーション方式には，コンパイル方式〔5〕とイベント・ドリブン方式（テーブル・ドリブン方式）〔6〕がある．MELFUNの模擬方式は両者を混合した方式である．

ユーザ・インタフェース：MELFUNはバッチ型と会話型の模擬を支援している．とくに会話型の模擬では時間を溯る機能を支援することにより，マイクロプログラムのデバッグ等を容易にしている．

信号値：MELFUNで扱う信号値は，0，1，X（不定），Z（ハイ・インピーダンス）の4値である．

移植性：MDLのコンパイラ，シミュレータ本体等の開発では，すべてFORTRAN77を使用し，様々な計算機への移植を容易にした．

2.2 システム構成

図2－1にMELFUNのシステム構成を示す．

ラン・タイム・ルーチンは，以下の二つのプログラムを含む．

（a）メイン・プログラム

模擬の実行制御を行うプログラムで，その主な機能は，

・模擬制御命令の解釈／実行
・外部信号の入力
・機能モジュールの起動
・信号値の保存
・時間の管理

（b）基本関数プログラム

MDL－Fで許される算術，論理演算等を実行するサブルーチンを含む．この他には，各々ゲートタイプの動作を記述したサブルーチンを含む．

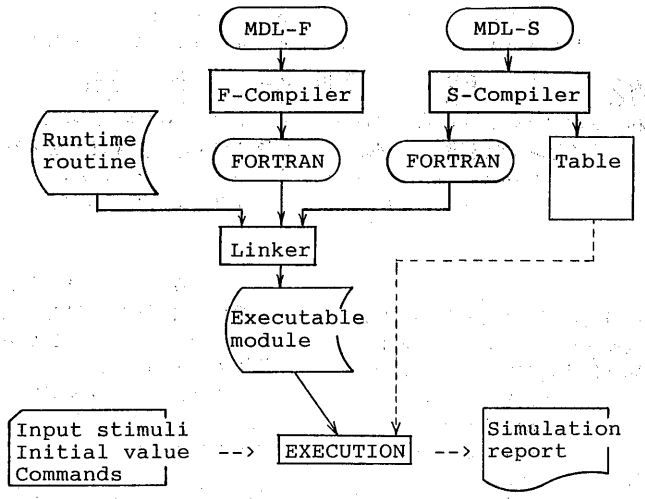


図２－１　ＭＥＬＦＵＮのシステム構成

Ｆコンパイラは，ＭＤＬ－Ｆによって記述された機能モジュールの動作をＦＯＲＴＲＡＮサブルーチンに変換するプログラムである。一般に，１個の機能モジュールから複数個のサブルーチンが生成される。

Ｓコンパイラは，ＭＤＬ－Ｓによって記述された回路の接続関係をテーブル形式に変換するプログラムである。また，メイン・プログラムと「Ｆコンパイラによって生成されたサブルーチン」を接合するサブルーチン（接合サブルーチン）を生成することもＳコンパイラの機能である。

模擬の実行前に，ラン・タイム・ルーチンとＦＯＲＴＲＡＮコードに変換された機能モジュールおよび接合サブルーチンをリンクすることで実行モジュールができあがる。ここで使用するリンケージ・エディタは計算機システムが提供するユーティリティである。

実行時に入力波形，メモリ等の初期値および制御命令が入力される。また，会話型模擬では，実行途中で割込がかかり，信号値等の変更や繰り返し実行ができる。

ソース・プログラム規模は以下のとうりである。

・メイン・プログラム　　６　Ｋステップ
・基本関数プログラム　　２　Ｋステップ
・Ｆコンパイラ　　１７　Ｋステップ
・Ｓコンパイラ　　１４　Ｋステップ

３　ハードウェア記述言語ＭＤＬ

ハードウェア記述言語はハードウェア仕様の明確化，論理シミュレータ入力用言語等に用いられ，数多く提案されている(7)．ＭＤＬは構造化設計を支援する言語で，ＭＤＬ－ＦとＭＤＬ－Ｓから成る．設計者は機能的にまとまった部分（機能モジュール）の動作をＭＤＬ－Ｆで記述し，それらの間の接続関係および階層構造をＭＤＬ－Ｓで記述することによってＭＥＬＦＵＮを実行できる．

３．１　ＭＤＬ－Ｆ

ＭＤＬ－ＦはＤＤＬ(8)をベースとした言語である．そのため状態遷移表現を用いてモジュールの動作を記述できる．

ＭＤＬ－Ｆの特徴は，

・１個のオートマトンで１個の機能モジュールを表現する．
・モジュール内の個々のデータ・フローを記述する部分とそれのコントロールを記述する部分を明確に区別する．
・多相クロックによる制御の記述が可能である．
・レジスタ転送として同期転送と非同期転送を用意し，これにより並列動作と逐次動作を区別する．

図３－１にＭＤＬ－Ｆの記述例を示す．この例は，参考文献(2)から引用した簡単な計算機の動作を表現している．以下では，おのおのＤＩＶＩＳＩＯＮごとに記す．

IDENTIFICATION DIV.

ここでは管理情報を記述する．記述項目は機能モジュール名，日付，管理者名，プロジェクト名，版名の5項目である．

INTERFACE DIV.（界面部）

機能モジュールの外部端子名を記述する．端子名に付随して端子のレンジ幅を記述することもできる．クロック端子は入力端子と区別して独立に定義される．

FACILITY DIV.（設備部）

レジスタ，メモリ，ターミナルといったファシリティを宣言する．ここでもレンジの指定ができる．レジスタにはクロックに同期して転送が完了するものと，クロックと無関係に転送が完了するものがある．図3－1のディレイド・レジスタは前者に属する．

OPERATION DIV.（操作部）

界面部および設備部で記述した名前を用いて個々のデータ・フローを記述する．各々データ・フローには名前（操作名）が付与される．図3－1においてADSが操作名である．ADSの動作内容は，a）CARをインクリメントする，b）CARをMARへ転送することである．これらの動作は同時に実行されることを表現しているので記述の順序を変えても意味は変わらない．また，模擬結果も記述の順序に左右されない．このような場合に用いられる転送が同期転送である．転送記号は ’<－’ である． 一方，非同期転送は，’：＝’で表現される．

CONTROL DIV.（制御部）

操作部で定義した操作名を用いてモジュールの動作を記述する．モジュール内の同期，非同期部分を各々SYNCHRONOUS，ASYNCHRONOUS SECTIONに記述する．図3－1において，STATE ADSSは一つの状態を示す．回路がこの状態にあって，かつクロックCLKが＜1＞の時に実行される動作がPHASEの内容である．つまり，操作ADSが起動され，実行完了後，状態はIFTTへ遷移することを意味している．

3.2 MDL－S

MDL－SはSDL[9]をベースとした言語である．モジュールの階層およびモジュール間の接続関係は，MDL－Sを用いて記述できる．特に，接続関係の記述では信号単位とモジュール単位の記述を許している．

図3－2にMDL－Sの記述例を示す．以下では，MDL－Fとの対応で各DIVISIONの内容を述べる．

IDENTIFICATION DIV.

基本的にはMDL－Fと同じであるが，JOINが加わる．JOINを用いて，接続関係の記述形式を指定する．ここでは信号単位の指定をしている．この他にモジュール単位の指定ができる．

INTERFACE DIV.（界面部）

クロック端子を特別視しない点を除いてMDL－F

```
IDENTIFICATION DIVISION                    ;
   NAME     CPU                            ;
   DATE     1985,10,18                     ;
   AUTHOR   K.ENOMOTO                      ;
   PROJECT MELFUN PROJECT                  ;
   VERSION A00                             ;
END DIVISION                               ;
INTERFACE DIVISION                         ;
   EXTERNAL    CLK                         ;
   CLOCK       CLK                         ;
END DIVISION                               ;
FACILITY DIVISION                          ;
   DELAYED REGISTER   CAR(0:9),
                      MAR(0:9),
                      ACC(0:15),
                      IR(0:15)             ;
   STATE REGISTER     ST(0:2)              ;
   MEMORY             M(0:1023,0:15)       ;
END DIVISION                               ;
OPERATION DIVISION                         ;
   ADS:   CAR(0:9) <-  CAR(0:9)(+)1        ;
          MAR(0:9) <-  CAR(0:9)            ;END;
   IFT:   IR(0:15) <-  M( MAR,0:15)         ;END;
   DEC:   MAR(0:9) <-  IR(6:15)             ;END;
   LDA:   ACC(0:15)<-  M( MAR,0:15)         ;END;
   STA:   M(MAR,0:15):=ACC(0:15)            ;END;
   ADD:   ACC(0:15)<-
          ACC(0:15)(+) M( MAR,0:15)         ;END;
   BRA:   CAR(0:9) <-  IR(6:15)             ;END;
   BRP:  IF ACC(0:0)<>1B1 THEN
          CAR(0:9) <-  IR(6:15)             ;
         ENDIF                             ;END;
END DIVISION                               ;
```

```
CONTROL DIVISION                           ;
   SYNCHRONOUS CONTROL SECTION             ;
      STATE  ADSS                          ;
         PHASE  CLK POSITIVE               ;
            ACT  ADS; GO IFTT              ;END;
      STATE  IFTT                          ;
         PHASE  CLK POSITIVE               ;
            ACT  IFT; GO DECC              ;END;
      STATE  DECC                          ;
         PHASE  CLK POSITIVE               ;
            ACT  DEC; GO IRR               ;END;
      STATE  IRR                           ;
         PHASE  CLK POSITIVE               ;
            IF   IR(0:5) = 4    THEN
                 ACT LDA                   ;ENDIF;
            IF   IR(0:5) = 8    THEN
                 ACT STA                   ;ENDIF;
            IF   IR(0:5) = 16   THEN
                 ACT ADD                   ;ENDIF;
            IF   IR(0:5) = 32   THEN
                 ACT BRA                   ;ENDIF;
            IF   IR(0:5) = 33   THEN
                 ACT BRP                   ;ENDIF;
            GO  ADSS                                ;END;
      END SECTION                          ;
   END DIVISION                            ;
END MODULE                                 ;
```

図3－1 MDL－Fの記述例

と同じである．

FACILITY DIV.（設備部）

回路を構成するモジュール（機能モジュールとは限らない）のタイプ名とそれを参照するオカレンス名を記述する．

CONNECTION DIV.（接続部）

界面部および設備部で定義した内容を用いて接続関係を記述する．モジュールの端子名は，下位の構造記述あるいは機能記述で定義されることを仮定している．下位モジュールの内容が，さらに構造記述であれば，それはモジュールの階層を表現していることになる．

```
IDENTIFICATION DIVISION                    ;
      NAME     DBY12C                      ;
      JOIN     SIGNAL                      ;
      DATE     1983, 8 , 31                ;
      AUTHOR   T_HITOMI                    ;
      PROJECT  MELFUN_PROJECT              ;
      VERSION  A00                         ;
END DIVISION                               ;
INTERFACE DIVISION                         ;
 EXTERNAL     RD1 , RD2 , QA  , QB ,       ;
              QC  , QD  , T1C , T2C        ;
 INPUT        RD1 , RD2 , T1C , T2C        ;
 OUTPUT       QA  , QB  , QC  , QD         ;
END DIVISION                               ;
FACILITY  DIVISION                         ;
  TYPE       NAND2 , JKFF05                ;
  NAND2      GN                            ;
  JKFF05     G1 ,  G2  , G3 , G4           ;
END DIVISION                               ;
CONNECTION DIVISION                        ;
  SIGT1 = < ¥SYS.T1C >< G1.TC >            ;
  SIGT2 = < ¥SYS.T2C >< G2.TC ,
                        G3.TC >            ;
  SIGRD = < GN . P1Y >< G1.RC ,
                        G2.RC ,
                        G3.RC ,
                        G4.RC >            ;
  SIGRD1= < ¥SYS.RD1 >< GN.P1A>            ;
  SIGRD2= < ¥SYS.RD2 >< GN.P1B             ;
  SIGQA = < G1.Q >< ¥SYS.QA >              ;
  SIGQB = < G2.Q >< ¥SYS.QB ,
                    G3.J    >              ;
  SIGQC = < G3.Q >< ¥SYS.QC ,
                    G4.TC   >              ;
  SIGQD = < G4.Q >< ¥SYS.QD >              ;
  SIGFB = < G3.QC>< G2.J    >              ;
  ¥ISAKI=T0<G1.J ,G1.K ,G2.K ,
            G3.K ,G4.J ,G4.K >             ;
  ¥¥AKI$= <G1.QC,G2.QC,G4.QC >             ;
END DIVISION                               ;
END MODULE                                 ;
```

図３－２　ＭＤＬ－Ｓの記述例

４　模擬手法

ＭＥＬＦＵＮではコンパイル方式とイベント・ドリブン方式を混合した方式で模擬を実行している．前者は機能モジュールの模擬に対して，後者はモジュール間の模擬に対して適用されている．

４．１　モジュール間の模擬

模擬アルゴリズム

機能モジュール間の模擬では，多くの論理シミュレータで採用されているイベント・ドリブン・アルゴリズムを用いている．すなわち，機能モジュール間の接続関係表を参照しながらイベントの発生にしたがってサブルーチンになった機能モジュールを呼び出している．

模擬アルゴリズムを図４－１に示す．「外部端子からの信号値入力」から「時刻の更新」までのループを繰り返し実行する．繰り返しの数は，入力波形の変化時刻の数に等しい．ただし，制御命令によって途中で止めることもできる．

機能モジュールの評価では次事象同期式模擬アルゴリズム[10]を用いている．このアルゴリズムは，いわゆるタイム・マッピング・アルゴリズムの特別な場合で，単位遅延を仮定した，それである．このアルゴリズムでは，全信号の状態値が安定するまで，イベントの発生にしたがって各機能モジュールが評価される．

データ構造

模擬時に用いるデータ構造を図４－２にしめす．モジュール定義表は模擬対象となる機能モジュールのオカレンスの数だけ用意される．動作を実行するサブルーチンへは，この定義表へのポインタが渡される．ファシリティ表はファシリティの値を格納する．信号表は信号値，ファンアウト数，ファンアウト表へのポインタを格納する．ファンアウト表はモジュール定義表へのポインタを格納する．ファンイン表は信号表へのポインタを格納する．

各々表の幅は基本的に４バイトで，信号値などの値を格納するところのみ３２バイトである．すなわち，一つの値を１バイトの空間に格納しているので，３２ビット幅の値を一括して処理している．

接合サブルーチン

メイン・プログラムから機能モジュールを呼び出すことができるように，Ｓコンパイラは一個のサブルーチンを生成している．このサブルーチンは，モジュール定義表からタイプ番号を検索し，呼び出すサブルーチンを決定するプログラムである．したがって，メイン・プログラムは常にこの接合サブルーチン名のみを呼び出している．

このサブルーチンは機能モジュールを呼び出すとともに，ＭＥＬＦＵＮシステムが持つゲートタイプを呼び出すこともしている．まず機能モジュールを呼び出すことを試み，機能モジュールのタイプ番号でない時，ゲートタイプを呼び出すことを行う．

接合サブルーチンは，図２－１のリンクのところで挿入される．なお，機能モジュールとタイプ番号の対応はＦコンパイラが決定し，管理している．

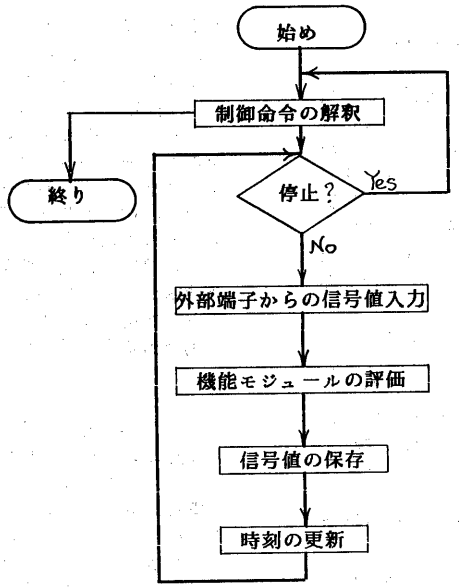
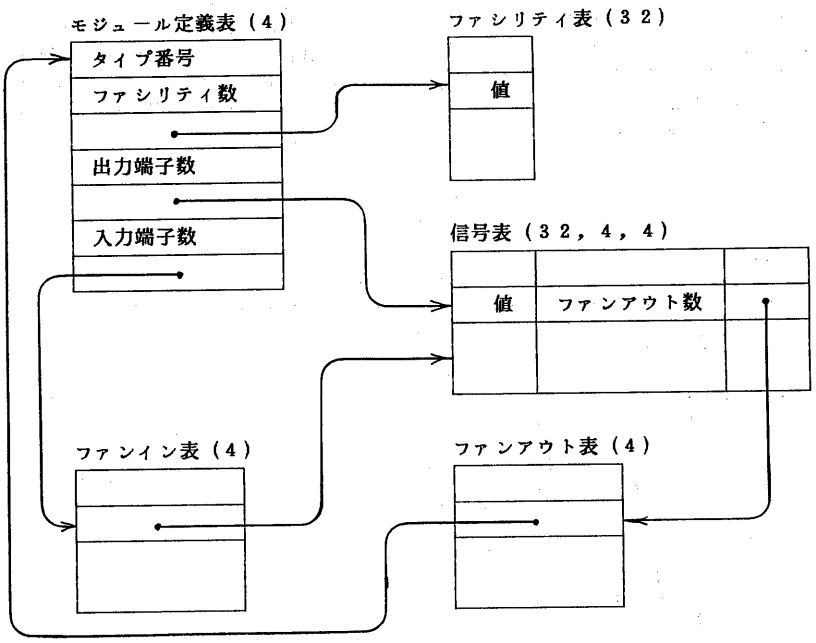


図４－１　模擬アルゴリズム



カッコ内の値は各々表の幅をバイト数で表現している。

図４－２　ＭＥＬＦＵＮのデータ構造

4.2 機能モジュールの模擬

生成されるサブルーチン

Fコンパイラは，MDL－Fによって記述された機能モジュールをFORTRANのサブルーチンに変換する．その際，文法チェック等も同時に行われる．出来上がるサブルーチンは，機能モジュールに対応するもの（親）一個と各々操作に対応するもの（子）複数個である．すなわち，操作名が四個定義してあれば全部で五個のサブルーチンが生成される．図3－1の記述から生成されたFORTRANサブルーチンの一部を図4－3に示す．

図4－2のデータ構造は，共通変数としてサブルーチンに渡るのでFコンパイラは，このデータ構造を仮定してFORTRANコードを生成している．図4－3における配列MEMは，モジュール定義表，信号表の一部，ファンイン表およびファンアウト表を含んでいる．変数STCDT等は，MEMにおける各々表の先頭アドレスを格納している．また，MEMFAC，MEMSIGは，ファシリティと信号の各々値を格納している．

親となるサブルーチンは，モジュール定義表へのポインタを引数として，接合サブルーチンから起動される．さらに，親は，制御部の内容に従って子を起動する．図4－3において，引数MODNOがモジュール定義表へのポインタである．（A）の部分は状態を調べるプログラムであり，該当する文番号へ遷移する．文番号10001，10002等は図3－1の状態名ADSS，IFTTに対応する．文番号10001の下でF0RT01を呼び出しているが，このサブルーチンは，操作名ADSに対応する．サブルーチン内で呼び出されるCMSKOT等は次に述べる基本関数プログラムである．

基本関数プログラム

MDL－Fで許される総ての演算については基本関数プログラムで処理される．演算子一個に対して一つの基本関数プログラムが対応する．図4－3の文番号10001の次の文においてICHR2Iという関数がある．基本関数プログラムのほとんどは，この様な関数サブルーチンの形をしている．したがって，実行結果を格納した関数が別の関数の引数となるような形で使われる場合もある．

信号値等は，文字変数を使って表現される．このため基本関数は，その引数に文字属性を定義することにより，容易に実現できた．図4－4はAND論理演算の場合の基本関数プログラムである．

4.3 会話型実行

模擬の実行形態として，バッチ型および会話型の両方を支援している．ここでは会話型模擬について記す．

模擬アルゴリズムは，基本的に図4－1と同じである．バッチ型との相違は，機能モジュールの評価の前後で割込をかけ，ユーザへの問い合わせを行っている点である．ユーザが使える命令は，

a）信号値の表示および変更
b）ファシリティ値の表示および変更
c）模擬中断点の設定
d）信号値等の保存および再格納
e）現在時刻の表示

等である．このうちd）を用いると，同一周期にて「機能モジュールの評価」を繰り返し実行できるので，a），b）を併用することによってマイクロ・プログラムのデバッグ等が容易に行える．

```
**************
      FUNCTION  CAND  ( ARG1, ARG2 )
*************************************************************
      CHARACTER*32      CAND, ARG1, ARG2
      CHARACTER*1       LOW, HIGH, UNKNWN
      DATA LOW,HIGH,UNKNWN /'0','1','X'/
*************************************************************
      DO  10   K=1, 32
            IF(ARG1(K:K).EQ.LOW.OR.ARG2(K:K).EQ.LOW) THEN
               CAND(K:K)=LOW
        ELSEIF(ARG1(K:K).EQ.HIGH.AND.ARG2(K:K).EQ.HIGH)  THEN
               CAND(K:K)=HIGH
        ELSE
               CAND(K:K)=UNKNWN
        ENDIF
   10 CONTINUE
      RETURN
*
      END
```

図4－4　AND論理演算の基本関数プログラム

```
*
      SUBROUTINE    FORT00  ( MODNO )
*
      IMPLICIT  INTEGER ( A,B,D-Z )
      IMPLICIT  CHARACTER*32 (C)
      INTEGER           CDTHED
      CHARACTER*32      MEMFAC,MEMSIG
      COMMON  /XCTABS/ MAXFAC,LENFAC,MAXSIG,LENSIG
      COMMON  /XCTABL/ STCDT,STFCL,STSGN,STFRE,EDFIFO,LENMEM,MEM(1)
      COMMON  /XPNTTB/ CDTHED,FCLHED,FIHEAD,DRGHED,DRSIZE,OUTHED
      COMMON  /XCTFAC/ MEMFAC(1)
      COMMON  /XCTSIG/ MEMSIG(1)

*
      GOTO(10001,10002,10003,10004) ICHR2I(MEMFAC(FCLHED+00000001-STFCL
     .+1))                                                              (A)
          WRITE( 006,30001 )
     .                MODNO,ICHR2I(MEMFAC(FCLHED+00000001-STFCL+1))
30001     FORMAT( 1H ,'*** STATE REGISTER ERROR ***'
     .          / 1H ,'MODULE NO = ',I8
     .          / 1H ,'STATE = ',I8 )
          STOP 'CPU - '
*
10001 CONTINUE
          IF(ICHR2I(MEMSIG((MEM(FIHEAD-00000000)-STSGN)/3+1)).EQ.1) THEN
            CALL FORT01
            MEMFAC(FCLHED+00000001-STFCL+1)=CI2CHR(00000002)
          ENDIF
      GOTO 20000
*
10002 CONTINUE
          IF(ICHR2I(MEMSIG((MEM(FIHEAD-00000000)-STSGN)/3+1)).EQ.1) THEN
            CALL FORT02
            MEMFAC(FCLHED+00000001-STFCL+1)=CI2CHR(00000003)
          ENDIF
      GOTO 20000
```

・・・（省略）

```
****************** SUBOPERATION DIVISION ******************
*                                                           *
*     OPERATION NAME :  ADS                                 *
*                                                           *
*************************************************************
*
      SUBROUTINE   FORT01
      IMPLICIT  INTEGER ( A,B,D-Z )
      IMPLICIT  CHARACTER*32 (C)
      INTEGER           CDTHED
      CHARACTER*32      MEMFAC,MEMSIG
      COMMON  /XCTABS/ MAXFAC,LENFAC,MAXSIG,LENSIG
      COMMON  /XCTABL/ STCDT,STFCL,STSGN,STFRE,EDFIFO,LENMEM,MEM(1)
      COMMON  /XPNTTB/ CDTHED,FCLHED,FIHEAD,DRGHED,DRSIZE,OUTHED
      COMMON  /XCTFAC/ MEMFAC(1)
      COMMON  /XCTSIG/ MEMSIG(1)
*
      CALL  CMSKOT( MEMFAC(DRGHED+00000002),0000,0009
     .,'00000000000000000000000000000000'
     .,'10010000000000000000000000000000',CPLUS(CMSKIN(MEMFAC(FCLHED
     .+00000003-STFCL+1),0000,0009,'00000000000000000000000000000000'
     .,'10010000000000000000000000000000')
```

・・・（省略）

図４－３　Ｆコンパイラによって生成されたサブルーチン

5 適用例

ＭＥＬＦＵＮは，これまでにマイクロプロセッサを含む基板の論理シミュレーション，およびＶＬＳＩの機能レベル・シミュレーションに使用された実績をもっている． ここでは，後者の例を記す．

・模擬回路の規模
　２０Ｋゲート相当の回路で，ＣＭＯＳゲートアレイ２個とＲＡＭから成る．
・ＭＤＬ記述量
　ＭＤＬ－Ｆ・・・・・３０００　行
　機能モジュール数・・　　４０　個
・模擬時間
　３．６秒／周期　　（ＶＡＸ１１／７８０）

6 おわりに

機能シミュレータＭＥＬＦＵＮの実現方式，それのハードウェア記述言語ＭＤＬおよび適用例について述べた．現在，ＭＥＬＦＵＮはＶＬＳＩ設計データベースに接続され使用されている．今後の課題を以下に記す．

模擬速度の向上

Ｆコンパイラは最適化機能を持っていないので，生成するサブルーチンに冗長なコードがある．これの最適化により模擬速度の向上が期待できる．

模擬精度の向上

ＭＥＬＦＵＮは機能設計レベルでの使用に重点を置いたため単位遅延を仮定した．しかしながら，将来，混合レベル・シミュレータとして利用していく場合，遅延の導入は避けられない．模擬速度を低下させることなく遅延の導入を計ることが課題である．

機能記述の図的表現

ハードウェア設計者にとっては一次元的な言語記述より二次元的な図的表現のほうが親しみやすい．そこでＭＤＬ－Ｆを図的表現する試みをおこなっている．

参考文献

1) Chappell,S.G.et al.:Function Simulation in the Lamp System, 13th DA conf., pp.42-47 (1976)

2) 上原貴夫他: 大型コンピュータなどの論理設計に適用できるＣＡＤシステム，日経エレクトロニクス，１９７９．１１．１２，ｐｐ．１０４－１３０（１９７９）

3) 市村徹他: ミックスシミュレータＦＡＬシステム，情報処理全国大会（第３０回），ｐｐ．１９４５－１９４６，（１９８５）

4) Sasaki,T.et al.:MIXS:A Mixed Level Simulator for Large Digital System Logic Verification, 17th DA Conf., pp.626-633 (1980)

5) Nash,D. et al.: Functional Level Simulation at Raytheon, 17th DA Conf., pp.634-641 (1980)

6) 桶浦尚登他: テーブルドリブン方式論理機能シミュレータ，情報処理，設計自動化研究会資料，２，１５，（１９８３）

7) 中村行宏他: ハードウェア記述言語とその応用，情報処理，Ｖｏｌ．２５，Ｎｏ．１０，ｐｐ．１０３３－１０４０，（１９８４）

8) Duley,J.R. and Dietmeyer,D.L.:A Digital System Design Language, IEEE Trans. Com-put., Vol.C-17, No.9, pp.850-861, (1968)

9) VanCleemput,W.M. : A Hierarchical Language for the Structural Description of Digital Systems, 14th DA conf., pp.377-385 (1977)

10) Ｍ．Ａ．ブルーア編，池田敏雄校閲: ディジタル計算機の自動設計，産業図書，ｐｐ．１３８－１４１，（１９７３，１０）