B2　電子回路の動作シミュレーション

佐藤昭治，勝枝嶺雄，氏家一彬　（日立中央研究所）

1．まえがき

集積回路技術が高度に発達し，集積度の著しく高い，いわゆる大規模集積回路（Large Scale Integrated Circuit－LSI）が，最近，各種用途に，使われるようになって来ている．LSIは部品でありながら，一種のシステム的様相を強く持ったもので，したがって多品種少量生産となる傾向がある．このようなことから電子回路動作のシミュレーション・プログラムは，LSI－CADの一つの道具として，ますます重要視されるようになった．

電子回路動作シミュレーションのプログラムは，これまでも数多く開発され使用されているが，回路記述言語，特にトランジスタ・モデルなどの取扱い，あるいは温度依存や，製造過程で決まる定数をパラメータで表現する機能，ユーザ定義の非線型素子の特性式の入力などは，一部のプログラムに部分的に入っているだけで十分ではない．また，回路動作のシミュレーションには，直流，交流，過渡の三状態があるが，それらに統一的に使えるトランジスタなどのモデルの記述法がないなどの問題がある．

筆者らは，これらの点を考慮し，計算機言語になじみの薄い回路設計者にも使い易く，かつ回路設計への高度な応用も可能であることを目指して，汎用回路動作のシミュレーションプログラム── HICAD（HITACHI Computer Aided Design）を開発して来た.[1] HICADは一部既に実際に使用されており，また現在も拡張改良が行なわれている．

回路動作のシミュレーション処理（回路解析）では，大型の行列演算および微分方程式を解く問題を含んでいるが，最近は大型の行列演算しかも回路解析で見られるゼロ要素の多い，いわゆるSparse行列の演算処理に多くの関心が向けられている．

ここでは，HICADの回路記述言語とSparse 行列処理に多くの紙数を割いて述べる．

2．回路記述言語およびオペレーショナル・コマンド

2．1　回路記述言語上の二，三の問題点

一つの問題向き言語である回路記述言語の，多くが持っている，これまでの問題点のいくつかを上げると，つぎのようになろう．

1）　サブ回路の取扱い　　回路設計者にとって，トランジスタなどの等価回路モデル，更に一般的にサブ回路を，マクロ的に扱えるようにすることは，回路記述を格段に容易にする．すなわち，トランジスタは，抵抗，従属電源，キャパシタンスなどの基本素子によってモデル化されるが，一般に一つの回路に複数個使用され，それらをいちいち全部記述することは大変である．それで同じトランジスタは，一回記述して，名前をつけて登録しておき，その名前で

必要な所に呼び込めるようにすることが必要である．しかし，トランジスタ・モデルは作りつけにしてしまうと融通性がなくなるという問題がある．すなわち回路設計者が，任意にモデルを作る必要が生ずること，さらにトランジスタを含む大きな回路がサブ回路として記述したい，すなわちサブ回路の多段使用という要求もおこる．

2） パラメータ・モデイファイ　　IC，LSIでは，製造条件や温度依存などで，（回路連結状態はそのままで）素子定数や回路変数（素子の電流，電圧）の初期値が変って来る．また，素子値定数間に相関関係がある場合があり，したがって相関的記述と相関的モデイファイも必要となってくる．さらに，サブ回路の多段使用における素子値の変更といった問題も生ずる．

3） 直流ー交流，直流ー過渡の接続解析　　過渡解析で最初にsteady stateの解を得るために直流解析を行なうことはよくやられている．それと同様に交流解析の始めに，交流電源をゼロとして直流解析を行なうことも必要である．交流解析の前に直流解析を行なう意味は，直流解析の結果を用いて交流解析に使う素子値を決めるためである．直流解析と他の解析では使用する方程式（従って計算上のマトリックス）が異なる．また，トランジスタなどの等価回路が異なったりすることもある．たとえばトランジスタの等価回路として，直流解析では，エバース・モル・モデルの非線型回路を使い，交流解析ではHybrid π モデルを使うなどである（図2.4参照）．素子の数が少ないときは，直流解析をまず行ない，その結果を使って人間が手で交流解析の回路（回路としては，直流解析で用いたものと，等価であるが）の素子値を設定する，あるいは，過渡解析の初期値（状態変数解析法ではI_L, V_Cであることが多い）をセットすることはできる．しかし，少し素子の数が多くなると，それは大変厄介なものであり，また間違いが入りやすい．そこで直流解析と他の解析の自動接続が必要となる．しかし接続解析を要する二つの解析間で（たとえばトランジスタの等価回路として）異なる回路が使われる場合，一方の回路から他方の回路への移行は単純ではない．

その他：使用者定義の関数を自由に使いたいということがある．回路の非線型性でよく使用されるものは，組み込み関数として入れておいて，使用が便利なようにしておかねばならないが，それだけでは十分ではない．新らしい素子のモデリングの研究のため，あるいはモデリングの新らしい理論の成果を十分利用できるようにするために，任意の関数を，使用者が自分のプログラムで与えることができる必要がある．ところが，一般に問題向け言語で書かれたプログラムから，システム・プログラムとは別の時点で記述されたユーザを呼び込むことは整合性のcheckなどから簡単ではない．

2．2　HICADにおける回路記述言語

前記問題点をHICADでは，どのように解決しようとしているかを，ここでは実例（図2.1(a)(b)(c)）を用いて説明する．

回路記述の最小単位はbranchで，つぎの形式をしている．

<node>ー<node>，<素子名>＝<value>

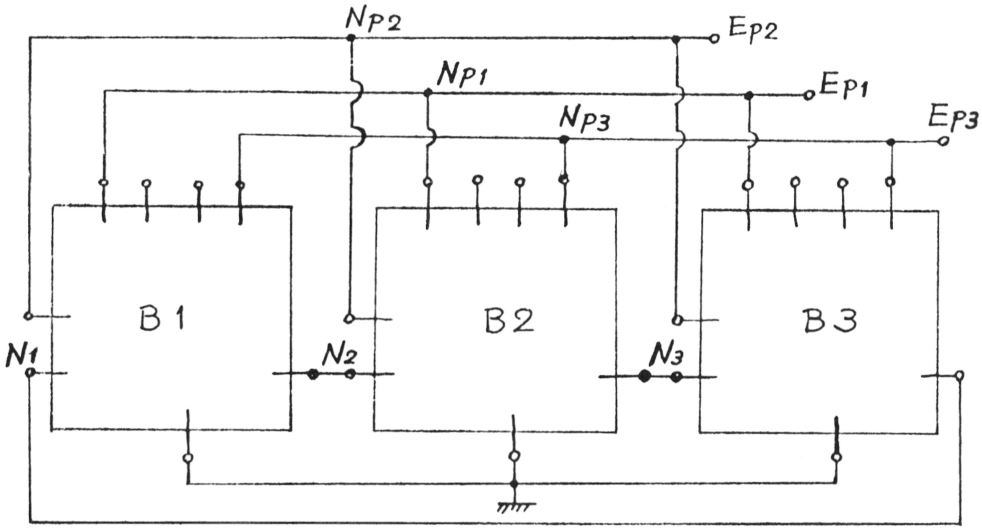
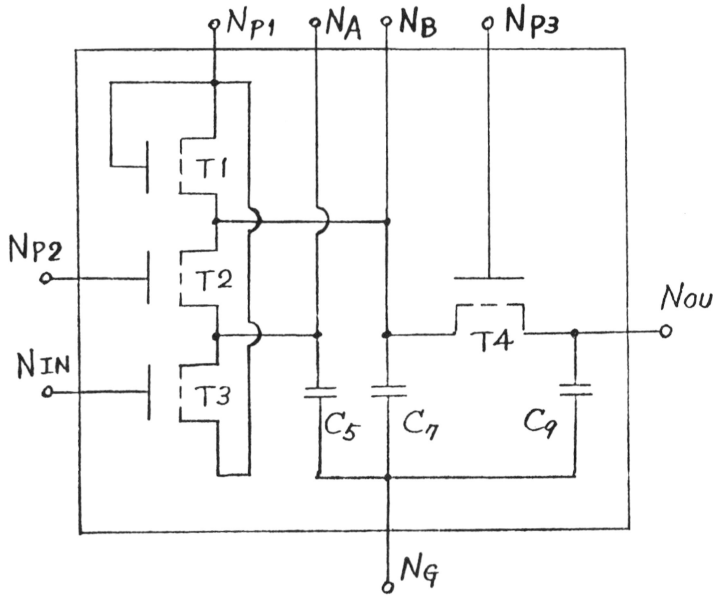


図 2.1(a)　MOSシフト・レジスタ回路（メイン）



MOS トランジスタは 別に サブ回路として記述する。

図 2.1(b)　MOSシスト・レジスタ1段回路（サブ回路）

```
*****  MICAD LIST  *****          PAGE  1

C
C SHIFT REGISTER CIRCUIT ANALYSIS
C
*  MICAD DESCRIPTION
      ELEMENTS SR3B
      NO  - VP1 , EP1 = PULSE ( TIME ,   0, 0, PL1, 0, PL3, AMP )
      NO  - VP2 , EP2 = PULSE ( TIME ,   0, 0, PL2, 0, PL3, AMP )
      NO  - VP3 , EP3 = PULSE ( TIME , PL2, 0, PL1, 0, PL3, AMP )
      NO  - VI  , EIN = PULSE ( TIME ,   0, 0, PL3, 0, 100,   5 )
      SET ( VI, N2, NO, NP1, NP2, NP3, NA1, NB1 ) , B1 = BIT
      SET ( V2, N3, NO, NP1, NP2, NP3, NA2, NB2 ) , B2 = BIT
      SET ( V3, N4, NO, NP1, NP2, NP3, NA3, NB3 ) , B3 = BIT
      END
C
C SUB CIRCUIT , BIT
C
      ELEMENTS BIT
      PORT ( NIN,NOU,NG,NP1,NP2,NP3,NA,NB )
      NG  - NOU , C9 = JCAP2 ( 0.6 , 0.0355P , 0.0206P )
      NG  - NB  , C7 = JCAP2 ( 0.6 , 0.149P  , 0.18P   )
      NG  - NA  , C5 = JCAP2 ( 0.6 , 0.0366P , 0       )
      SET ( NB , NG , NP1, NP1 ) , T1 = MOST ( 1 , 1.25 )
      SET ( NA , NG , NP2, NB  ) , T2 = MOST ( 1 , 1.25 )
      SET ( NP1, NG , NIN, NA  ) , T3 = MOST ( 1 , 1.25 )
      SET ( NOU, NG , NP3, NB  ) , T4 = MOST ( 1 , 1.25 )
      END
C
C SUB CIRCUIT , MOS TRANSISTOR MODEL
C
      ELEMENTS MOST ( A1,A2 )
      PORT ( NS,NSS,NG,ND )
      NG  - NS , IRGS = G(0)
      NS  - NSS, IRSS = G(0)
      ND  - NS , IRDS = MOST3 ( VRGS,VRSS,A1, 0.5 , 0.29 , 25U, 0,    C
                                          A2, 0.9 , 0    )
      END
C
C
C
*  COMPILE
      TYPE TR
C
C
C
*  ANALYSIS
      INITIAL NO
C
      MAX TIME    290N
      OUT INTERVAL 5
      STEP SIZE    1E-12
C
      LIST
      B1.VC5,B2.VC5,B3.VC5
      B1.VC7,B2.VC7,B3.VC7
      B1.VC9,B2.VC9,B3.VC9
      /
C
      PLOT1(,,,,,,30,,10N)
      B1.VC7,B2.VC7,B3.VC7
      /



*****  MICAD LIST  *****          PAGE  2

C
      MODIFY
      PL1=15N
      PL2=30N
      PL3=45N
      AMP=7
      /
C
      EXECUTE(LIST,PLOT1)
```

図2.1(c)　MOSシフト・レジスタの記述例

node名の先頭文字はN，素子が，電圧源，電流源，抵抗，インダクタ，容量に対応してそれぞれ，E，J，R，L，Cの文字を素子名の先頭に置くものとする．ブランチの仮想電流方向は，前のnode名から後のnode名に流れるものとする．

サブ回路の取扱いの問題を解決するため，すなわち回路記述の簡略化を図るために，メイン回路，サブ回路共，独立した記述単位とし，ELEMENTS文で始まり，END 文で終るものとする．サブ回路の記述中にサブ回路を取り込んだメイン回路と接続するnodeを指定するものが必要である．それにはPORT文を使い

PORT(<node>，<node>，……)

の形式で書き，ELEMENTS文の直後におく．これに対応してメイン回路には，サブ回路と接続するnode名，それにネスト名およびサブ回路名を指定するSET文がある．その記述形式は

SET(<node>，……)，<ネスト名>=<サブ回路名>(<実引数>，……)

である．サブ回路の中に他のサブ回路を使うこと，すなわちサブ回路の多段使用もできるようにする．

つぎに，素子値や素子変更の初期値のモディファイに関しては，2つの方法を取ることができるようにする．すなわち，素子値などの直接指定変更と，素子値をパラメータを用いて表現(関数を使用する場合もある)し，そのパラメータを変更させることによって，素子値などを変更するの2方法である．パラメータは一つのシミュレーションの間では，一定であり，その意味ではそれは定数扱いであるので記号定数と呼ぶことにする．記号定数を関数のパラメータに使って表現することにより，素子値間の相関的記述，および相関的モディファイが可能になる．素子値やパラメータ・モディファイの指定はオペレーショナル言語のプログラム(次節参照)で行なうのである．

問題点(3)の接続解析における回路の切替えの方法に関しては，使用者がよく知っていることであるから原則として使用者に切替えの指定をさせるのがよいのではないかというのが筆者らの考えの出発点である．そこで回路解析システムとしては，回路記述言語の機能を拡張し，解析間の回路トポロジーの自動変更やデータの受渡し表現が容易にできるように構成する．[2)]

(1) branchの表現　前述のbranchの表現機能を拡張して

<node>-<node>，<素子名>=<value>，/<anal-designator>/

とする．ここで<anal-designator>は，そのbranch が意味を持つ解析の種類を指示する．ただし，電源はそれ自身の中にどの種類の解析で意味を持つかわかるので<anal-designator>は，おかない．

(2) 直流解析から他の解析へ接続する際にデータの受渡しを指定する文　直流解析の結果を，過渡または，交流解析に受渡す文として，JOIN文と代入文をおく．代入文の形式は次のとおりとする．

<symbolic constant>=<value>

<branch current name>=<value>

<branch voltage name>=<value>

<symlolic constant> は解析の前に値を与えてやることはできるが，解析の途中では変化しないものとする．<value>は次の(a)，(b)，(c)のいずれか一つであるとする．

(a) constant

(b) branch current または branch voltage の値

(c) (a)，(b)を変数とする関数の一つ

左辺の変数名は，接続される解析，すなわち交流または過渡解析の変数を指し，右辺の変数名は，直流解析の変数を指示するものとする．

JOIN文は接続解析のための代入文ブロックの先頭におく(一種の宣言文である)．以下に説明の補足をかね，交流解析用の素子値を決めるために初め直流解析を実行する場合の実際例を述べる．

(3) 接続解析の回路記述の例

使用する回路は，図2.2のAmplifierであるとする．図2.2の中で使用しているトランジスタの等価回路として，直流解析ではエバース・モル・モデル(非線型回路)とし，それに対して交流解析ではHybrid πモデル(線型回路)を使用するものとしよう．エバース・モル・モデル，Hybrid πモデルを，それぞれ図2.3，2.4に示す．

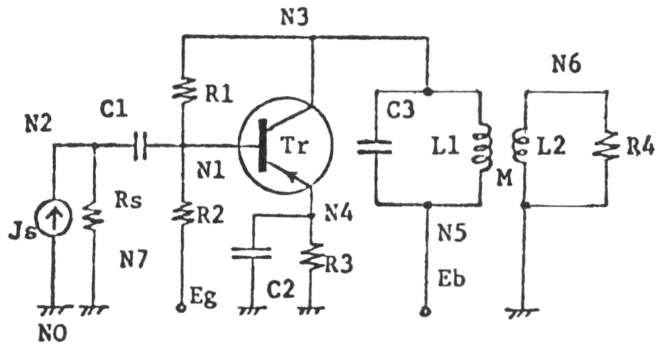


図2.2 Amplifier

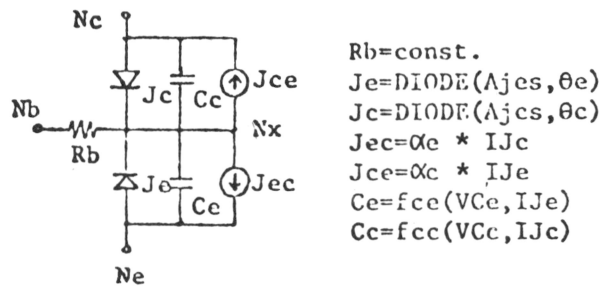
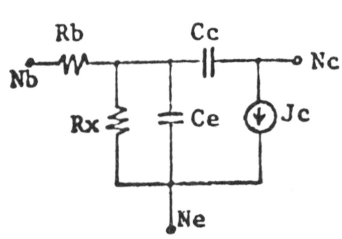


図2.3 Ebers-Moll Model




```
Rb=const.
Rx=1/(θe*dc(IJe)*(1-αc))
Ce=fce(dc(VCe),dc(IJe))
Cc=fcc(dc(VCc),dc(IJc))
Jc=αc*θe*dc(IJe)*VRx
  Cf. dc():result of dc
      Ixx :current
      Vxx :voltage
```

図2.4　Hybrid π Model

そのとき，交流解析で使用するHybrid πモデルの素子値，たとえばC_eに関していえば，直流解析で求められたVC_e, IJ_e（これを今$d_c(VC_e)$, $d_c(IJ_e)$であらわす）によって決まる値（今，これを，$f_{ce}(d_c(VC_e), d_c(IJ_e))$で表現する）である．$R_x$, C_cに関しても同様である．J_cは従属電流源で

$$J_c = G_m \times VR_x$$

で表わされるがG_mは

$$\alpha_c \times \theta_e \times d_c(IJ_e)$$

である．

したがって，図2.2の回路記述は，図2.5であり，トランジスタの等価回路図2.3，2.4の回路記述は合併して図2.6のようになる．<anal－designator>が前文と同じ場合は省略する．

図2.6のJOINT文以下にある文で直流解析の計算結果を使って交流解析の素子値や，電流増幅率G_mをセットするのである．

```
ELEMENTS AMP
N0-N2,Js=Ac(1,0)          ··· AC source
N2-N0,Rs=500
N2-N1,C1=0.05E-6
N0-N7,Eg=1                ··· DC source
   :
   :
SET(N4,N1,N3),Tr=Tr#1(40,0.99,40,0.5)   ···call model
N6-N0,L2=40E-6
N6-N0,R4=500
MUT(L1,L2),M=99E-6        ··· Mutual inductance
END
```

図2.5　Network Description 1

```
ELEMENTS TR#1 (θc,αc,θe,αe)
PORT (Ne,Nb,Nc)
Nb-Nx,Rb=98                    ,/DAT/
Nc-Nx,Jc=DIODE (Ajcs,θc)       ,/DT/
      ,Cc=fcc (VCc,IJc)
Nx-Nc,Jce=αc*IJe
Ne-Nx,Je=DIODE (Ajes,θe)
      ,Ce=fce (VCe,IJe)
Nx-Ne,Jec=αe*IJc
      ,Rx=A1                   ,/A/
      ,Ce=A2
Nx-Nc,Cc=A3
Nc-Ne,Jc=Gm*VRx
JOIN                           ,/A/
   Gm=f2(dc(IJe),αc,θe)
   A1=f1(dc(IJe),αc,θe)
   A2=fce(dc(VCe),dc(IJe))
   A3=fcc(dc(VCc),dc(IJc))
END
   Cf.   /A/ ... AC
         /DT/ .. DC,Transient
```

図2.6 （b） Network Description 2

2．3 HICADにおけるオペレーショナル言語

一般に回路設計にシミュレーションを応用するとき，一つの回路記述に対して，素子定数や初期値の変更，あるいは出力指定の変更などを行なって，数回乃至十数回シミュレーションを行なう，このような使用に便利であるように“回路記述”と“シミュレーションの制御や素子定数変更などを指定する文の記述”を分離している．後者のシミュレーションの制御などを指定するものをオペレーショナル言語と呼び，それには次の文がある．

(1) どの状態のシミュレーションを実行するかを指定する文
(2) 過渡解析のときに初期条件解析を行なうかどうかを指定する文
(3) 過渡解析のときに最大解析時間，最初の時間刻み幅を指定する文
(4) 過渡解析において時間刻み何回ごとに出力するかを指定する文
(5) 過渡解析において解析終了を回路変数条件で指定する文
(6) ラインプリンタに，指示した回路変数をリスト形式で出力する文
(7) ラインプリンタに，指示した回路変数をプロット形式で出力する文
(8) メーソプロッタに，指示した回路変数の変化をプロットさせる文
(9) 解析結果の印字にユーザ指定のラベルを出力する文
(10) 素子値，回路変数の初期値，および記号定数の値，の設定，変更を指定する文

3．シミュレーション処理の概略

回路シミュレーションの処理を三段階に分ける．第一段階は，回路の記述のよみこみ，文法

上のチェックを行ない，次の段階で処理しやすい中間データの形に変換し，一つのファイルとする．これを“回路データ・ファイル”と名づける．この段階の処理は，解析の種類に無関係な共通したもので回路記述の単位毎（サブ回路も一つの単位）に行なう．第二段階では解析の指定をよみこみ，第一段階で作り出された“回路データ・ファイル”から解析の指定（DC，AC，Trの何れか）に対応する<anal－designator>をもつbranchをとり出し，シミュレーション・ランで使用するマトリックスの作成を行なう．その外，この段階の主な仕事にメイン回路と，サブ回路のつなぎがある．この段階で作られるファイルを“解析データ・ファイル”と名づける．第三の段階は，“解析データ・ファイル”をもとにして，シミュレーションを実行するところである．シミュレーション結果をある基準で判断し，それにより，たとえば素子値を変更して，くり返しシミュレーションを実行するなどのシミュレーションの制御を行なうために，使用者は，そのシミュレーション制御の手順をオペレーショナル・コマンドを用いて書く．このシミュレーション制御プログラムの実行は，第三段階である．以上の処理構成をDC－ACの接続解析を例にとり，処理の順序を示すと次のようになる．またブロック図を図3.1に示す．

(1) 回路記述単位毎に“回路データ・ファイル”を作る（図3.1の①）

(2) “回路データ・ファイル”から<anal－designator>がDCであるbranchをとり出し，直流解析用の“解析データ・ファイル”を作る（②，③）

(3) シミュレーション制御プログラムをコンパイルし，実行プログラムを作成する（④）

(4) 直流解析を実行する（⑤，⑥，⑦）

(5) 直流解析の結果を保存する（実行プログラム，Mプログラム，⑧）

(6) “回路データ・ファイル”から<anal－designator>がACであるbranchをとり出し，交流解析用の“解析データ・ファイル”を作る（⑧，⑨，⑩）

(7) 直流解析の結果を交流解析の初期値としてセットする（実行プログラム，Mプログラム，⑧）

(8) 交流解析を実行する（⑪，⑫）

〔注〕第1段階で動作するシステム，プログラム……Phase 1
第2段階 〃 〃 ……Phase 2
第3段階 〃 〃 ……M-Program, Phase 3

以上のことからもわかるように，第一段階と第二段階は切り離されている．第二，第三段階は，連続動作もできるし，また切り離してもできる．第二段階で取り込むサブ回路（トランジスタなどの等価回路，あるいは，それらを中に含んで，外から呼び出されて使われる回路）は，“回路データ・ファイル”からである．このような構成とすることにより，処理は簡単になり，処理スピードが向上する．更に直流－過渡，直流－交流の接続解析を容易にする．

またこのような記述法を使えば，従来異種の解析間で同一の回路を異なる記述で表現しなければならなかったようなものが，直流，交流，過渡の何れの解析でも使えるユニバーサルな回

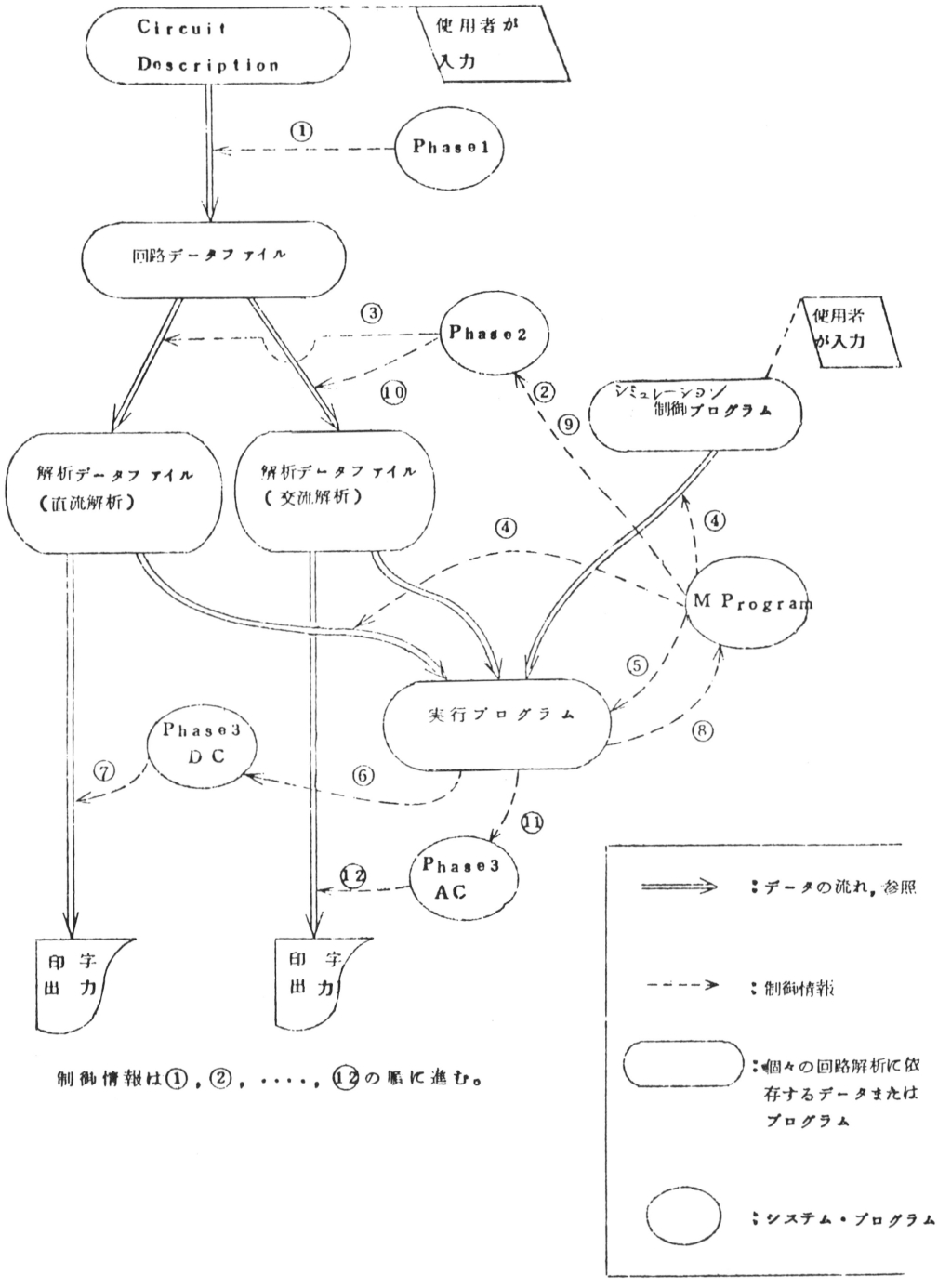


図 3.1　直流ー交流の接続解析に注目したブロック図

路が記述できることになる.

4. スパース行列演算処理の一方法

CADとしての電子回路動作シミュレーションで，トランジスタなどの物理素子のモデリングや，人間とのインターフェイスの問題を一応さておき，純粋に数学的な解析の問題に限るとき

(1) 回路解析の定式化

(2) 微分方程式の解法

(3) 大規模行列演算を含む線型計算

の問題が中心であるとみることができる.

CADとして電子回路シミュレーションを使用するには，過渡解析では，数百及至数千の時間刻み毎の数値積分と，連立方程式の解法を含む線型計算の実行がある. 解析の中で取扱う行列の大きさは，(200～300)×(200～300)のものもあり，尤大な計算とメモリ・エリアがが必要となる. このために第一世代の回路シミュレーションプログラムは，取扱える回路の規模に大きな制限を受けていた.[3] ところが，回路シミュレーションで扱われる行列は，ゼロ要素を多く含んでいるのが普通であり，回路の規模が大きくなればなる程，ゼロ要素の割合が高くなる傾向にある. ゼロ要素の多い行列はスパース行列(Sparse Matrix)と呼ばれている. (スパース行列が現われるのは，回路解析に限られたことではなく構造解析，電力システムの解析，あるいはLPで扱われる係数行列などがある). そこでスパース行列の性質を利用して，回路解析で現われる大きな行列を，効率よく扱えないものかということが問題となった.

連立一次方程式の係数行列がスパースである場合，その解法にクラウト法やガウスの消去法の適用が効率良いといわれ，実際に巧みな適用で，桁違いに計算時間が短かくなった例が報告されている.[4] ここではHICADに組み込まれつつあるスパース行列の積および転置の演算について述べる.

まず計算スピードを上げるためにゼロ要素があった場合，演算を行なわないでスキップする方法(ゼロスキップ法)が当然考えられる. この方法は要素毎にゼロかどうかを判定することになるので，速さはそれほど変らないし，またメモリ・エリアは全く得をしない. そこで，つぎに非ゼロ要素のみを記憶して演算を行なう方法を考える.

4. 1 スパース行列に適した積演算

$m \times n$ の行列 A と，$n \times q$ の行列 B の行列の積を

$$(a_{ij}) \times (b_{ij}) = (z_{ij})$$

$$z_{ik} = \sum_{j=1}^{n} a_{ij} \cdot b_{ij}$$

とするとき，積の結果の行列 (z_{ij}) の i 行の要素はベクトルで，つぎのように表現できる.

$$(z_{i1}, z_{i2}, \cdots\cdots, z_{iq}) = (\sum_{j} a_{ij} \cdot b_{j1}, \Sigma a_{ij} \cdot b_{j2}, \cdots\cdots, \Sigma a_{ij} \cdot b_{iq})$$

$$= \sum_j (a_{ij} \cdot b_{j1}, \cdots\cdots, a_{ij} \cdot b_{jq})$$

$$= \sum_j a_{ij}(b_{j1}, \cdots\cdots, b_{jq}) \qquad \cdots\cdots\cdots\cdots (1)$$

非ゼロ要素a_{ij}を一つ固定し行列(b_{ij})のj行の要素(非ゼロ要素のみ)と積を作りながらz_{ik} ($k=1, 2, \cdots\cdots, q$)に足し込んで行く．これを$j=1, 2, \cdots\cdots, n$について行なって一行分の結果を一度に出すのである．この方法をline法と呼ぶことにする．ここで行列Bの要素が行毎に連らなっているデータ構造であれば計算は早い．式(1)で，もし，a_{ij}がゼロのときは

$$a_{ij}(b_{j1}, b_{j2}, \cdots\cdots, b_{jq})$$

は計算する必要はない．またb_{je}がゼロのところも$a_{ij} \times b_{je}$は計算する必要はない．したがってスパースティ(ゼロ要素の割合)が高くなればなるほど，急速に乗算回数が減ずることは明らかである．そのためにデータ構造が特殊になって，それに伴ない余分な処理が必要になるかもしれないが演算時間およびメモリ・エリアは格段に小さくなることが期待される．これらの処理に都合がよいようにデータ構造を定める．

4．2　スパース行列のデータ構造

行毎に第一行目から順次非ゼロ要素のみを実数の1次元配列(これをAMTXと呼ぶ)につめて記憶する．何行目の何列目の要素が非ゼロであるかに相当する情報も整数の1次元配列(これをMINFと呼ぶ)につめて記憶する．そのほか，行列の行数，列数およびMINF，AMTXへのポインタなどを記憶するテーブル(これをMLと呼ぶ)をおく．具体的には$m \times n$行列のデータ構造をつぎのようにする．

まず，i行目の非ゼロ要素の列番号および要素につぎの記法を用いる．

列番号：i_1，i_2，i_3，……，i_{pi}

要　素：$a_i^{i_1}$，$a_i^{i_2}$，$a_i^{i_3}$，……，$a_i^{i_{pi}}$

そのとき，行列要素を図4.1のように記憶する．

4．3　$A \times B$

前節のデータ構造をbaseとして行列の乗算$A \times B$のアルゴリズムを例で述べる．

$$A = \begin{bmatrix} a_{11} & 0 & a_{13} \\ a_{21} & 0 & 0 \\ a_{31} & a_{32} & 0 \end{bmatrix} \times B = \begin{bmatrix} b_{11} & b_{12} & 0 \\ 0 & b_{22} & 0 \\ 0 & b_{32} & 0 \end{bmatrix} = Z = \begin{bmatrix} z_{11} & z_{12} & z_{13} \\ z_{21} & z_{22} & z_{23} \\ z_{31} & z_{32} & z_{33} \end{bmatrix}$$

乗算結果の一行分をおくワーキングエリアを用意しておく．それをw_1, w_2, w_3とする．

(1)　ワーキングエリアをクリアする．

(2)　a_{11}を固定して，行列Bの一行目とかけて，その結果をワーキングエリアに加算する．

Bの要素の列番号とワーキングエリア番号が一致するところに加算する．

すなわち

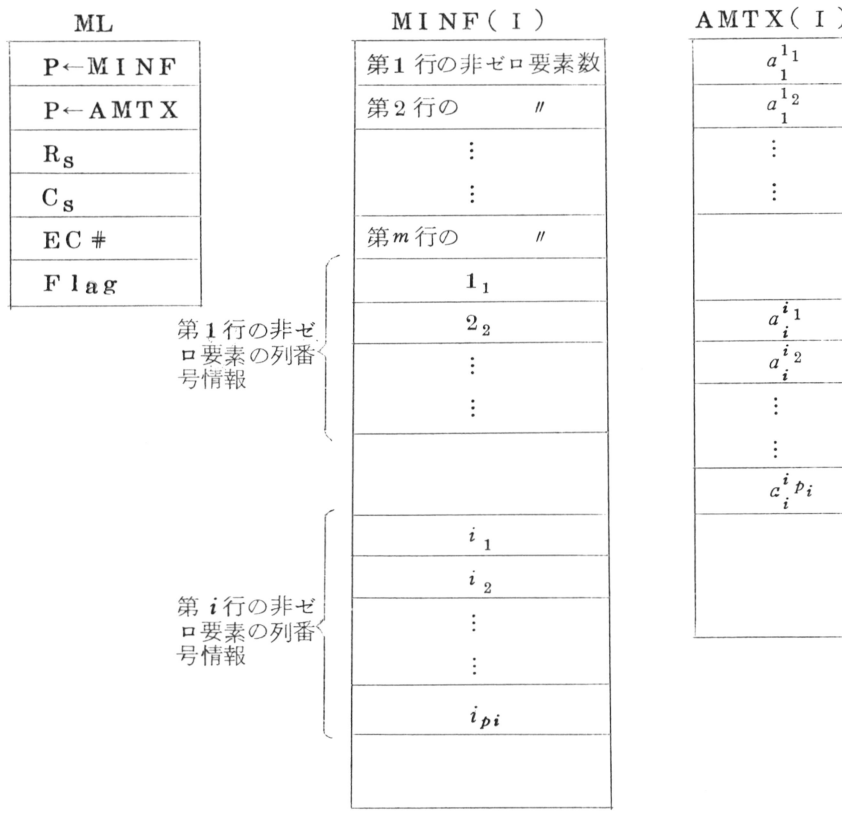


図4.1　スパース行列のデータ構造

$(w_1) + a_{11} \times b_{11} \rightarrow w_1$

$(w_2) + a_{11} \times b_{12} \rightarrow w_2$

(3)　a_{13} を固定し，行列 B の三行目とかけてその結果をワーキングエリアに加算する．

$(w_2) + a_{13} \times b_{32} \rightarrow w_2$

この例では，(2)，(3)で A の第一行の全要素と B との要素の積が全部すんだことになる．

(4)　乗算結果の一行分がワーキングエリアにできている筈であるから乗算結果の非ゼロ要素のみ，何列の要素かの情報もつけて行列 Z のデータエリアにつめる（データ構造は図4.1と同じにとる）

以上，(1)～(4)を行列 A の各行にくり返えせば乗算の結果が得られる．

ここで注意すべきは，A，B とも行毎に要素が参照され，Z も行毎に作り出されることである．

このアルゴリズムを Sparse－line 法と呼ぶことにする．これに対し従来の行列積のアルゴリズムを一般法と呼ぶことにする．

4．4　スパース行列の転置

4.2で定めた，データ構造の行列の転置のアルゴリズムは次のようにする（ここで述べるアルゴリズムは基本的には文献5のMcNameeの方法と同じである）．

$A^T = Z$としたとき，ZはAを二度の行毎の参照で作る．最初の参照でZの各行に入る非ゼロ要素をカウントし，Zの各行のエリアを決める．第二回の参照でAからZに要素を移す．

4．5　方式の比較検討

演算実行時間と所要データエリアに関し，つぎの(A)～(F)の6つの行列積演算法を比較する．

(A)　行列積演算でもっとも一般に行なわれている（すなわち，$Z_{ik} = \sum_j a_{ij} b_{jk}$の）方法（一般法）．

(B)　行列の要素が非ゼロかどうかを見て，非ゼロの場合のみ要素間の乗算を行なう行列積演算法（ゼロスキップ法）．

(C)　第4.1節で述べた方法をFull行列間の積に適用したもの（Full－line法）．

(D)　McNameeの方法[4]（$A \times B^T$が基本演算となっている）．

(E)　アルゴリズムは，McNameee法で行列のデータ構造を第4．2節で述べた形式をとるもの（Modified McNameee法）．

(F)　第4.2節で述べたデータ構造にline法（第4.1節参照）を適用したもの（すなわち第4.3で述べた方法）（Sparse－line法）．

(1)　演算実行時間　　比較は，行列A，Bが共にSparsity　90，70，50，30，0の%の場合について行なう．ここでSparsity　90%の行列とは各行毎に，ゼロ要素の割合が90%（したがって非ゼロ要素の割合10%）で，かつ，主対角は必らず非ゼロ要素の行列であるものとする．ゼロ要素の行列の中での位置は主対角を除き擬似乱数を発生させてきめる．行列の大きさはA，Bとも50×50とする．$A \times B$の演算実行時間比較を図4.2に示した（H5020 FORTRANでプログラムし，比較した）

(2)　所要データ・エリア　　データの所要メモリ・エリアは

(A)　一般法〔(B)，(C)法のデータ構造は(A)と同じ〕

(D)　McNameee法

(F)　Sparse－line法〔(E)方式のデータ構造はFと同じ〕

の間で比較すれば十分である．

FORTRAN言語をbaseとして考えて，データ構造（4.2節）のML，MINFは，整数の一次元配列であるが，2byte整数が使える場合（たとえばH8000シリーズ）と4byte整数だけしか使用できない場合では，その部分のデータ・エリアは二倍の開きとなる（要素の値は実数4byte換算とする）．$n \times n$行列で2byte整数，4byte整数に対応して各方式のデータ・エリアは次式で表現される．

〔 〕はガウス記号

(i)　4byte整数のとき

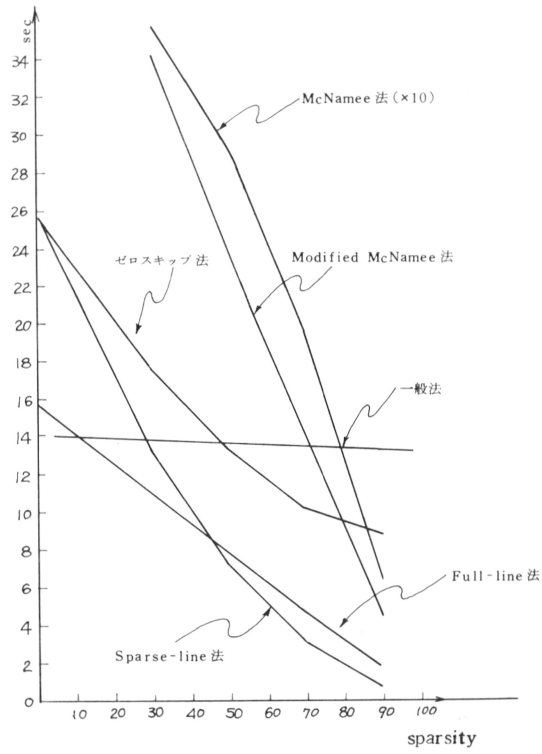


図4.2　行列積演算$A \times B$の実行時間比較

Ⓐ：　　n^2　　$\times 4$（byte）

Ⓓ：$(\lceil(1-\alpha)\times n^2\rceil+\lceil\frac{n}{2}\rceil+\lceil\frac{(1-\alpha)\times n^2}{2}\rceil+4)\times 4$

Ⓕ：$(\lceil(1-\alpha)\times n^2\rceil^{注1}+n^{注2}+\lceil(1-\alpha)\times n^2\rceil^{注3}+6)\times 4$

(ii)　2byte整数のとき

Ⓐ：　　n^2

Ⓓ：$\lceil(1-\alpha)\times n^2\rceil\times 4$（byte）$+(n+\lceil(1-\alpha)\times n^2\rceil+4)\times 2$（byte）

Ⓕ：$\lceil(1-\alpha)\times n^2\rceil\times 4$　　$+(n+\lceil(1-\alpha)\times n^2\rceil+6)\times 2$

sparsityが90，70，50，30％の各場合における所要データ・エリアの比較を図4.3に示す．この図から推量されるごとく，2byte整数では，Sparse line法，McNamee法

脚　注；注1）非ゼロ要素のエリア（AMTX）
注2）各行の非ゼロ要素数を入れるエリア（MINFの一部）
注3）非ゼロ要素の列番号を記憶するエリア（MINFの一部）
注4）行数，列数，非ゼロ要素数などを記憶する（ML）

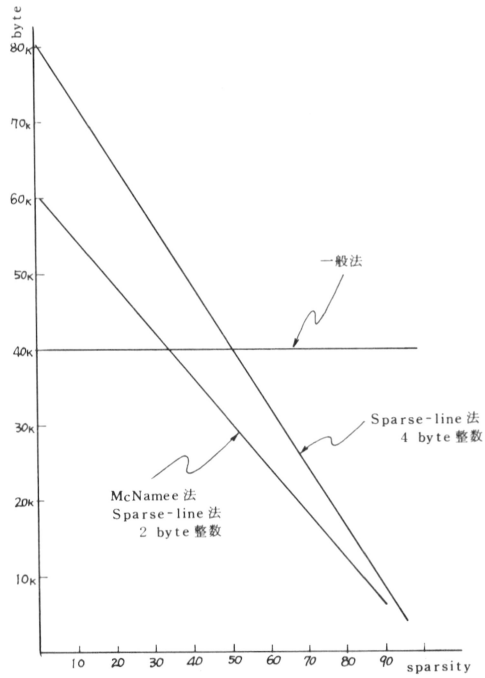


図 4.3　所要データ・エリアの比較

共sparsityが33.7％まで下がるまでFull行列のエリアを越えない（4byte整数ではSparse－line法はsparsity　50.5％でFull行列のエリアとクロスする）.

5. あとがき

HICADは，始めH5020TSSの上で動く会話型のシステムが開発された．つぎにBatch処理用のプログラムが計画，開発に移され，一部は稼動している．何れも直流および過渡動作シミュレーションが主である.

TSS上の会話型システムは，計算時間のかかる大型回路や出力の多い過渡応答のシミュレーションには適さない．しかし，精密なトランジスタ・モデルや寄生素子を組み込んだ精密なシミュレーションの前の，大まかな見当をつけるためのシミュレーションには，会話型は便利である．会話型，Batch処理型の使いわけが必要である.

今後の問題としては，トランジスタなどのモデル・バンクの作成，Sparse 行列演算処理手法を更に追求してシミュレーションの高速化を計るなどのことがある.

最後に，HICADに関して有益な御意見をいただいた東工大岸源也教授，日立中研永田，谷口両主任研究員に厚く感謝致します.

6. 参考文献

1) 佐藤，勝枝外；昭45年信学全大 918，919，920（昭45，8）

2) S. Sato, M. Katsueda, et al; Proc. KICCST 1970, A-3-2

3) H. B. Lee; 1969 WESCON Technical Papers, Section 23/1 pp1-3

4) G. D. Hachtel, et al; IEEE Trans. CT-18 No. 1(1971) p101

5) J. M. McNamee; CACM Vol 14, No 4, pp256-273 (1971, 4月)

本 PDF ファイルは 1972 年発行の「第 13 回プログラミング―シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの https://www.ipsj.or.jp/topics/Past_reports.html に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の捜索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場 (=情報処理学会電子図書館) で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者（論文を執筆された故人の相続人）を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者捜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思います。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長 (tsuji@math.s.chiba-u.ac.jp) までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日 〜 2021 年 3 月 19 日
掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程
https://www.ipsj.or.jp/copyright/ronbun/copyright.html