並列 MOS 論理シミュレーションの回路記述とそのコンパイラについて†

竹之上 典昭†† 古賀 義亮†††

MOS 論理回路の論理シミュレーションにおいてスイッチレベルの論理シミュレーションが注目されている．スイッチレベルの論理シミュレーションは，MOS 論理回路をスイッチネットワークとして捉えその論理を解析する方法であり，MOS 論理回路の論理を忠実にシミュレートすることができる．スイッチレベルの論理シミュレーションには Bryant や Hayes などが提案する方法が知られているが，LSI 等多数の MOS トランジスタをシミュレートする方法としてネットワーク型コンピュータによる並列 MOS 論理シミュレーションも提案されている．並列 MOS 論理シミュレーションは，ネットワーク型コンピュータを用いてスイッチネットワークをシミュレートするため，MOS 論理回路の入力が問題点の一つになっている．本論においては，従来人手で行っていた入力を自動化するための回路記述言語を与え，これをコンパイルする方式について述べるとともに実際にコンパイラを試作し，この方式の有効性，実用性を明らかにする．

1. はしがき

MOS 論理回路（MOS トランジスタで構成された論理回路）の論理シミュレーションの研究が活発に行われている．MOS 論理回路の論理シミュレーションには，大きく二つの方法がある．その一つは MOS 論理回路を AND, OR 等の論理素子（以後ゲートと呼ぶ）で構成された回路に等価変換し，ゲートレベルの論理シミュレーションを行う方法[4)-6)]であり，もう一つは MOS トランジスタをスイッチとして捉えスイッチレベルの論理シミュレーションを行う方法[2),3),8)]である．

ゲートレベルの論理シミュレーションは，従来の方法に依存するものであり[1)]，MOS 論理回路からゲートの論理回路への変換ツールも研究されている[9)]．

スイッチレベルの論理シミュレーションは，MOS トランジスタの接続情報等の回路の形態を残して論理シミュレーションを行うものである．その方法には MOS 論理回路を小さな回路に分割し，スイッチネットワークとして論理シミュレーションを行うもの[3)]，MOS 論理回路全体をスイッチネットワークとして論理シミュレーションを行うもの[8)]，さらに MOS 論理回路全体をスイッチネットワークとして扱いネットワーク型コンピュータによる並列 MOS 論理シミュレーション（以後並列 MOS 論理シミュレーションと呼ぶ）を行うもの[2)]等がある．

並列 MOS 論理シミュレーションは，LSI 等 MOS トランジスタ数の多い回路をシミュレートする方法として優れている．しかし，この方法では MOS 論理回路を従来の AND, OR 素子として扱わず，ネットワークとして表すために MOS 論理回路をいかに表現してシミュレーション入力とするかという問題がある．すなわち，並列 MOS 論理シミュレーションにおいてネットワーク型コンピュータの上に MOS 論理回路のスイッチネットワークをいかに展開するかという問題として捉えることができる．このような展開は人手に頼って行っているが，回路の構造とネットワーク型コンピュータの構造を適合させるため，MOS 論理回路の規模が大きくなるとシミュレーションの入力のうち MOS 論理回路をネットワーク型コンピュータの上へ人手によって展開することは困難になる．並列 MOS 論理シミュレーションを行うには MOS 論理回路の入力を容易にし，いかなるネットワーク型コンピュータの構造にも対処可能な方法が必要である．

この論文では，このようなシミュレーション入力を生成する困難性を解決するために並列 MOS 論理シミュレーション用回路記述言語 PALM (PArallel logic simulation description Language for Mos transistor circuits) を提案する．さらにこのように記述された MOS 論理回路をスイッチレベルのネットワークに展開するとともに，与えられたネットワーク型コンピュータ上へ適合させるための PALM コンパイラの開発を行ったのでこれについて述べる．従来のコンパイラのように言語で記述された文を機械語に変換するばかりでなく，ある構造を持つ原処理を別の構造を持つ

† A Circuit Description Language and Compiler for Parallel MOS Logic Simulation by NORIAKI TAKENOUE (Department of Ground Defence Science, National Defence Academy) and YOSHIAKI KOGA (Department of Electrical Engineering, National Defence Academy).

†† 防衛大学校陸上防衛学教室
††† 防衛大学校電気工学教室

処理系に適合させる機能を持つことがこのコンパイラの一つの特徴である．このような方式の有効性，実用性を明らかにするために実際にこのコンパイラを試作したのでそのコンパイラの構造を述べ，コンパイル例を通してネットワーク型コンピュータ上でネットワークの問題を処理するために必要な機能について説明する．

2. スイッチレベル論理シミュレーションの必要性

MOS 論理回路のスイッチレベルのシミュレーションでは従来行われてきたゲートレベルのシミュレーションとは本質的に異なる面がある．

ゲートレベルの論理シミュレーションでは AND, OR 等の論理素子がシミュレーションの基本単位であるが，スイッチレベルのシミュレーションでは MOS トランジスタ（スイッチ）が基本単位となる．

スイッチを結合したネットワークにおいてはトランスファゲートなどのようにスイッチを流れる電流の方向はそれを構成するネットワーク全体の動作によって決定され，トランスファゲートのトランジスタには電流の方向を制御する機能はない．したがって各種論理素子を構成する複数のトランジスタに関して，スイッチレベルの振舞いをシミュレートすることが必要となる．

スイッチレベルの論理シミュレーションは論理回路設計者にとって論理設計上きわめて有用な道具となる．例えば，**図1**の MOS 論理回路などはそのよい例であろう．これは NMOS と PMOS を混在させた3入力の排他的論理和（XOR）の回路であり八つのトランジスタにより構成されている．通常のブール代数による論理合成から構成された NMOS 回路の場合の 1/2 から 1/3 のトランジスタ数で構成されている．もしこの回路をゲートレベルの論理シミュレーションで行うならば AND, OR, NOT 論理素子によって構成された XOR ゲートとして処理され，スイッチレベルの論理シミュレーションはできないことになる．スイッチレベルの論理シミュレーションを行うことによって，より少ないトランジスタによる論理回路が構成できる場合があることを示している．このようにスイッチレベルの論理シミュレーションはトランジスタレベルでの論理設計の分野を開拓する一つの大きな原動力となる道具を提供するものである．

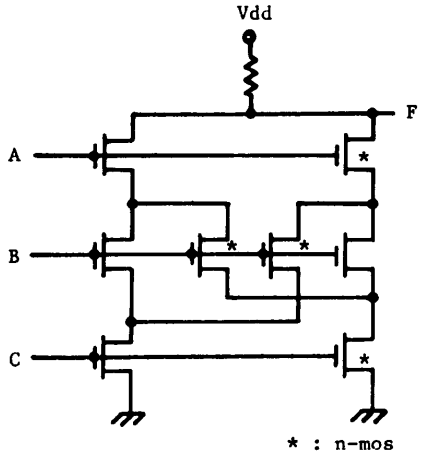

図 1 3入力の排他的論理和の一例
Fig. 1 An example of 3-input Exclusive-OR circuit (called 3-XOR).

3. 並列 MOS 論理シミュレーションの方法

並列 MOS 論理シミュレーションは，ネットワーク型コンピュータとして構成された並列処理用コンピュータを用い，MOS 論理回路のスイッチネットワークの論理動作をシミュレートする[2]．以下に並列 MOS 論理シミュレーションのための構成要素及び方法について述べる．

3.1 ネットワーク型コンピュータ

並列処理を行うコンピュータとして，ここでは三つの相互接続ができるポートを持つコンピュータをノードとしてネットワーク状に結合したネットワーク型コンピュータを対象とする．各ノードコンピュータはプロセッサ・メモリ等を内蔵しており単独で動作することも可能である．すべてのノードコンピュータは同一な機能をもっておりネットワーク内におけるノードコンピュータ間の親子関係はない．ネットワーク型コン

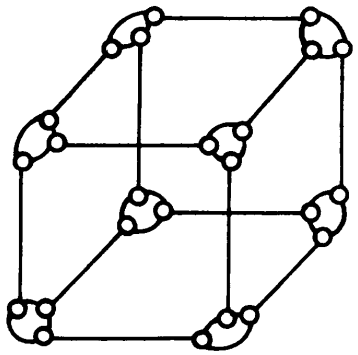

図 2 ネットワーク型コンピュータの一例
（二つのノード間の最大距離が $O(\log_2 N)$ の結合）
Fig. 2 An example of Network-computer: CCC. (Distance are $O(\log_2 N)$.)

ピュータを構成するノードコンピュータの個数及び結合法は自由に設定でき図2のようなCCC (Cube Connected Cycle) と呼ばれる結合をしたものでもよい．CCC は二つのノード間の最大距離が $O(\log_2 N)$ 程度であり，並列処理等を行うために最も適したネットワーク構成法の一つといわれている[7]．

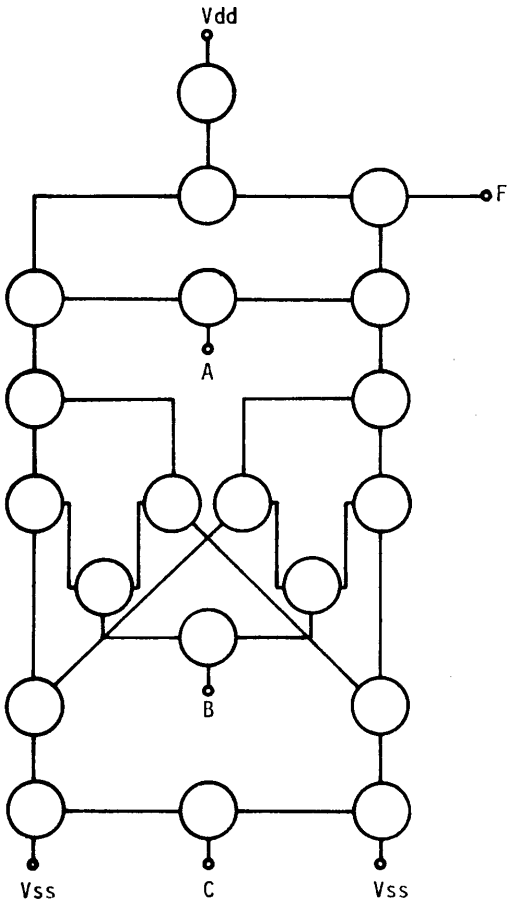

図 3 タスクネットワーク（排他的論理和回路）
Fig. 3 Task network of 3-XOR.

3.2 MOS 論理回路の展開（回路の入力）

MOS 論理回路のシミュレーションを行うためには，MOS 論理回路をネットワーク型コンピュータ上にいかに展開するかという問題がある．ここでは，次のような方法により MOS 論理回路の展開を行っている．

まず MOS 論理回路を MOS トランジスタ（スイッチ）・抵抗・結線の各素子の回路ネットワークに展開する．これらの素子に関する動作をネットワーク型コンピュータのノードコンピュータに配分し並列に処理できるような形式のタスクとする．このタスクを接続したネットワークを“タスクネットワーク”と呼ぶ．

例えば，図1の排他的論理和の回路をタスクネットワークに置き換えると図3のようになる．このタスクネットワークをネットワーク型コンピュータ上に展開する．すべてのタスクは各ノードコンピュータに配分される[2]．

3.3 タスクの機能

各ノードコンピュータに配分されたタスクは三つの相互接続された手（抵抗の場合は二つの手）を有し，相互に接続されたタスク間で通信を行うことによりスイッチレベルの論理機能をシミュレートする．

タスクには表1のように NMOSGATE (N タイプの MOS トランジスタ)，PMOSGATE (P タイプの MOS トランジスタ)，RESISTOR(抵抗)，CONNECTION（結線）の機能がある．タスクの手は機能・接続するタスクによってその属性は表1のような種類がある．JOINT は RESISTOR, CONNECTION の手が他のタスクと接続していることを示している．INPORT, OUTPORT, VALUE, NONE はそれぞれ入力端子，出力端子，値 (Vdd, Vss, true, false)，未定義を表す手であることを示している．

SOURCE, GATE, DRAIN は NMOSGATE, PMOSGATE の端子の役割をもった手であることを示している．これらのうち GATE が直接に入力端子となったり，値を持つ場合には INGATE, DATAGATE という属性に変化する．

表 1 タスクと手の属性
Table 1 Relation of Task and Hand attributes.

手の属性＼タスク	NMOSGATE	PMOSGATE	RESISTOR	CONNECTION
JOINT			○	○
INPORT	○	○	○	○
OUTPORT	○	○	○	○
SOURCE	○	○		
GATE	○	○		
DRAIN	○	○		
INGATE	○	○		
DATAGATE	○	○		
VALUE	○	○	○	○
NONE			○	

○：手の属性として存在するもの

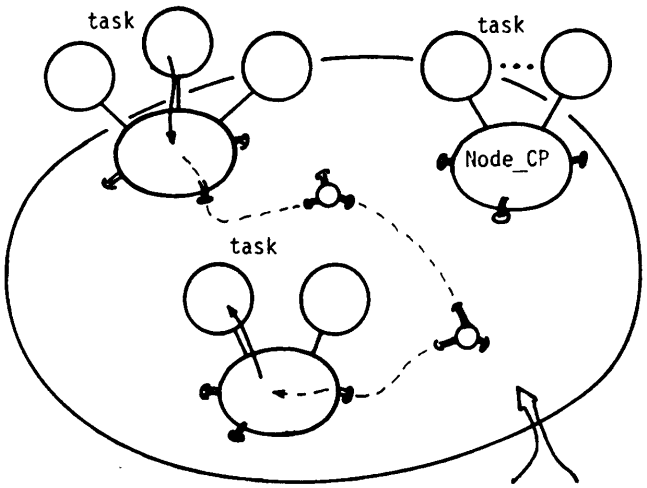

図 4 タスク間通信をサポートするネットワーク型コンピュータのモデル図
Fig. 4 A model of Network-computer properties supporting communication of tasks.

3.4 タスク間通信

タスクネットワークに表現された各機能はシミュレーションデータ(真・偽・不定)を相互に伝送することによりシミュレーションが実行される．この相互通信を“タスク間通信”と呼ぶ．タスク間通信は図 4 のモデル図に示すようにそれぞれのノードコンピュータに伝送先を指定してデータ伝送を依頼するかたちで行われる．データ伝送の経路はノードコンピュータのシミュレーションとは独立した通信管理機能によって最短経路が決定される．したがってタスクは相手の番地の情報を保持する．その番地に関する情報は相手のタスクの存在するノードコンピュータ番号，ローカルタスク番号（タスクネットワークを展開した際，各ノードコンピュータ内においてあらためて与えられる一連のタスク番号）及び手の番号である．これらの番地に関する情報はノードコンピュータにタスクを配分するにあたってコンパイラが決定する．これは従来の一般的に用いられているコンパイラにはなかった新しいコンパイラの機能といえよう．

タスク間通信は，ネットワーク型コンピュータが有するシミュレーションとは独立した機能であり，結合されたタスク間は論理的に双方向の信号線が設定されているのと同等の機能を果たしている．

3.5 シミュレーションの実行

各ノードコンピュータにおけるタスクはすべて独立しており，その処理は互いに接続しているタスク間で行う通信により実行される．タスク間通信において真・偽・不定の 3 値を送り合うことにより，タスクの機能及び手の属性からタスクのイベントを管理しシミュレーションが実行される．そのためシミュレータが必要とするタスクの情報はタスクの機能・手の属性・タスクの接続情報でありコンパイラはこれを生成する．

4. MOS 論理回路記述

並列 MOS 論理シミュレーションのための回路記述において必要な内容及び PALM の記述文法について以下に述べる．

4.1 構造記述

スイッチレベルの論理シミュレーションにおける MOS 論理回路は，MOS トランジスタを中心として構成されたスイッチネットワークとして捉えられる．そのためトランジスタ間の接続構造を忠実に記述する必要がある．

4.2 階層的な回路記述

MOS 論理回路の構造を記述する場合，一度に大きな回路を記述する代りに小さな回路（一つの機能ブロックを構成するような回路）に分割し階層的に記述することができるようにする．階層的な記述は，回路の中の繁雑なトランジスタ端子の接続を小さな回路内の接続に置き換える．また，小さな回路に分割することにより，大きな回路を機能ブロックごとの接続として記述でき，回路内の部分的な変更を容易に行うことができる．

4.3 回路のモジュール化

すでにシミュレーションを終えた回路は，それらをモジュールとして用いて新たな回路を合成することができる．モジュールは内部の機能を保存するため入出力端子を明確に定義し，定義された端子を通してのみ他の回路と接続する．そのため回路記述は互いに独立して記述する．これにより各回路記述はライブラリモジュールとして蓄積することができる．

4.4 並列処理

MOS 論理回路の並列処理を記述する場合，従来は，手により並列処理するための回路の分割・ネットワーク型コンピュータへ展開するタスクの配置等を細かく記述する方法を用いていた．ここでは大規模な MOS 論理回路のシミュレーションを行うことができるようにネットワーク型コンピュータのネットワーク

情報だけをコンパイラに与えて自動的に処理させ，またネットワーク型コンピュータの構造の変更等に柔軟に対処するためにコンパイラに処理させる方法を提案する．

4.5 制御記述

コンパイラに対してシミュレーションの母体となる回路を指示したり，回路の入出力を定義し，シミュレーションのためのデータファイルやシミュレーション結果を納めるファイルを指定する必要がある．また，ネットワーク型コンピュータのネットワーク情報もコンパイラに指示する必要がある．

そのためこれらの情報を制御記述として回路記述とあわせて与える．これにより，コンパイルからシミュレーションまでの一連の処理を容易にすることができるとともに各種の論理シミュレーション情報を一つのファイルに履歴として残すことができる．

4.6 PALM の文法

以上のような方針のもとに，まずコンパイラが扱うことのできる並列 MOS 論理シミュレーション用回路記述言語 PALM の文法を定めた．

PALM は制御記述と回路記述から構成される．**表2**は，PALM の予約語である．制御記述には，コンパイラがコンパイルを進めていく上で必要とする情報を記述する．#entry はシミュレーションを行う母体となる親回路を示し，#inport は親回路のポートの中で入力端子となるものを #outport は出力端子となるものを示している．この三つの制御記述によりシミュレーションを行う対象とその入出力関係を与えることができる．#data はシミュレーションのデータファイル名を示し #result は出力先のデータファイル名を示している．#hardmap がネットワーク型コンピュータのネットワーク情報のファイル名である．さらに #include によりコンパイル中に他の回路記述をライブラリとして読み込みコンパイルする機能を持たせている．

表 2 PALM の予約語
Table 2 Reserved words of PALM.

制御記述語	回路記述語		
#entry	定義	部品	値
#inport			
#outport	circuit	nmos	false
#data	line	pmos	true
#result	structure	resistor	Vss
#hardmap	end		Vdd
#include			

回路記述は，**図5**において定義された構文に基づき個々の回路の構造を定義する．

回路記述は circuit 文によって行う．circuit 文では回路名及び他の回路と接続するためのポートと回路の本体を定義する．また回路の中で，子回路（部品）を接続する線路を必要に応じて line 文により定義する．回路の本体は structure 文から end 文の間に定義する．

スイッチレベルの論理シミュレーションの基本素子としては nmos (N タイプの MOS トランジスタ)・pmos (P タイプの MOS トランジスタ)・resistor (抵抗) の三つがある．nmos と pmos の三つのポートの属性は左から gate, drain, source となっている．

PALM で記述された回路は，互いに独立したものとして扱われる．そのため一度に大きな回路を記述しコンパイルすることなく，小さな回路に分けて記述し分割コンパイルすることができる．各回路はポート定義されたポートのみで結合する．図5の構文からもわかるようにPALM では，回路記述の中にさらに回路記述を行うことはできない．レベルの異なる回路記述を許す利点は，共通の線路

```
<回路記述> ::= circuit <回路名> ( <ポート名の並び> );
                  <宣言部>
                structure
                      <回路の本体>
                end;

<回路名> ::= <名前>

<ポート名の並び> ::= <ポート名>| <ポート名の並び>, <ポート名>
<ポート名> ::= <名前>

<宣言部> ::= <空>| <ライン宣言>
<ライン宣言> ::= line <ライン名の並び>;
<ライン名の並び> ::= <ライン名>| <ライン名の並び>, <ライン名>
<ライン名> ::= <名前>

<回路の本体> ::= <部品の並び>
<部品の並び> ::= <部品>| <部品の並び>, <部品>
<部品> ::= <部品名> ( <実パラメタの並び> );|
           <部品名> ( <実パラメタの並び> ): <注釈>;
<部品名> ::= nmos|pmos|resistor|<回路名>
<実パラメタの並び> ::= <実パラメタ>|
                       <実パラメタの並び>, <実パラメタ>
<実パラメタ> ::= <値>| <ポート名>| <ライン名>

<値> ::= true|false|vdd|vss
```

図 5 PALM 回路記述についての文法規則
Fig. 5 Syntax of PALM (PArallel logic simulation description Language for Mos transistor circuits).

を子のレベルの回路中で使用できることであるが，PALM では各回路接続のためのインタフェースを明確にするためにポート以外の共通線路が生じるような記述はできないようにしてある．

5. PALM コンパイラ

PALM コンパイラは，スイッチネットワークとして認識される MOS 論理回路をネットワーク型コンピュータ上に展開する機能もあわせて有し PASCAL によって記述してある．

PALM においてはこの機能を，回路記述ごとの中間ファイルの作成（パス1），#entry 指定された親回路記述の解析及び部品となる回路記述の解析・接続（パス2）によってタスクネットワークを生成し，さらにネットワーク型コンピュータ上へ展開（パス3）という三つの過程によって行っている．この過程を図6に図示する．

5.1 パス1の機能

パス1においては制御記述のパラメタの取り込み，各回路記述ごとにコード変換を行い中間ファイルを作成する．

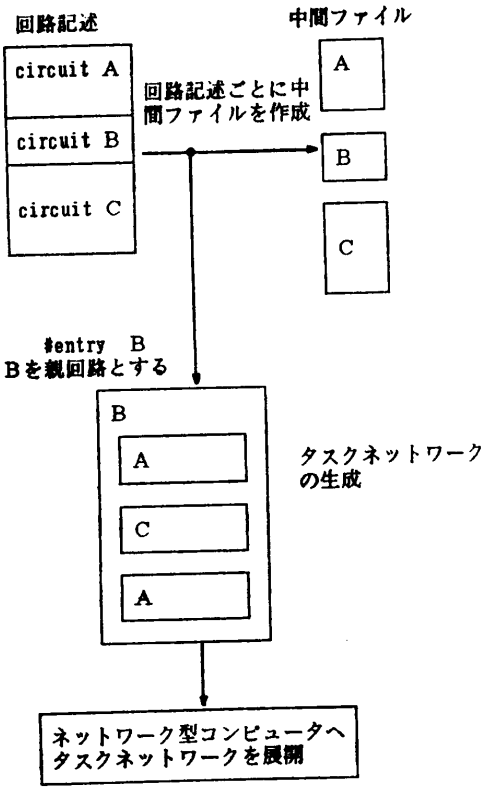

図 6 PALM コンパイラの処理手順
Fig. 6 Process of PALM compiler.

これは，メモリの節約により大きな回路記述のコンパイルを可能にすると同時にパス2においてタスクネットワークを生成する場合に必要な子回路の検索のための回路管理及び検索ルーチンを省略するという効果をもたらしコンパイラをコンパクトに設計できる．

中間ファイルの作成にあたって，表2にある予約語はコード変換を行い，回路のポート名・ライン名・子回路名はそのままファイルに書き込まれる．

5.2 パス2の機能

パス2においては #entry 指定された親回路中間ファイルをオープンし回路情報を読み込みながらタスクネットワークを生成する．同時にネットワーク接続の最適化，並びに未定義ポート名・ライン名のチェック等を行う．

PALM ではこれらの機能を，タスクスタック（タスクの情報を蓄積）・ポートスタック（ポート名及びポートの接続情報を蓄積）・ラインスタック（ライン名及びラインの接続情報を蓄積）という三つのスタックを用い図7のプログラム構成の概要に示すようにコンパクトに実現している．この処理手順を図8を例として以下に示す．

この図はコンパイルの進行過程を回路記述（パス2においては中間ファイルになっている）とスタックの状態により示したものである．パス2では，まず親回路中間ファイル adder をオープンし，port_name_set において回路のポート名 x, y, c0, s, c をポートスタックに蓄積し，ポートの接続のための準備を行う．続いてラインの定義があるので，ポートと同様にライン名 s1, c1, c2 をラインスタックに蓄積し，ラインの接続のための準備を行う．

次に構造解析部に入る．図8の例では最初に子回路 xor が読み込まれるので xor を回路名として手続き main_circuit を再帰的に呼出し，子回路中間ファイル xor を処理する．ここでは，親回路同様にポート名・ライン名を蓄積し，構造解析部に入る．続いて読み込まれるのは pmos, nmos という element (MOS トランジスタ・結線・抵抗）であるので図のようにタスクとして蓄積し，その接続情報（手の属性・接続するタスクの番号及び手の番号・値）を接続情報ポインタとして設定する．

構造解析が終ると line_task_set においてラインスタックに蓄積された接続情報（h1, h2 に接続している接続情報ポインタ）を処理しライン名 h1, h2 をラインスタックから取り去る．この際，2点接続（接続情

```
procedure Pass2_Compile(entry_name);

  procedure main_circuit(circuit_name);

    begin
      open(circuit_name);
      port_name_set;                          ポート名，ライン名の
      if line_name_exist then                 スタックへの蓄積
           line_name_set;
      while not eof(circuit_name) do          構造解析部
            case word of
              element_operation
                  : element(word);            ---部品の解析
              sub_circuit
                  : begin
                      main_circuit(word);     ---子回路記述のコンパイル
                      port_connection;        ---親回路と子回路の接続
                    end;
            end;
      line_task_set;                          回路中のラインをタスクに変換
      close(circuit_name);
    end;

begin
    main_circuit(entry_name);                 親回路記述のコンパイル
    port_task_set;                            ポートの接続情報をタスクに変換
end;
```

図 7 PALM におけるパス 2 コンパイルの概要
Fig. 7 Procedure of pass 2 compile: PALM.

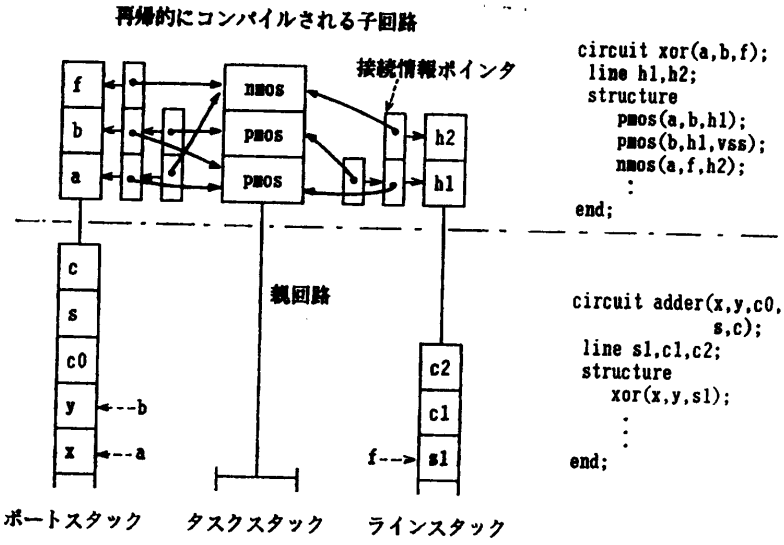

図 8 PALM のパス 2 におけるスタックの構造とその接続関係
Fig. 8 Stack structure and connection in pass 2 compile.

報ポインタが二つ）のものはタスクを設けず互いに接続し，3点接続以上のものは三つの手を持つタスク (CONNECTION) を新たに生成する．タスク間の接続を行う手続き main_circuit が終了すると，親回路のコンパイル手続き port_connection に戻る．

port_connection では，xor の各ポートに adder の x, y, s1 が接続しているので xor の a, b, f の接続情報ポインタを x, y, s1 につなぎかえ，ポートスタックからポート名 a, b, f を取り去る．

同様にして構造解析部及び line_task_set を終了する．しかし，手続き main_circuit では，他の回路との接続を行うための接続情報をポートスタックに残しているので親回路においては port_task_set において，line_task_set と同様の方法でポートの接続情報ポインタを処理し，一つのタスクネットワークを生成する．

このように再帰呼出しにより処理ができるようにして親回路中間ファイルの処理を一時中断しながら記述順序に従って子回路中間ファイルを処理する．さらに子回路との接続を行うポート情報がポートスタックに積み上げられているため子回路記述の中でポートに異なった名前を用いていてもそのポート名を変更することなく，接続情報ポインタを親回路のポートスタックもしくはラインスタックに付け替えて親回路と子回路の接続を行うことができる．

5.3 パス 3 の機能

パス 3 においては，パス 2 で得られたタスクネットワークをシミュレータを構成するノードコンピュータ上に展開し，各タスクには，配置されるノードコンピュータの番号と手によって結合されるタスクの配置先のノードコンピュータの番号を付加し，タスク番号をローカルタスク番号に変更する．この際，隣接したタスクは極力隣接したノードコンピュータに配置（ネットワーク型コンピュータとタスクネットワークとを木構造状に展開し，タスクのノードコンピュータへの配置を決定する）[2] し，中継数の増加によるタスク間通信の負荷を軽減している．

ここでタスク番号をローカルタスク番号に変更する目的は，タスク間通信における番地をダイレクトに決定し，通信によるオーバヘッドを軽減するためで

ある.

またパス 3 においては，シミュレータにおける回路のシミュレーション打切りのための情報として回路の入出力間の最大距離の計算を実行する.

以上の機能により MOS 論理回路をネットワーク型コンピュータ上に展開し，各タスクの機能とその接続情報のみを与えることによりスイッチレベルの論理シミュレーションを行うタスクネットワークを生成する.

6. コンパイル例

図 9 は nmos と pmos のトランジスタが混在した全加算器の回路である．この回路を例として PALM による記述，コンパイル結果等を以下に示す.

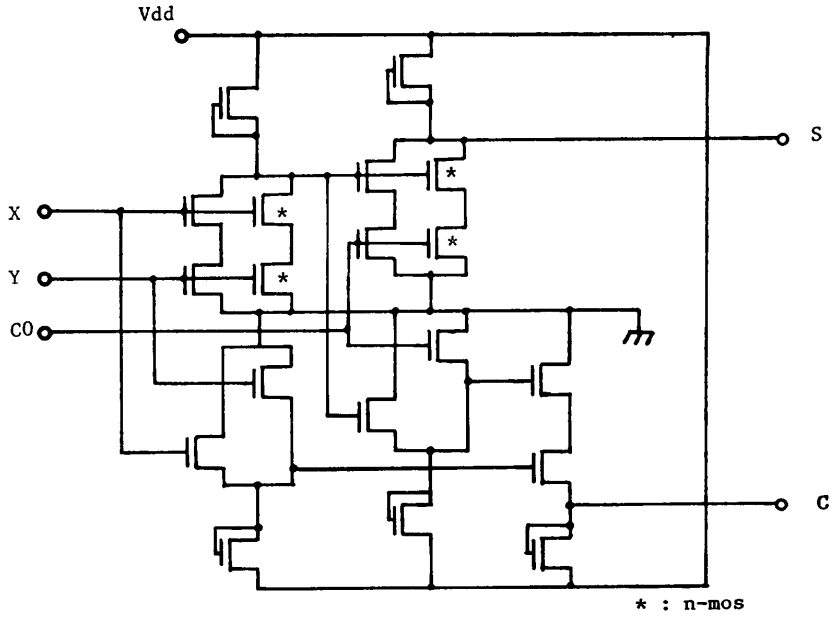

図 9 全加算器の回路例 (TFADDER)

Fig. 9 An example of Full-adder composed of NMOS and PMOS transistors (called TFADDER).

図 10 はその回路記述例である．回路記述 tfadder によって，この全加算器は二つの xor 2, 二つの and 2, そして一つの or 2 回路により構成されていることが示される．また各 xor 2, and 2, or 2 回路はその下に記述され，トランジスタと抵抗によって構成されている．このように同じレベルの回路記述を回路本体の部品の一つとして記述することにより階層的な記述を可能にしている．また回路間の接続は定義されたポートのみにより行われている.

制御記述においては，コンパイラに対して #entry によって tfadder が親回路であることを示し，#inport, #outport によって x, y, c0 が入力端子，s, c が出力端子であることを指定している．シミュレーションのデータはファイル testdata にあり，シミュレーショ

表 3 ネットワーク型コンピュータの構造連結表
(ノード数＝6)

Table 3 A connection of a Network-computer.
(6 Node computers)

ノード＼ポート	0	1	2
0	1	3	5
1	0	2	4
2	1	3	5
3	0	2	4
4	5	3	1
5	0	4	2

```
#entry TFADDER
#inport x,y,c0
#outport s,c
#data <testdata>
#result <testresult>
#hardmap <mosplus>

circuit tfadder(x,y,c0,s,c);
  line s1,c1,c2;
  structure
      xor2(x,y,s1);
      xor2(s1,c0,s);
      and2(x,y,c1);
      and2(s1,c0,c2);
      or2(c1,c2,c);
end;

circuit xor2(a,b,s);
  line h1,h2;
  structure
      resistor(Vdd,s);
      pmos(a,s,h1);
      pmos(b,h1,Vss);
      nmos(a,s,h2);
      nmos(b,h2,Vss);
end;

circuit and2(a,b,f);
  structure
      resistor(Vdd,f);
      pmos(a,f,Vss);
      pmos(b,f,Vss);
end;

circuit or2(a,b,f);
  line h;
  structure
      resistor(Vdd,f);
      pmos(a,f,h);
      pmos(b,h,Vss);
end;
```

図 10 TFADDER の PALM による記述例

Fig. 10 An example of TFADDER description by PALM.

ン結果は testresult に格納するよう指定されている．ネットワーク型コンピュータの構造情報は mosplus というファイルにあることを示している．ネットワーク型コンピュータの構造情報は表3のような連結表によって示す．

図11は，PALM コンパイラによって得られた図9の tfadder のコンパイル結果である．Depth によりこのタスクネットワークの深さを与えている．2行目以降各行ごとに一つのタスクの情報が与えられ，自らの番地（ノードコンピュータ番号，ローカルタスク番号），タスクの機能（図中に表示されている番号は1を基準として，表1のタスクに左から順番に対応），3本の手の情報（手の属性，結合しているタスクの情報（ローカルタスク番号，手の番号，ノードコンピュータ番号））が示されている．シミュレータはこれらタスクの情報を基にタスク間通信を行いながらシミュレーションを進めていくことができる．このタスクネットワークを図示した結果が図12である．図12において in 1, in 2, in 3, out 1, out 2 は図9の X, Y, C0, S, C 端子にそれぞれ対応している．シミュレータにおいては各端子，タスク等はすべて番号により識別されるために，図10の回路記述において記述された順序に従いコンパイラが番号付けを行ったものであり図11のコンパイル結果に対応している．

```
Depth =  15
 5  1 3    value  1  *  *    joint  3  1  4     none  *  *  *
 2  1 2     gate   5  1  5    drain  4  1  1   source  2  1  2
 2  2 2     gate   6  1  1    drain  1  2  2    value  0  *  *
 3  1 1     gate   5  1  4    drain  4  0  1   source  2  1  3
 3  2 1     gate   4  1  0    drain  1  2  3    value  0  *  *
 1  1 3    value  1  *  *    joint  6  2  2     none  *  *  *
 5  2 2     gate   4  1  4    drain  6  1  2   source  3  1  3
 3  3 2     gate   6  1  4    drain  2  2  5    value  0  *  *
 0  1 1     gate   5  1  1    drain  7  1  3   source  3  1  2
 2  3 1     gate   6  1  5    drain  1  2  0    value  0  *  *
                    :              :              :
 3  7 4   outport  1  *  *    joint  1  1  0    joint  6  0  2
 2  6 4    joint  7  2  3    joint  2  1  5    joint  1  1  1
 0  5 4   outport  2  *  *    joint  3  1  1    joint  5  1  3
```

ノードコンピュータ番号
結合しているタスクの手の番号
結合しているタスクのローカルタスク番号
手の属性
タスクの機能
ローカルタスク番号
ノードコンピュータ番号

図 11 TFADDER 回路記述のコンパイル結果（タスクネットワーク情報）
Fig. 11 Result of compiled TFADDER description (Information of Task network).

図 12 から PALM によって生成されたタスクネットワークには2点間の中継のみを行うようなタスクは存在せず，MOS 論理回路を構成するために必要な最小限のタスクのみから構成されていることがわかる．また，タスクと MOS 論理回路のトランジスタ・抵抗・結線が一対一に対応し，相互に結合されていることは PALM によって得られたデータはそのままレイアウト用の基礎データとして活用できることを示している．

7. あとがき

本論文ではネットワーク型コンピュータの上で並列 MOS 論理シミュレーションを行うための回路記述言語 PALM 及び PALM コンパイラを開発し，その試作を行った．

PALM は MOS 論理回路を記述するための言語であり，PALM コンパイラは記述された MOS 論理回路をスイッチレベルのシミュレーションを行うためにネットワーク型コンピュータ上に展開することを自動的に行うものである．スイッチレベルの論理シミュレーションは MOS 論理回路の設計法に新たな分野を開拓するためのツールとなる．その意味からも MOS トランジスタ数の多い回路の論理シミュレーションを高速に処理する並列 MOS 論理シミュレーション用の回路記述言語 PALM とそのコンパイラの役割は大きいといえよう．

今後の課題としては，故障の挿入記述を行ってフォールトシミュレーションを行うことができるようにすることがある．

参考文献

1) 樹下行三：論理装置の CAD，情報処理学会，オーム書店(1981).
2) 竹之上，古賀：ネットワーク型コンピュータによる MOS 論理回路シミュレーションの一方法について，情報処理学会論文誌，Vol. 26, No. 3, pp. 420-428 (1985).
3) Bryant, R. E. : A Switch-Level Model and Simulator for MOS Digital Systems, *IEEE Trans. Comput.*, Vol. C-33, No. 2, pp. 160-177 (1984).
4) Lighter, M. R. and Hachtel, G. D. : Implication Algorithms for MOS Switch Level Functional Macro Modeling Implication and Testing, 19 th Design Automation Conference,

Paper 38. 3, pp. 691-698(1982).
5) Ramachandran, V.: An Improved Switch-Level Simulator for MOS Circuits, 20th DAC, Paper 21. 3, pp. 293-299 (1983).
6) Kozak, P., Bose, A. K. and Gupta, A.: Design Aids for the Simulation of Bipolar Gate Arrays, 20th DAC, Paper 21. 2, pp. 286-292 (1983).
7) Gottlieb, A. and Schwarty, J. T.: Network and Algorithms for Very Large Scale Parallel Computation, *IEEE Comput.*, Vol. 15, No. 1, pp. 27-36 (1982).
8) Kawai, M. and Hayes, J. P.: An Experimental MOS Fault Simulation Program CSASIM, IEEE 21st DAC, Paper 2. 1, pp. 2-9 (1984).
9) 羽山，渡里，京井：MOSマスク解析における論理検証，情報処理学会設計自動化研究会資料，DA-19-6 (1983).

(昭和 60 年 10 月 24 日受付)
(昭和 61 年 4 月 17 日採録)

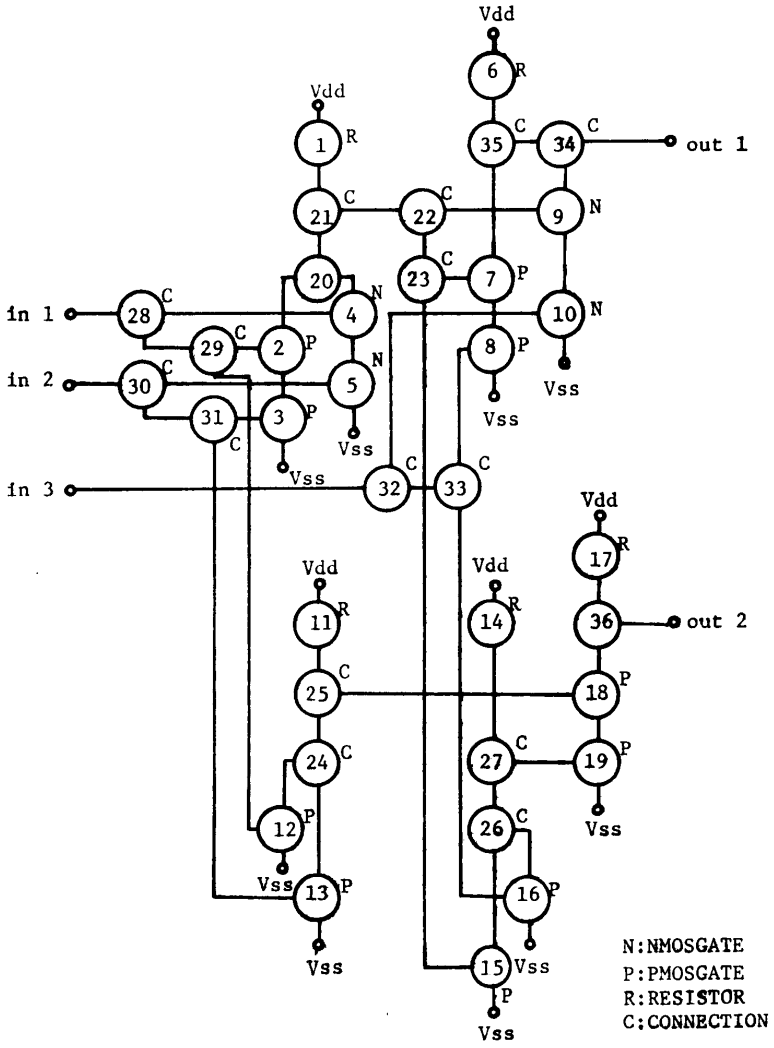

図 12 タスクネットワーク (TFADDER 回路)
Fig. 12 Task network of TFADDER.

竹之上典昭（正会員）

昭和 30 年生．昭和 53 年防衛大学校電気工学卒業．昭和 58 年同研究科電気工学卒業．同年防衛大学校陸上防衛学教室勤務．主な研究は MOS 論理回路の論理シミュレーションの並列処理方式．

古賀 義亮（正会員）

昭和 11 年生．昭和 34 年防衛大学校電気工学卒業．昭和 39 年同研究科電子工学卒業．昭和 42 年まで京都大学工学部数理工学研究生，昭和 45 年までイリノイ大学工学部コンピュータ・サイエンス客員研究員，同年防衛大学校電気工学教室勤務，昭和 52 年教授，現在に至る．工学博士．おもな研究分野は高信頼計算方式，コンピュータネットワーク等．電子通信学会，IEEE 各会員．