コンピュータネットワークによる並列論理シミュレーションの一考察†

竹之上 典昭†† 古賀 義亮†††

論理回路の高集積化が進み，論理シミュレーションの高速化が求められている．本論文では3ポートコンピュータネットワークを用いて，論理シミュレーションを並列処理する方法について述べる．並列処理を行うためのネットワーク構造，3ポートノードコンピュータの構造，並列処理記述のための回路記述言語，並列論理シミュレーションの方法について提案を行う．以上の方法を検証するために，大型コンピュータ上に3ポートコンピュータネットワークを構成し，そのなかで，並列論理シミュレーションを行っている．その結果ここで提案する方式が実際の回路を扱う場合に有用であることを明らかにする．

1. まえがき

論理装置の CAD の一環として用いられている論理シミュレーションは，VLSI の発達により回路の集積化が進み，その実行に多大な時間を費やすようになった．将来さらに回路の集積化が進めば1台のコンピュータによる論理シミュレーションは実行不可能な状態にまでなるであろうといわれている．

論理シミュレーションの高速化を図るには，1台のコンピュータの処理速度を高速化するだけにとどまらず，複数のコンピュータを使用する並列処理もあわせて開発する必要にせまられている．すでに IBM 社から，論理シミュレーションを並列処理する Yorktown Simulation Engine (YSE)[6] の報告があり，そのことからも並列処理の必要性は明らかである．

本論文では，3ポートコンピュータネットワークを用いて並列にゲートレベルの論理シミュレーションを行う方式について提案を行い，その検討を行う．その際，並列処理に適したコンピュータネットワークの構造，ノードコンピュータの構造，回路記述言語についても，新たな提案を行う．ここで提案するコンピュータネットワークは，これを構成するすべてのノードコンピュータが二つのグループに分かれて交互に動作するものである．この動作によってコンピュータネットワークにおいて並列処理を行う場合の大きな問題点の一つであるノードコンピュータ相互間のデータ伝送を円滑に行っている．ここで各ノードコンピュータは，三つの端子から受信するデータを並列処理するために，三つの CPU をもつものと考える．回路記述言語は，シミュレーションする回路をコンピュータネットワークに入力するためのものであり，並列処理を目的として新たに設計したものである．

以上の事柄について以下に説明を行い，論理シミュレーションの並列処理の可能性を明らかにするために，シミュレーションを行ったので，その結果について述べる．

2. コンピュータネットワーク

本論文で取り扱うコンピュータネットワークは，各ノードコンピュータが，3本の入出力端子をもつ3ポートコンピュータネットワークである．

図1は，3ポートコンピュータのネットワーク例であり各ノードコンピュータには，**表1**のような連結表をもたせる．また，コンピュータネットワークを用いて，論理シミュレーションを並列処理するために，論理回路の各素子の機能をノードコンピュータにタスク(仕事)として割り当てて処理する方法を用いる．この方法で並列処理するにあたって次のような条件を考える．

(1) ノードコンピュータの数はネットワークの大きさによって定まる．したがって，シミュレーションする回路の素子数が多い場合，それぞれのノードコンピュータに一時的に実行可能となるまでタスクを蓄積できるようにする．

(2) 論理回路をコンピュータネットワーク上に展開するにあたって，ネットワーク上のいずれかのノードコンピュータから論理回路の情報の入力ができるようにする．

(3) 各ノードコンピュータには，タスクをできる

† A Study for Parallel Logic Simulation in Computer Network by NORIAKI TAKENOUE (Department of Ground Defense Science, National Defense Academy) and YOSHIAKI KOGA (Department of Electrical Engineering, National Defense Academy).

†† 防衛大学校陸上防衛学教室
††† 防衛大学校電気工学教室

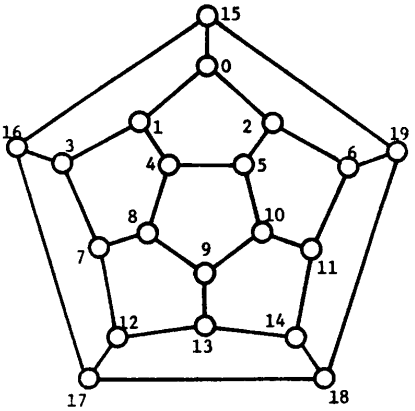

図 1 3ポートコンピュータネットワーク例
Fig. 1 A sample of 3 port computer network.

表 1 連結表
Table 1 Connection map.

ノード	0	1	2
0	1	2	15
1	3	4	0
2	0	5	6
3	7	1	16
4	1	8	5
5	4	10	2
6	2	11	19
7	3	12	8
8	7	9	4
9	8	13	10
10	9	11	5
11	10	14	6
12	7	17	13
13	12	14	9
14	13	18	11
15	16	0	19
16	17	3	15
17	16	18	12
18	17	19	14
19	18	6	15

だけ均等に配分する.

(4) 各ノードコンピュータ相互間のデータ伝送において，データの衝突を起こさないようにする.

3ポートコンピュータネットワークを用いる論理シミュレーションでは，与えられた論理回路を2分木構造に変換して展開する．そこで，まず完全2分木となる論理回路をコンピュータネットワーク上に展開してシミュレーションを行い，論理回路展開時におけるネットワークの動作，データの流れ，タスクの蓄積状況等を調べる.

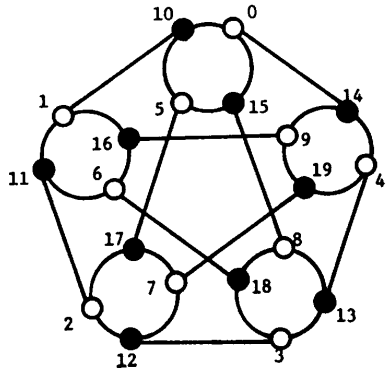

図 2 AON の一例
Fig. 2 A sample of AON (alternate operable network).

その結果は，図1のネットワークでは**表2**のようになる．この表で＊は，そのレベルでの展開動作が1回以上行われたことを示す．表内の数は，そのレベルまでにノードコンピュータに蓄積されたタスクの量を示している．表2において0番のノードコンピュータから入力された2分木は，次々と展開され，レベル7以上になるとすべてのノードコンピュータに展開されている．この状態では，すべてのノードコンピュータが動作することになるので，データ伝送に際していたるところでデータの衝突が起きる．データの衝突をさけるためには，ノードコンピュータ間の通信にプロトコル等による制御を施さなければならない．これはコンピュータネットワークを用いて並列処理を行う場合プロトコルの交換により，そのためのオーバヘッドが大きくなる.

ここで，**図2**の3ポートネットワーク（その連結表は**表3**）で同様に完全2分木を展開するシミュレーションを行うと，**表4**のような結果が得られる．この表で＊の出現に着目すると奇数レベルでは右半分に，偶数レベルでは左半分に現れている．また，蓄積量も＊が現れたノードコンピュータだけが増加している．これはノードコンピュータが二つのグループに分かれて交互に動作していることを示している.

このような交互動作を行うネットワークを AON (Alternate Operable Network) と呼ぶ．AON は偶数個のノードコンピュータから成り，図2のように二つのグループ（白ノードと黒ノード）に分割すると，互いに反対のグループのノードコンピュータと結合した連結2部グラフのコンピュータネットワークである．この特性を利用すると，二つのグループは交互にデータ伝送を行うので，データの衝突を防ぐことがで

表 2 2分木の展開
Table 2 Result of simulation for spread binary tree.

Level \ Node	N0	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	N16	N17	N18	N19
1	1	*1	*1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	*1	*1	*1	*1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	*2	*2	1	*1	*1	0	*1	*1	0	0	0	0	*1	0	0	*1
4	1	*2	*2	1	2	2	1	*2	*3	*2	*3	*2	*1	0	*1	*2	1	*1	*1	1
5	*5	2	2	*3	*3	*3	*3	*3	*4	*6	*4	*3	*3	*4	*3	2	*2	*3	*3	*2
6	5	*7	*7	*5	*5	*5	*5	*6	*6	*8	*6	*6	*8	*12	*8	*4	*6	*6	*6	*6
7	*9	*11	*11	*13	*12	*12	*13	*13	*11	*12	*11	*13	*18	*16	*18	*12	*11	*14	*14	*11
8	*25	*20	*20	*25	*24	*24	*25	*30	*25	*22	*25	*30	*27	*24	*27	*22	*27	*31	*31	*27
9	*45	*52	*52	*51	*45	*45	*51	*51	*54	*50	*54	*51	*49	*44	*49	*54	*56	*57	*57	*56
10	*109	*101	*101	*109	*103	*103	*109	*94	*96	*108	*96	*94	*98	*100	*98	*112	*108	*100	*100	*108
11	*225	*217	*217	*201	*210	*210	*201	*201	*197	*192	*197	*201	*196	*216	*196	*216	*209	*192	*192	*209
12	*433	*434	*434	*409	*418	*418	*409	*406	*411	*394	*411	*406	*413	*384	*413	*418	*393	*397	*397	*393

表 3 連結表 (AON)
Table 3 Connection map of AON.

ノード	0	1	2
0	10	15	14
1	11	16	10
2	12	17	11
3	13	18	12
4	14	19	13
5	15	10	17
6	16	11	18
7	17	12	19
8	18	13	15
9	19	14	16
10	5	0	1
11	6	1	2
12	7	2	3
13	8	3	4
14	9	4	0
15	0	5	8
16	1	6	9
17	2	7	5
18	3	8	6
19	4	9	7

きる．また適当な動作時間の管理を行えば，複雑なプロトコルを必要としない特徴がある．

図3はノードコンピュータ120個の AON 上に完全2分木を展開した場合のタスクの蓄積量を3次元表示したものである．中央を縦に走る断層面によって，AON が交互に動作していることがわかる．

ここでは，コンピュータネットワーク上での並列論理シミュレーションを円滑にしかも高速に行うために，このような特性をもつ AON を用いる．

3. ノードコンピュータの構造

AON を構成するノードコンピュータの構造について述べる．本方式によるシミュレーションでは，AON の特性上からノードコンピュータには，次のような能力が要求される．

（1） 3方向同時にデータの伝送（送信または受信）ができること．

（2） 送受信と同時並行的にデータ処理（データフロー処理に近い実行）が行えること．

（3） 配分されたタスクを一時的に蓄積できること．

この要求に答えるため次の二つのノードコンピュータの構造について検討する．

まず**図4**に一つの CPU を使用したノードコンピュータの構造を示す．このノードコンピュータは，一つの CPU と三つのバッファ，そしてローカルメモリで構成されている．すべての処理は一つの CPU によってなされる．三つのポートに対するデータの送受信のため各ポートにはバッファが必要である．3方向から同時に受信したデータの処理は一つの CPU で逐次に処理する．

次に，**図5**に三つの CPU を使用したノードコンピュータの構造を示す．各 CPU (x, y, z) は，同等の能

表 4 2分木の展開 (AON)
Table 4 Result of simulation for spread binary tree with the TNC.

Level \ Node	N0	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	N16	N17	N18	N19
1	1	0	0	0	0	0	0	0	0	0	*1	0	0	0	*1	0	0	0	0	0
2	1	*1	0	0	*1	*1	0	0	0	*1	1	0	0	0	1	0	0	0	0	0
3	1	1	0	0	1	1	0	0	0	1	1	*1	0	*1	1	*1	*2	*1	0	*2
4	*2	*2	*2	*1	*2	1	*3	*3	*2	*3	1	1	0	1	1	1	2	1	0	2
5	2	2	2	1	2	1	3	3	2	3	*3	*5	*6	*3	*5	*2	*4	*4	*6	*4
6	*7	*8	*11	*13	*8	*7	*9	*8	*7	*7	3	5	6	3	5	2	4	4	6	4
7	7	8	11	13	8	7	9	8	7	7	*16	*18	*20	*22	*12	*16	*16	*18	*17	*15
8	*31	*34	*34	*33	*33	*36	*33	*37	*41	*29	16	18	20	22	12	16	16	18	17	15
9	31	34	34	33	33	36	33	37	41	29	*69	*65	*64	*63	*69	*75	*64	*71	*73	*69
10	*152	*130	*132	*134	*135	*143	*136	*130	*129	*144	69	65	64	63	69	75	64	71	73	69

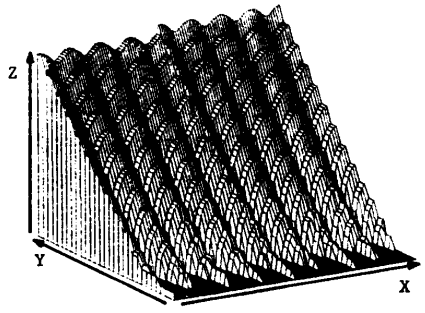

図 3 AON での2分木の展開
Fig. 3 Stored tasks in each node computer for spread binary tree on the AON
X: ノード数, Y: レベル, Z: タスクの蓄積量

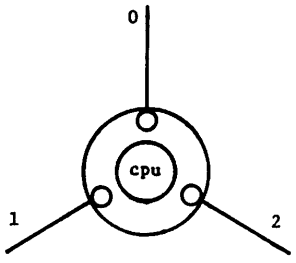

図 4 一つの CPU
Fig. 4 A node computer with 1 CPU.

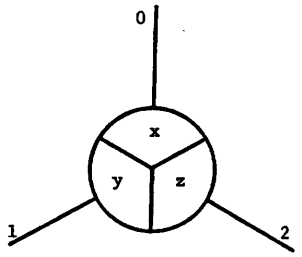

図 5 三つの CPU (TNC)
Fig. 5 A node computer with 3 CPU TNC (three port node computer).

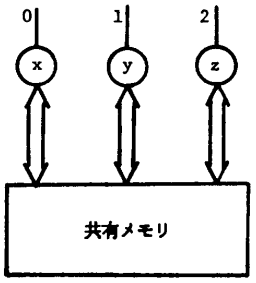

図 6 TNC(メモリとの結合)
Fig. 6 Internal expression of TNC

力をもち外部との入出力には一つのポートのみを介して行う．また，各 CPU は図 6 のように共有メモリによって結合されている．この構造では，各 CPU は，それぞれの担当ポートのみに対してデータの送受信および処理を実施することができ，共有メモリに蓄積されたデータを三つの CPU で並列に処理することができる．

ここで，図 4 の構造と図 5 の構造のいずれが並列処理に適しているかを比較する．3方向からの並列に処理されるべきデータを図 4 の構造の場合では，逐次に処理しなければならないが，図 5 の構造では，三つの

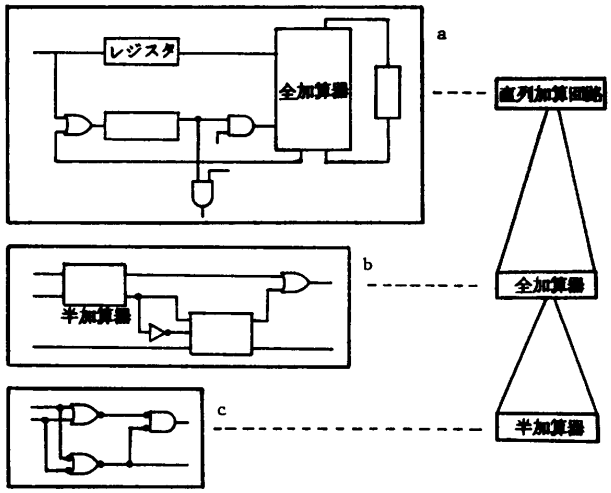

図 7 回路の階層構造
Fig. 7 Hierarchical expression of serial adder.

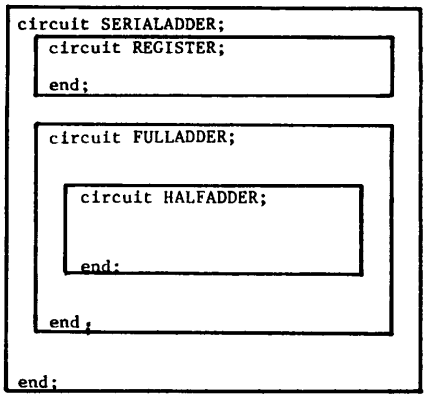

図 8 回路文の構造
Fig. 8 Description of serial adder's hierarchical structure.

CPU でメモリの同一領域を同時にアクセスしないときには並列に処理できる．以上のことからここでは図5の構造のノードコンピュータを採用し，TNC (Three port Node Computer) と呼ぶこととする．この AON 内部での TNC の結合は，ローカルネットワークなどと比べてより密にすれば，すなわち，メモリと CPU 間のデータの転送と，同程度の高速データの伝送とすれば，データフロー処理に近い効果が得られる．

4. 回路記述言語

コンピュータネットワークを用いて並列論理シミュレーションを行うためには論理回路の情報をデータとしてコンピュータに入力する必要がある．また，並列処理の実行を行うためにも新しい回路記述言語が必要となる．そこで，次の点に注目した回路記述言語を作成した．ここでは，この言語の要点のみについて述べることとする．

（1） 回路記述にあたり物理的な意味との対応を付けやすい表現にする．
（2） 並列処理の記述が容易である．
（3） 構造化言語の体系を用い階層的な記述を可能にする．

回路の記述は，PASCAL の手続きと同じ構造と役割をもつ回路文 (circuit) という手続きによって行う．

たとえば図7a のような直列加算回路を例とすると，直列加算回路は，全加算器，レジスタなどの回路で構成されている．また，直列加算回路の部品としての全加算器（図7 b) も半加算器（図7 c) やその他の素子によって構成され，半加算器自体も NAND や NOR などの論理素子によって構成されている．

このような階層構造を回路文で記述すると，図8のようになる．この図では，REGISTER や FULLADDER は，SERIALADDER の中のみで有効である．また，論理回路の物理的な意味と対応づけるための宣言文等には，以下のものがある．

入出力端子 (port)
接続端子 (terminal)
接続線 (line)
部品の関数機能 (element ... function)
端子の接続 (connection)
並列処理を指示 (parallel)

図9は 74181 として市販されている ALU（論理演算ユニット）を回路記述言語規定に従って記述した例である．var から function の終りまでが宣言部分であり，circuit SUB は，circuit ALU の中のサブシステムとしての回路である．

実行部分は begin parallel から end の間である．この parallel により，AON による並列論理シミュレーションを指示している．

5. AON での並列論理シミュレーションの実行方法

並列論理シミュレーションは，以下の二つの段階に分けて行う．

（1） 回路のネットワークへの展開

論理回路をコンピュータネットワーク上に展開する方法について図10の回路を例として説明する．これ

```
circuit ALU;
  var J;
  port  A,B,S,F:array[0-3]; M,CN,GY,CN4,PX,AB;
  line  LI1,LI2:array[0-3];
  element AND3(I1,I2,I3);
          AND4(I1,I2,I3,I4);
          AND5(I1,I2,I3,I4,I5);
          NOR2(I1,I2);
          NOR3(I1,I2,I3);
          NOR4(I1,I2,I3,I4);
  function      AND3:=I1 and I2 and I3;
                AND4:=I1 and I2 and I3 and I4;
                AND5:=I1 and I2 and I3 and I4 and I5;
                NOR2:=I1 nor I2;
                NOR3:=not(I1 or I2 or I3);
                NOR4:=not(I1 or I2 or I3 or I4);
  circuit SUB(A,B,P,Q);
          line LNOT,F1,F2,G1,G2,G3;
          begin
               LNOT:=not B;
               F1:=B and S[3] and A;
               F2:=A and S[2] and LNOT;
               G1:=LNOT and S[1];
               G2:=S[0] and B;
               G3:=pass A;
               P:=F1 nor F2
               Q:=not(G1 or G2 or G3)
          end;
  begin  parallel
          for J:=0 to 3 do  SUB(A[J],B[J],LI1[J],LI2[J]);
          GY:=NOR4(pass(LI2[3]),(LI1[3] and LI2[2]),
              AND3(LI1[3],LI1[2],LI2[1]),AND4(LI1[3],LI1[2],
              LI1[1],LI2[0]));
          CN4:=not(GY) or not(not(AND5(LI1[3],LI1[2],LI1[1],
               LI1[0],CN)));
          PX:=not(AND4(LI1[3],LI1[2],LI1[1],LI1[0]));
          F[3]:=(LI1[3] xor LI2[3]) xor NOR4(AND5(CN,LI1[0],
                LI1[1],LI1[2],M),AND4(LI1[1],LI1[2],LI2[0],M),
                AND3(LI1[2],LI2[1],M),(LI2[2] and M));
          F[2]:=(LI1[2] xor LI2[2]) xor NOR3(AND4(CN,LI1[0],
                LI1[1],M),AND3(LI1[1],LI2[0],M),(LI2[1] and
                M));
          F[1]:=(LI1[1] xor LI2[1]) xor NOR2(AND3(CN,LI1[0],
                M),(LI2[0] and M));
          F[0]:=(LI1[0] xor LI2[0] xor not(CN and M);
          AB:=AND4(F[3],F[2],F[1],F[0]);
  end;
```

図 9 ALU 記述例
Fig. 9 A sample of ALU description.

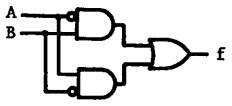

図 10 排他的論理和の回路
Fig. 10 An exclusive OR circuit.

は排他的論理和の回路である.

この回路の各素子をノードとする木として表現すると**図 11** のようになる. 図 11 は 2 分木構造になっているが, 一般の論理回路を木として表現しても 2 分木にはならないことが多い. しかし, 本方式で用いるノードコンピュータは TNC であるので, ここでは木をすべて 2 分木に変換する.

2 分木構造となった論理回路を AON 上に展開するには, 回路情報をデータとして各 TNC へ送り,

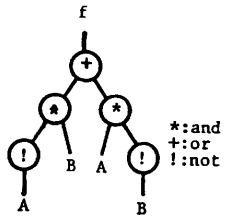

図 11 木としての表現
Fig. 11 Another expression of exclusive OR circuit.

各素子の機能をタスクとして蓄積する. そのために, この 2 分木を行きがけ順になぞって一列化を行い,

$f = +(*(!(A).\ B).\ *(A.\ !(B)))$

ただし {*: AND, +: OR, !: NOT} のような構造に変換する.

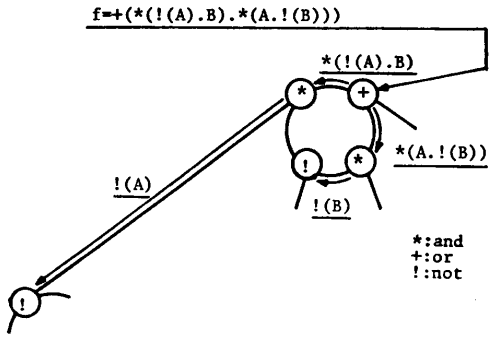

図 12 木の展開
Fig. 12 Decomposition method of exclusive OR's data on the AON.

LOC	
演算子	
トークン	DATA
トークン	DATA

図 13 タスクの構成
Fig. 13 Memory allocation of task.

各 TNC はこれを2方向へ分割しながら展開する．**図 12** に展開の様子を示す．各 TNC は先頭に出てくる演算子をタスクとして取り込み，その後に続くデータを左右に分割して次の TNC に送る．

各 TNC に取り込まれたタスクは，記憶空間のなかに**図 13** のような構成で与えられ，親のタスクのデータを指し示すポインタ (LOC) と，データを入れる箱 (DATA) をもつ．そして，各 DATA という箱にはさらにそのデータについて演算実行が可能となることを示すトークンを入れる場所がある．

(2) 実　行

AON 上へ展開された論理回路は展開の逆順，つまり木で表すと葉になるタスクから論理回路に与える値に相当するデータを流してシミュレーションを行う．

この場合，蓄積されたタスクのうち，どのタスクを実行するかという判断は，データに関して実行可能であることを示すトークンによって与え，トークンのそろったタスクから実行する．

6. AON での並列論理シミュレーションに関する検討

これまで述べてきた並列論理シミュレーションの方式について，そのアルゴリズムの検証，処理速度，ならびに実際の回路を扱う場合の問題点を明らかにするためにシミュレーションを行った．

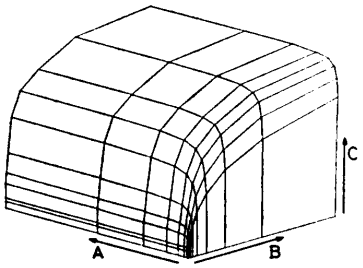

図 14 データおよびタスクの増加に対する処理速度の変化
Fig. 14 Execution speed vs. data and tasks.
A: データ数, *B*: タスク数, *C*: 処理速度〔倍〕

6.1 処理速度

本方式において論理回路はすべて2分木構造に変換される．そこで，まず完全2分木構造の論理回路を用いて本方式の処理速度について考察する．

AON の実用性を明らかにするために大型計算機の中に AON を構成し，その上に論理回路をのせてシミュレーションする方法を用いた．また，処理速度については2入力の AND, OR 等の論理素子が演算を行う時間をすべて1ゲートタイムとした．

本シミュレーションでは，図2の AON を用い，タスク数（完全2分木に展開したときの論理素子の数）を 1, 3, 7, 15, 31, 63, 127, 255, 511, 1023, データ数（論理回路の入力に与える値の組）を 1, 2, 4, 8, 16, 32, 64, 128 として実施した．

図 14 はシミュレーション結果を3次元表示したものである．この図はタスク数とデータ数を変化させた場合の処理速度（1台のコンピュータの処理速度を1とした場合）の変化の様子を表している．

この図から，シミュレーションの実効上の処理速度を速くすることができる要因が二つあることがわかる．その一つは，データ数の増加である．データ数が多いほど実効上の処理速度が速くなるのは，論理回路を AON 上に木構造として展開したため2分木構造をもったパイプラインが AON 上に構築されたような効果が得られたためである．もう一つは，タスクの増加である．これは，完全2分木のレベルを深くしたことにより並列に処理できるタスクが増加し，実効上の処理速度が速くなる．

この図からわかるように，パイプラインの効果は，データ数のわずかな増加によってすぐ飽和する．しか

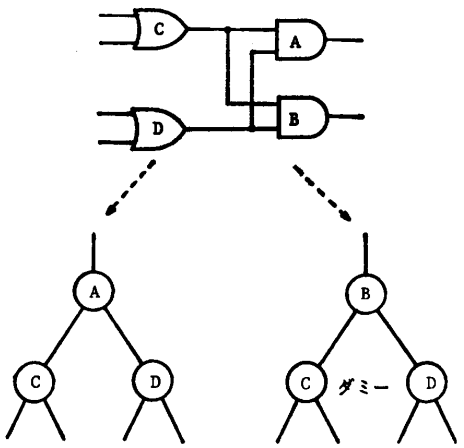

図 15 ダミーのタスクの増加
Fig. 15 Inserted dummy tasks.

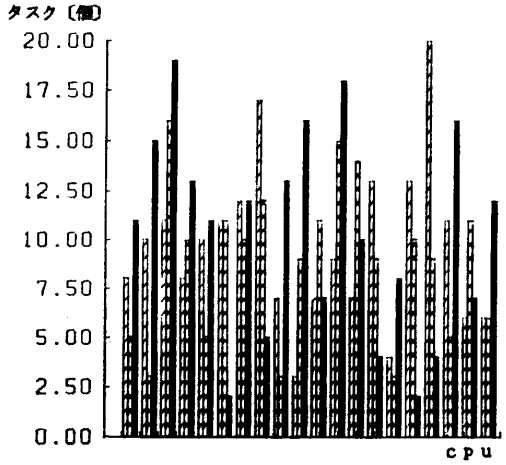

図 16 タスクの分布（一つのノードから）
Fig. 16 Distribution of task (from 1 node).

し，これは AON のノードの数を増せば改善される．タスク数の増加による処理速度の改善は，並列に処理されるタスクの増加にともなってさらに向上する可能性があることがわかる．このシミュレーションにおいて処理速度が最大となっているものは，タスク数 1,023，データ数 128 の場合の処理速度であって 22.7 倍となっている．TNC 20 個の AON についてこれだけの処理速度が得られることから，本方式による並列処理の効果があることがわかる．

6.2 実際の回路のシミュレーション

完全2分木のシミュレーションでは，よい結果が得られている．しかし，実際の回路では次のように処理速度を低下させる要因がある．

その一つは，タスクが不均等に展開されることであり，もう一つは論理回路を2分木構造に変換する際にダミーのタスクを入れる（**図 15**）ためのタスクの増加である．

これらを実際に ALU の並列論理シミュレーションを例にとって以下に説明する．

ALU には八つの出力端子がある．この八つの出力端子をそれぞれ根として2分木を構成すると八つの木ができる．ALU の八つの木をノード0のポート2の CPU（以下 $N(0,2)$ と記述する）から展開すると，**図 16** のように各 CPU にタスクが蓄積され，その総数は 575 になる．この棒グラフから各 CPU に蓄積されたタスクには，かなりのばらつきがあることがわかる．このタスクの蓄積量の平均は 9.58 で分散は 19.31 となっている．次に ALU の八つの木を $N(0,2)$ から一つ，$N(1,2)$ からまた一つと，$N(7,2)$ までの TNC

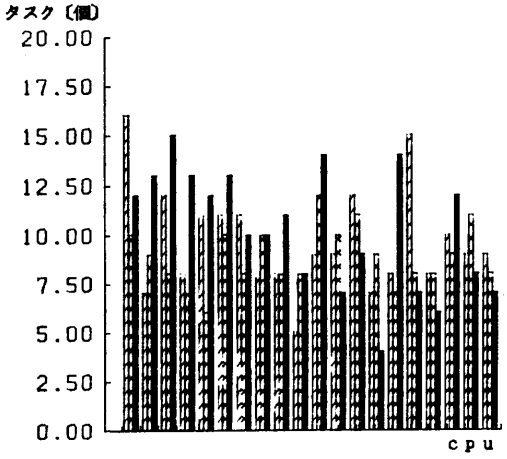

図 17 タスクの分布（八つのノードから）
Fig. 17 Distribution of task (from 8 nodes).

に分けて展開すると，**図 17** のようになる．この場合分散が 6.41 と小さくなり，AON にタスクをより均等に配分していることがわかる．

タスク配分の均等化は，そのまま処理速度に影響を与え，その処理速度は，2.6 倍から 3.2 倍へと速くなる．しかし，いずれの方法も AON に展開されたタスクの蓄積量は同じである．そこで ALU の記述（図 9）をもう一度見直してみよう．すると八つの出力端子のうち五つの端子は他の三つの端子に出力するための中間の部品からの出力端子であることがわかる．つまり，五つの端子を根とする木は，三つの端子を根とする木の部分木になっている．この特徴を利用すると，三つの木から展開して ALU のシミュレーションを行うことができる．この方法を採用するために

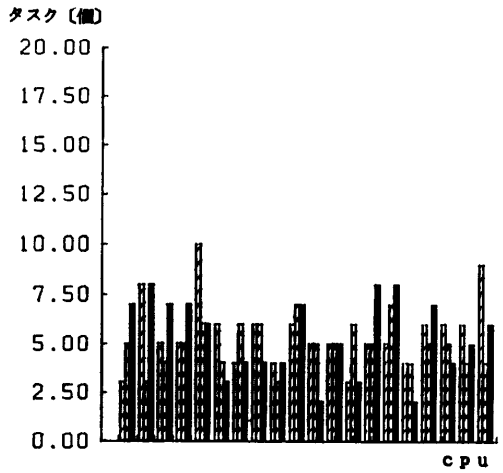

図 18 タスクの分布（三つのノードから）
Fig. 18 Distribution of task (from 3 nodes).

は，木を一列化する際に，部分木の根である演算子に出力端子記号を付加するだけでよい．出力端子記号を付加されたタスクは，シミュレーション結果を一時的に記憶しておきシミュレーション終了後，親となるノードコンピュータの呼びかけに応じて結果を回収する．図2の AON であれば，最大5回の交互動作によってすべての結果が回収できる．

次にこの方法によるシミュレーションを示す．部分木を取り除いた3の木を一つずつの TNC に分けて展開すると，**図 18** のようになり，タスクの総数も 317 に減少する．これは，八つの木を展開する場合に比べて，約 45% の減少であり，処理速度も 5.6 倍とかなり速くなっている．**図 19** は，シミュレーション時の CPU のタスク処理情況を描いたものである．処理されたタスクの量が各 TNC ごとほぼ一定の状態で処理されていることから，データが次から次へと流れ，処理が滞ることなく並列に進み，最終段階において急速にタスクの処理が減少して終了する様子がよくわかる．

以上，シミュレーション結果をもとに明らかになったことをまとめると，次のようになる．

（1） AON 上には，タスクがほぼ均等になるように，展開させる必要がある．

（2） 論理シミュレーションのデータが多いほうが処理速度が速くなる（パイプライン処理に近くなっている．）

（3） 処理速度は，1台のコンピュータによる逐次処理に比し ALU では 5.6 倍の処理速度である．

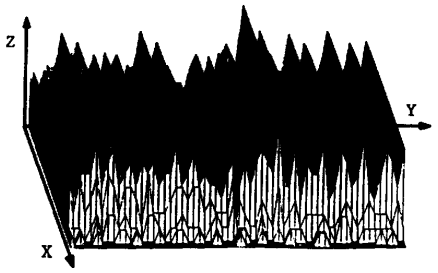

図 19 ALU シミュレーション時の CPU の処理状況
Fig. 19 Execute task distribution in a parallel logic simulation for ALU.
X: 時間，Y: CPU の数，Z: 処理されたタスクの数

（4） シミュレーション実行時に，タスクのデータに待ち行列が生じる．これは，データフローによる処理を行っているので，回路中の遅延がそのままデータの処理待ちとなるためである．しかし，ALU のシミュレーションの場合にはタスクの最大待ち行列は2であり，シミュレーションの実行には支障がない．

以上の結果から，本方式による並列論理シミュレーションが有限個のノードコンピュータ（TNC）で構成されるコンピュータネットワークにおいて，高速に処理されることが明らかになった．

7. まとめ

本報告では，有限個のノードで構成される3ポートコンピュータネットワークによる並列論理シミュレーションについて新たな提案を行った．

この方式を用いることにより有限個の TNC をもつ AON において，論理素子の多い回路をネットワーク上にたたみ込むように展開することによって並列論理シミュレーションが行えることが明らかになった．

ここで提案した AON は，ノードコンピュータ間のデータ伝送に衝突がないことから，論理シミュレーションのみならず他の並列処理の分野にも適用可能であると思われる．

参考文献

1) 古賀，児嶋：3ポート論理素子，電子通信，*Trans. IECE*, Vol. 61-D, No. 6, pp. 373-380 (1978).

2) 鈴木，古賀：3方路回線網の故障診断，電子通信，*Trans. IECE*, Vol. 61-B, No. 7, pp. 585-

592 (1978).
3) 高橋義造：並列処理のためのプロセッサ結合方式，情報処理，Vol. 23, No. 3, pp. 201-209 (1982).
4) 駒宮安男：コンピュータ基礎論，pp. 54-62，昭晃堂，東京 (1975).
5) Knuth, D. E.: The Art of Computer Programming (米田訳：基本算法/情報構造，Vol. 2，サイエンス社，東京 (1981)).
6) Pfister, G. F.: The Yorktown Simulation Engine: Introduction, 19th Design Automation Conf. Paper 7.1, pp. 51-54 (1982).
7) Lightner, M. R. and Hachtel, G. D.: Implication Algorithms for MOS Switch Level Functional Macromodeling Implication and Testing, 19th Design Automation Conf. Paper 38.3, pp. 691-698 (1982).

(昭和58年6月27日受付)
(昭和58年10月11日採録)