論理構成のシミュレーション・プログラム*

高島堅助** 津田宏明** 加藤満左夫**
戸田巌** 中村彰** 高山由***

要　約

新しい計算機または電子交換機などの論理設計が予想どおりの機能を果すかどうかを計算機でシミュレートして検査する方法は，設計の時間と人手を節約するのに極めて有効である．本文には同期式の AND-OR 論理素子を使用して設計された任意の論理回路をシミュレートするために作製したプログラム・システム LSS (Logic Simulator System) について述べる．

このプログラム・システムでは，回路図をまず素子ごとの AND-OR の入力表の形式にさん孔したものと，各素子の初期条件，各周期における外部端子条件をもとにして，指定された素子群の任意クロック数のタイム・チャートを作製することができるようにした．

計算機としては，240 語の磁心記憶装置，10,000語の磁気ドラムを持つ NEAC 2203 を使用し，約数百個までの AND-OR素子からなる論理回路をシミュレートできる．このプログラムを新しい論理設計の検査に使用し，満足な結果が得られた．プログラムはコンパイラ約 1,000語，制御用サブ・ルーチン約 1,300 語である．また磁気テープ装置を附加して，数千個以上の AND-OR 素子からなる装置をシミュレートする方法について述べた．

1. まえがき

新しい計算機，電子交換機またはその他の論理回路の設計・製作においては，論理設計自体の検査，組立配線完了後の論理の検査と手直しには非常に多くの人手と時間を要する．したがって論理設計のあらゆる面に電子計算機を応用することは極めて有効であり，従来幾つか報告されている[1)-7)]．論理設計のあらゆる面に統一的に計算機を応用することは理想的ではあるが，このためには IBM 704, 7090 のような大形の計算機を使用するのが普通のようである．筆者らが行なっている新しい計算機の設計において使用できる計算機は上記のものに較べれば小形・低速の NEAC 2203 であり，プログラム作製人員3名，期間約3カ月という制約があったので，設計された論理回路が予想通り働くかどうか計算機でシミュレートして，タイム・チャートを画かせて検査するという面だけに計算機の応用を限定した．このため作製したプログラムを LSS と呼び，現在使用中のものは LSSII であり，予想どおりの結果が得られた．

ここに述べる LSS の特徴は，1）シミュレーションをできるだけ高速化するため，コンパイラ形式と加減算による AND 回路のシミュレーションを行なったこと．2）一般に設計者が LSS を容易に使用し得るように制御用サブ・ルーチンを構成し，これを約25種のマクロ命令系で駆使し得るようにしたことの2点である．

2. 論理のシミュレーションの方法

論理回路に使用されている基本回路素子の各々に名前をつけて S_i で表わし，S_i の論理動作は論理関数

$$F_i=F_i(\cdots\cdots S_j\cdots\cdots)$$

により表わすことにする．全素子は1クロックに1段の動作を行なうと仮定し，クロック n における S_i の出力の状態を $S_{i,n}$ で表わすと，

$$S_{i,n}=F_i(\cdots\cdots S_{j,n-1}\cdots\cdots)\quad(S_k, m=0, 1)$$

で与えられる．ただし $S_{j,n-1}$ は素子 S_i に結合されている素子 S_j の $n-1$ クロックにおける状態である．

全素子の n クロックにおける状態を計算機でシミュレーションするには計算機の記憶装置に F, S, S′, A, L の五つの部分を設け，**第1図**(a) のように記憶しておく．

ただし，

F: F_1, F_2,……F_k.

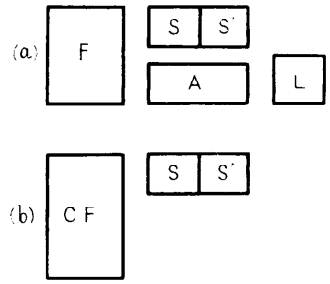
S: $S_{1,n}$, $S_{2,n}$,……$S_{k,n}$

S′: $S_{1,n-1}$, $S_{2,n-1}$……$S_{k,n-1}$

A: (S_1, a), (S_2, a_2)……(S_k, a_k)

* A Universal Program System Simulating Logic, by Kensuke Takashima, Hiroaki Tsuda, Masao Kato, Iwao Toda, Akira Nakamura (Electric Communication Laboratory) and Yoshi Takayama (Nippon Electric Co. Ltd., Tokyo)

** 電気通信研究所　*** 日本電気株式会社

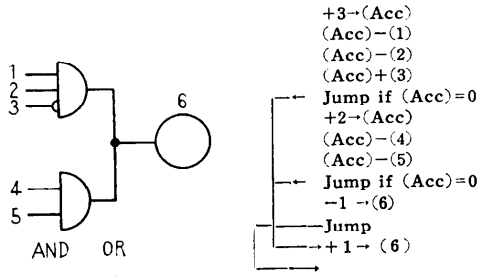


第1図　(a) 解釈形式　(b) コンパイラ形式

F, S, S′ の内容は前述のとおり，Aは素子名 S_j から $S_{j,n}$ または $S_{j,n-1}$ の記憶されている番地 a_j を引く表である．Lは F_j を読み，次に F_j 中の S_m とA中の a_m から $S_{m,n-1}$ をとり出して，F_j の値 $S_{j,n}$ を計算して，S_j とAから a_j を引き，$S_{j,n}$ を記憶するという操作を全 S_j について実行するルーチンである．1クロックごとにSとS′を交替させて（Lにおける番地変更用インデックス・レジスタの内容の交換による）繰り返えせば，任意クロック数のシミュレーションが実行できる．

この方法において，解釈ルーチン式であること，Sを読み書きするごとにAを引く必要のあることの2点を改良すれば高速化が図られる．そこで，あらかじめFとAを使って全素子のシミュレーションを行なうプログラムを作製（compile）し，これをC.Fとして，**第1図**(b)のようにS, S′に対してC.Fを天下り的に実行させれば良い．この際 C.F の長さは長くなるが，磁気ドラムか磁気テープに入れておいて，高速記憶装置に少しずつ読み出して実行すればよい．このように予めシミュレーション・プログラムを編集しておくやり方を**コムパイラ形式によるシミュレーション**と呼ぶことにした．

ここで使用する基本回路素子は AND-OR 回路であるが，これを 10 進法計算機でシミュレーションするのに加減算法を用いた．S, S′ の中で，一つの素子の状態を記憶するのに1語を用いて，論理値1,0を，それぞれ数値 +1，−1で表わすようにし，これを状態値と呼ぶ．**第2図**の実例で説明すると，まず AND ゲートの入力個数3を累算器に入れ，各入力の結合が正ならば，入力素子の状態値を引き，否定ならば加える．全入力について加減算して，結果が0のときのみ，その素子の状態値を＋1とし，それ以外の場合は−1にする*．この方法によると，命令数は（全AND入力数)+(全素子数)+α であり，プログラムの長さ，実行時間はかなり短くなる．



第2図　加減算による AND 回路のシミュレーション

コンパイラ形式と加減算法を用いた場合と解釈ルーチン，入力の1，0を逐次判定する方法を用いた場合とを比較すると，2203 の場合命令のステップ数は約 1/20 程度になる．

3. LSS II の構成

3・1 LSS II の概要

LSS II は，最初に行なった LSS I を使用した経験により多少の変更，改良を加えたプログラムである．このプログラムによってシミュレーションできるのは基本回路素子として AND-OR を使用した論理回路であれば良い．最大素子個数はコムパイルされたプログラムが9,000語を越えてはならないことから制限され，経験上，素子の個数は数百個程度が限度である．

LSS II は論理入力表コンパイラと制御プログラムの2部からなっている．

シミュレートしようとする論理回路の論理配線図または論理式から，論理入力表をテレ・タイプライタで紙テープにさん孔する．この表は各素子についてAND 入力として結合されている他の素子の名前を並べた表である．この論理入力表を論理入力表コンパイラにより読み取り，シミュレーション・プログラムを作製し，計算機の外部ドラムに入れる．制御プログラムは初期条件設定，各クロックの外部端子条件設定，シミュレーション・プログラムの実行，タイムチャートの印刷，条件付飛び越しなど約 20 数種のサブ・ルーチンよりなっている．各サブ・ルーチンには Symbolic のマクロ命令が対応させてあり，使用者はこのマクロ命令を並べて自分の望むようにプログラムを組むことができる．

---

* AND の入力数を累算器に入れる代りに0を入れ，（累算器）≦0の判定をすれば多数決論理のシミュレーションができる．

3・2 論理入力表コンパイラ

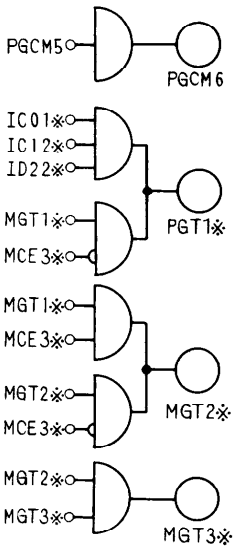
LSS Ⅱ でシミュレーションされる論理回路の構造は基本回路として AND-OR 回路を用いてあれば良い．シミュレーションは AND-OR 1段の論理を持つダイナミック型の論理の進行と全く同様に，1クロックごとの全素子の状態を計算することにより行なわれる．もし，基本回路が1段の AND-OR 論理でなく AND-OR の多段の論理を含む場合は，2段目以上は1段のものに追加されたものとして，各 AND-OR 論理ごとに名前を設けて記憶装置に状態を割り当てておき，追加されたものについては，n クロックにおける各入力の状態値から計算した状態値をその素子の n クロックの状態値として記憶し，またシミュレーションプログラムを，前段に追加された AND-OR 論理が，それより後段のものより先に実行されるようにコンパイルしておけばよい．

LSSⅡ によりシミュレーションする論理回路に対しては，上記各 AND-OR 素子および外部からの入力端子に対し，すべて5文字からなる名前をつけ，論理入力表紙テープを作製しておく．この際，多段の AND-OR 基本回路を含む場合には上記の理由で，段間に追加されたと見なされる AND-OR 素子は各段ごとのグループに分けて論理入力表を作製する必要がある．

コンパイラの本質的な機能は，このようにして作られた論理入力表から各素子の論理をまねするプログラムを作製し，外部ドラムに逐次格納することである．論理入力表は**第3図**に示す形式で書かれる．これは**第4図**の回路を論理入力表にしたものである．

```
   FIG NO. MG 1-6※1-EX    DATE 37 09 07,*
## PGCM 6,
01 01 PGCM 5,
## MGT 1*,
01 03 IC01※ IC12※ ID22※,
02 02 MGT 1※ -MCE 1※,
## MGT2※,
01 02 MGT 1※ MCE 3※,
02 02 MGT 2※ -MCE 3※,
## MGT 3※,
01 02 MGT 2※ MGT 5※,
```

第3図　論 理 入 力 表



第4図　論理回路図の一例

コンパイラは，このような表の全素子の分についてまず ## の付いた素子の名称のみを読み取り，分類して素子分類表を作る．いま，内部記憶ドラムの0～999番地を WD 1，1000～1999 番地を WD 2 とする．分類された順番の番号が WD 1 におけるその素子の状態を記憶する番地になる．コンパイルに際しては，この表を WD 2 に記憶しておき，もう一度入力表テープを読み，加減算法による各素子のシミュレーション・プログラムを作製して行く．この時の加算，減算，store の命令の番地部は，前に作った分類表を使って引く．なお，これらの命令には index 指令を設けて，WD 1 または WD 2 の中，加減命令に対しては index 2，store 命令に対しては index 1 の指定を受ける（AND-OR を1クロックの間に多段に付ける場合，前に附加した AND-OR 素子の場合は，加減，store 命令共に index 1 を指定しておく）．**第5図**は**第3図**の表をコンパイルした結果である．

以上が基本的な仕事であるが，一つの問題がある．

```
000  060
380001  214098  429003  390001  439004  100001  380001  100000  112099  380003
214019  214020  214021  429010  380002  214076  204034  429010  390001  439011
380001  100000  112076  380002  214076  214034  429017  380002  214080  204034
429017  390001  439018  100001  380001  100000  112080  380002  214080  214034
429024  380002  214081  204034  429024  390
```

第5図　コンパイルされたシミュレーションプログラム

これらのプログラムを外部ドラムから100語ずつ磁心記憶装置の一部 C_1 に読み込んで実行するのに，100語ごとの切れ目が1個の素子のルーチンの中途に入った場合の処理が必要である．

このため，100語が終った時，磁心記憶装置の一部 C_2 のルーチンに帰る時，C_2 の0番地または1番地に飛び込んだ時には，次の100語を C_1 に読み込んで再び C_1 に行く時，C_1 の0番地または1番地に飛ぶようにした．

リンク設定ルーチン

コンパイルは適当な素子の群を1ブロックとして，数個のブロックに分けて行なうことができ，後から各ブロックの最後にリンク設定用のパラメータを設定し，全体がつながるようにした．これは多段の AND-OR 論理を使用する場合，一群の素子の論理を訂正したい時，前に作ったルーチンの後に訂正したルーチンをつないだりする場合に必要である．論理入力表のレーベルから各ブロックについて，一段の本来の素子ならば，DEL，段間に追加する素子ならば，NOD の種別と訂正ルーチンの訂正番号 CORNO·*n*，このブロックの外部ドラムにおける開始ブロック番地，およびその最終番地の四つのパラメータを含む語を各ブロックごとに作り，これら全部を大きさの順に分類し，その順序で各ブロックの最終番地に次のブロックの外部ドラムにおける開始ブロック番地を書き込んで行く．最後のブロックの最終番地は0を書き込んでおき，これにより全体の終了を判定する．**第6図**にこのルーチンで設定した場合に生ずる結果の確認用出力の一例を示す．

```
DEL  COR  N. 00   B 030   ALP 0819
NOD  COR  N. 00   B 090   ALP 1108
```

第6図　リング設定ルーチンの出力

各種の誤りの検出について

論理入力表テープをコンパイルする際，最初，表の形式上の誤りを検出するようにプログラムしておいたところ，書き写し時およびさん孔時の誤りが多く，すこしコンパイルすると誤りでテープ修正，かけ直しといつまでたってもコンパイルができない結果になった．このため，この点を改良し，分類表を作る最初の Run の時に，入力表示テープを再生しながら，誤りが発見されれば，誤った原因，場所を印字し，タイプライタの鍵盤から再生テープ上に正しいデータを打つようにした．こうすれば，1回で Format 上のミスを全部無くしてのち，コンパイルを実行できる．ただし，再さん孔を行なうため，最初のテープの読み込みに要する時間が長くなることは止むを得ない．

論理回路の論理入力表テープを作製する際の諸規約を**第1表**に示す．

入力表示テープの第1回目の読み取りの際に監視され，処理される誤りを**第2表**に示す．

第1表　論理入力表の作製規約

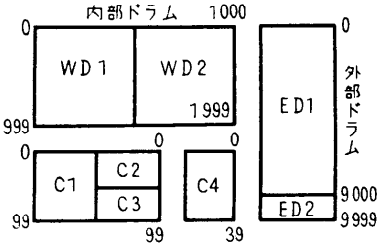
(a)	レーベルの終り，を打つ．
(b)	素子の名前の前には ## を打つ．
(c)	素子の名前のあとに，を打つ．
(d)	AND ゲートの番号を2桁の数字で打つ．
(e)	AND ゲートに入る入力個数を2桁の数字で打つ．
(f)	ゲート番号の次に余白を打つ．
(g)	素子の AND ゲートの個数の次に余白を打つ．
(h)	素子の名前は5桁の文字または数字で表わす．
(i)	AND ゲートの入力となる素子の名前の間には余白を打つ．最後の名前の次には，を打つ．
(j)	ゲートの入力が否定の場合は，その入力素子の名前の前に-を打つを打つ．
(k)	，た後は必ず復改を打つ．
(l)	Tape end mark として @ を打つ．この時は，を打ってはならない．
(m)	Block end mark として/をさん孔する．，を打ってはならない．
(n)	入力表の最後には//を打つ．
(p)	多段のゲートの場合，追加された AND-OR 素子のみについてまとめた入力表を作り，その Label の，のあとに NOD を打つ．

第2表　論理入力表の誤まり検査項目

(i)	使用記号の誤まりと脱落
1.	レーベルの最後の，を落としたとき，または他の記号を用いた場合
2.	素子の識別 Mark「## 余白」を誤まった場合
3.	素子の End mark，を誤まった場合
4.	ゲート番号とゲート個数の間，またはゲート個数とゲートの入力素子の名前の間に余白を打たなかった場合
5.	ゲートに入る素子の名前の End mark の誤まり
6.	ブロック End mark の誤まり
(ii)	ゲートの入力個数を示す数字と，入力の素子の個数が合わない場合
(iii)	同一素子に対する論理入力表が2回以上表われた場合
(iv)	##□□で示されていない素子の名前がゲートの入力として表われた場合

3·3　制御用プログラム

制御用プログラムは，シミュレーションを行なって必要なタイム・チャートを印刷させる間に要する20数種のサブルーチンと，サブ・ルーチンを指定するためのマクロ命令で書かれた作業プログラムを読み込むアセンブラの二つからなる．作業プログラムが実行される場合の 2203 の各種記憶装置に対する機能の割当ては**第7図**のとおりである．



第7図 記憶装置の機能割当て（LSSⅡ）

WD1, WD2; 内部磁気ドラム2,000語を二つに分け，1,000個までの素子のnクロック目の状態と$n+1$クロックの状態をそれぞれ記憶する．

ED1; 10,000語の外部磁気ドラムのうち9,000語で，コンパイルされたプログラムおよびシミュレーションに必要なグループ表，特殊の（使用者が作った）サブ・ルーチンなどの記憶に使用する．

ED2; 制御用サブ・ルーチンを格納する場所で，1,000語を使用する．

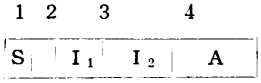
C4; 2203 の core Ⅲ と称するもので，ここに pseud code よりなる作業プログラムが記憶される．この中の38，39はパラメータに使用される．

C3; シミュレーションの実行中の主制御ルーチンがここにおかれ，C4 の命令を逐次読み，これを解読してインデックス・レジスタをセットし，ED2 から指定されたルーチンC2 に読み出し，C2 のルーチンに飛ぶ．終れば再び，ここのルーチンにもどる．

C2; 制御用サブ・ルーチンは ED2 からここに読み込まれ実行される．

C1; 制御用サブ・ルーチンの working memory として使用される．たとえば，シミュレーションのサブ・ルーチンが，C2 に入ると，ED_1からコンパイルされたルーチンが100語ずつここに読み込まれ実行される．

作業プログラムをマクロ命令で書いてさん孔テープを作り，アセンブラで読ませると，pseud code に変換され，C_4 の中（場合によっては ED1）に格納される．一つの pseud code は1語で表わされ，次の形を取る．



S=0 のとき

I_1, I_2 はそれぞれ index 1 および index 2 に入れるべきパラメータであり，Aは読み出すべきサブ・ルーチンの開始位置の語の外部ドラムにおける番地である（主制御ルーチンは，ED_2 内の A を含む50語のブロックを C_2 に読み込み，A から開始する）．

S=1 のとき

$I_1=I_2=0$ ならば，pseud code の飛び越し命令で，次の pseud code としてA番地の内容がとられる．

$I_1=n$ ならば，繰返し命令で，Aからこの pseud code までを n 回繰り返えす．

$I_1=0$, $I_2=999$ ならば，一時停止の命令で restart ボタンにより，次の pseud code がとられる．

主制御ルーチン

これは外部ドラムに格納されているが，制御プログラムがアセムブルされて，実行開始される直前に C3 に読み込まれ，C4 の0番から，逐次 pseud code を読み出して，指定されたサブ・ルーチンを ED_2 から C2 に読み出し，実行開始させる．

制御用マクロ命令とアセンブラ

制御プログラムは，一般の使用に容易かつ便利なように，マクロ命令を Symbolic code で書けば良いようになっている．アセンブラは約360語で，Symbolic code を pseud code に直し，格納する．

Symbolic code で program を書く際には，次の四つの Directive が使用される．

nZc この符号は以下※※※※までの命令をC4のn番地以下に格納する．

nZDm この符号以下※※※※までの命令を外部ドラムの m 0番地から40語のブロックの n 番目以降に格納する．

※※※※ 一つのプログラムの終り．

mZE C4 の m 番地の pseud code からサブ・ルーチンを実行開始する．

各サブ・ルーチンの機能とそれに対する Symbolic code の詳細は附録に示す．

初期条件をセットし，1クロックごとにあるグループの素子の状態（主として外部入力端子）をセットして，指定したグループのタイム・チャートを印刷させる仕事は，マクロ命令を並べた作業プログラムで実行できるが，大体の手順は次のとおりである．

（1） 外部条件として，指定された素子のタイム・チャートをさん孔した表を読ませて，素子グループの

表* と，グループ状態表の二つにしてさん孔させる．（附録 CVIT）

1クロックごとの素子グループの状態表に変換する．（後述の CVIT）

たとえば，

```
850（グループ表の番号）
ABBDX    10111,
XYZAB    10110@
890
BACDY    10110@
000（終端記号）
```

は，

```
850  ABBDX XYZAB,（グループ表 850）
890  BACDY,       （グループ表 890）
850  11,          （状態表   850）
890  1,           （状態表   890）
850  00,          （状態表   850）
890  1,           （状態表   890）
```

の形に変換された紙テープになる．

（2） 上記の 850，890 などの素子グループ表テープを ED 1 の中に読み込む．（RTAB）

（3） WD 1，WD 2 の各語を−1（"0"）にセットする．（CLWD）

（4） 素子グループ表の番地に素子状態テープに従って±1を入れる．（TSEW）

（5） シミュレーション・プログラムを実行する．（SIML）

（6） シミュレーションの結果の全素子の状態を紙テープに1，0でさん孔する．（PALS）

（7） （4）以下を n 回繰り返えす．（REPT）

（8） 停止（HALL）

（9） さん孔したテープを読み込む．（RALS）

（10） 印刷用グループ表を読み込む．（RTABを所要回数）

（11） グループ表に従って指定した素子群のタイム・チャートをさん孔する．（PTCT 高速さん孔機でさん孔し，off-line で印刷する）

以上は大体の手順であるが，この他に，WD の指定した番地が1，0のいずれかならば，マクロ命令を一つ飛ばす．（ACDR），グループ表で指定した素子の WD における状態がグループ表の符号部分と全部一致していたらマクロ命令を飛ばす（CPSK），その他がある．

マクロ命令を使用して作った作業プログラムの一例を**第8図**に示す．この時の外部条件を設定するための

```
    ZC
HBRK.
CVIT 2Y.
HALT.
RSLT.
HALT.
RTAB.
REPT 9Y,-1X.
HALT.
RTAB.
CLWD.
RSWD.
SIML 1Y, 30Y.
PALS.
RSWD.
SIML 1Y, 30Y.
PALS.
REPT 9Y, -2X.
HALT.
RSLT.
HBRK.
RNSS 10Y.
RNSS 10Y.
PTCT 10Y, 897Y.
PTCT 10Y, 895Y.
PTCT 10Y, 893Y.
PTCT 10Y, 892Y.
PTCT 10Y, 891Y.
PTCT 10Y, 888Y.
PTCT 10Y, 887Y.
PTCT 10Y, 885Y.
PTCT 10Y, 883Y.
PTCT 10Y, 882Y.
HALT.
REPT 2Y, -12X.
JUMP Y.
※※※※
ZE 9
```

第8図　作業プログラムの例

タイム・チャートを**第9図**に示す．この制御プログラムにより作製されたタイム・チャートの一例を**第10図**に示す．

3・4 LSS II の応用結果について

これまでに LSS を応用した実例を**第3表**に示す．1は LSS I の試験用のサンプルとして使用したもの

* これは素子の WD 1 における素子の番地と土の符号を持つ語の集りで，グループ表の番号とは，この表が外部ドラムに格納された時の外部ドラムにおける表の開始ブロック番地（10,000 語を 10 語の 1,000 でブロックとした時である）．

```
885
IC01※  1,
IC12※  1,
ID22※  11,
J101※  11,
J102※  0,
J104※  11,
MGEQ※  0,
SPMC※  0,
P140※  0,
P051※  11,
P071※  11,
J051※  11,
SETFD  0,
MC07※  11,
MC08※  0,
MC09※  11@
000
```

第9図　外部条件のタイム・チャート

```
895
MR 31 ※  0 0 0 0 0 1 0 1 0 0
MR 32 ※  0 0 0 0 0 0 0 0 0 0
MCE3※  0 0 0 0 0 1 0 1 0 0
MW 31 ※  0 0 0 0 0 0 0 1 0 0
MW 32 ※  0 0 0 0 0 0 0 0 0 0
AWG2※  0 0 0 0 0 0 0 0 0 0
AWG3※  0 0 0 0 0 0 0 0 0 0
AWG4※  0 0 0 0 1 1 1 1 0 0
AWG6※  0 0 0 0 0 0 1 1 0 0
AWG7※  0 0 0 0 1 1 1 1 0 0
AWG5※  0 0 0 0 1 1 0 0 0 0
AWG10  0 0 0 0 0 0 0 0 0 1
AWG8※  0 0 0 0 0 0 0 0 0 0
AWG9※  0 0 0 0 1 1 1 1 0 1
DISG※  0 0 0 0 1 0 0 1 0 0
893
IG3※※  0 0 0 0 0 0 0 0 0 0
PGCM4  1 0 0 0 0 0 1 0 0 0
PGCM6  0 0 0 0 0 0 0 0 0 0
AWG3※  0 0 0 0 0 0 0 0 0 0
MGC1※  0 0 0 0 0 0 0 0 0 0
MGC2※  0 0 0 0 0 0 0 0 0 0
MGC3※  0 0 0 0 0 0 0 0 0 0
MGC51  0 0 0 0 0 0 0 0 0 0
MGC52  0 0 0 0 0 0 0 0 0 0
MGC9※  0 0 1 0 0 0 0 0 1 0
```

第10図　出力のタイム・チャート

で，10進1桁の加算回路である．これを使って LSS 自体を手直しした．2は簡単な例題として，論理回路

第3表　LSS Ⅱの応用結果

	論理回路名称	入力端子を含む素子個数	コンパイルドプログラム命令個数	コンパイルドプログラムの実行時間	Runningに使用した延時間	備考
1	加算回路	26	300 (150 W)	2秒	約5時間	練習用
2	計数回路	42	400 (200 W)	2秒	約2時間	練習用
3	磁気テープ装置制御回路	374	5,200 (2,600W)	20秒	約20時間	設計ミス3件
4	マージ制御回路	100	1,420 (710 W)	6秒	約50時間	設計ミス3件
5	Main control	729	12,000 (6,000 W)	50秒	約50時間	訂正約15%

設計者側での LSS の使用練習の意味で行なった．この結果完成したものが LSS Ⅰ であり，この LSS Ⅰを使って，8，9月中に実際に使用するために設計された論理回路3，4のシミュレーションを行ない回路の検査を行なった．検査方法としては，設計された論理回路3，4のシミュレーションを行ない，回路の検査を行なった．検査方法としては，設計者が制御プログラムおよび，外部条件のタイム・チャートを作製し，必要な素子のタイム・チャートを 30 クロック分ずつ作製し，予定どおりのタイム・チャートを画くかどうか調べる．この結果若干の論理設計の誤りが発見され，LSS として予期どおりの結果が得られた．処理時間も十分実用になると考えられる．この結果 LSS Ⅰを改善し，LSS Ⅱを作った．

5は新計算機の Main control の論理回路の例である．この場合，各種のレジスタ，演算回路など，全部シミュレートすると，素子の個数がかなり大きくなり，LSS でシミュレート不可能になるので，各種レジスタ，演算回路，すでに LSS で試験ずみの部分回路については，LSS ではシミュレートせずに，別にサブ・ルーチンを作ってシミュレートし，これを EXCT の命令で LSS の SIML の命令とつなぎ，Main control のみは検査できるようにした．非常に大規模の論理回路では，部分的に検査して，このようにつなぎ合わせるのは，きわめて有効な方法と思われる．この Main control は，割り込み機能を含み，動作がかなり複雑なため，かなり多くの論理的誤りが発見された．このような複雑なものを一旦配線を完了したあとで手直しすると，かなりの労力を要することが予想され，計算機による設計の検査は，論理設計において，有力な武器であることが立証された．

なお，実際の論理回路を LSS でシミュレーションする際，設計図から人手で論理入力表を書きうつし，

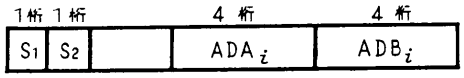
さらにそれを鍵盤さん孔する手段をとった．ところが書きうつしの際の誤り，さん孔の誤りが，論理的誤りに比べ遙かに大きい数になり，コンパイルに長時間を要するので，5の場合には，さん孔は2人独立に同じものをさん孔させ，あとで照合して喰い違った所を検査・訂正するという手段で，さん孔ミスを大幅に減らすことができた．

4. 大規模なシステムのシミュレーションの方法（LSS Ⅲ）

これまで述べた LSS Ⅱ によれば，数百個までの論理回路のシミュレーションはできるが，それ以上は不可能である．実際に計算機全体をシミュレートできるためには数千個の素子からなる論理回路をシミュレートすることが必要である．このためにまず計算機を LSS Ⅱ で可能な数個の部分回路に分け，まず，これらの部分回路をシミュレートし，完全に手直しした上で，数個の部分回路のシミュレーション・プログラムを磁気テープに入れて，シミュレーションを行なうプログラムを作製中である．

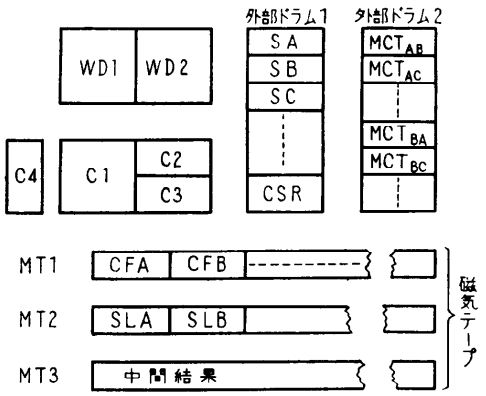
LSS Ⅱ における素子個数の制限は，まず，コンパイラにより作製されたシミュレーション・プログラムが9,000語以内ということと，WD の容量が2,000語であることから来る．このためには，コンパイラで作製されたシミュレーション・プログラムを磁気テープに入れれば，磁気テープ一巻で15万語(NEAC 2203)であるから，約1万個のシミュレーション・プログラムを入れることができる．次に各素子の状態の記憶に外部ドラムを使用すれば，十分に入れることができる．シミュレーション・プログラムとしてはすでに部分的に検査ずみのものを使えば良い．ここで一つ問題になるのは，部分回路相互間の論理配線の処理である．

数個の部分回路を A, B, C, D, ……とする．Aの素子 i にBからの配線があるとき，**第 11 図**のような



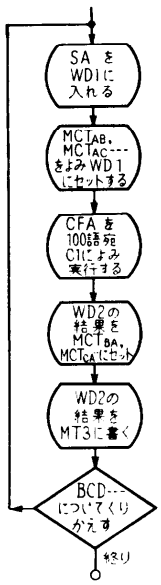
第 11 図　相互配線表の1語

1語により，その相互配線を表わす．ADB i は部分回路Bにおける WD 1 における番地，ADA i は部分回路Aにおける素子の WD 1 における番地である．Aの部分回路えの Bからの配線1本につき1語ずつを設け，これらの全部を表にしたものを相互配線表 MCT_{AB} で表わす．**第 12 図**はこの相互配線表を使用



第 12 図　LSSⅢ における記憶装置の機能割当て

した場合の全システムをシミュレートする時の記憶装置の割り当てを示す．WD, C 1, 2, 3, は LSS Ⅱ と同じである．SA, SB, SC……は各部分回路のシミュレーションの結果（WD 2）を記憶する．CSR は LSS Ⅱ におけると同様，制御用サブルーチンを記憶する．MT 1 は部分回路ごとのシミュレーション・プログラム，MT 2 は部分回路ごとの分類された素子の表，MT 3 は中間結果をそれぞれ記憶する磁気テープ装置である．このような構成によれば**第13図**のようなフロー・チャートで全システムのシミュレーションが可能になる．このような方法で全システムのシミュレーションを行なうに要する時間は NEAC 2203 の場合，磁気テープの走行速度で定まり，5,000 個の素子で約5分～10分程度と推定される．なお，この場合，中間結果を磁気テープに入れ，所要ブロック数のシミュレーション終了後，再読み取りを行なって，高速行印刷機でタイム・チャートを印刷させる．



第13図　LSSⅢのシミュレーションの手順

5. むすび

計画着手後，約3人・5カ月余りで一応実用し得るプログラム

システムLSS II を完成し，その有効性を実証し得ることができた．従来の比較的簡単な計算機の論理設計は兎に角として，最近の先廻り制御，割り込みの技術を大幅に取り入れた計算機の論理設計においては，組立て前にシミュレートとして，動作を完全に論理設計を検査しておくことは，組立後の手直しの時間と労力を大幅に節約するための極めて有効な手段である．

おわりにあたり本研究を行なう機会を与えられた，通信研究所新堀次長，梶電信課長，終始御指導をいただいた岸上調査役，新藤課長補佐に深甚なる謝意を表するものである．

参考文献

1) 淵，西野；電子計算機の配線に関する自動データ処理，情報処理 **1**, No.4 (1960-12) p.213
2) 矢島，高田；計算機による論理布線設計 電気学会連合大会 (1960) 講演予稿 No.38
3) T.A. Connolly; Automatics system and logical design techniques for the RW-33 computer system IRE Conv. Rec. **18** pt 2 (1960) p. 124
4) 伊吹；ディジタル機械論理設計への計算機の応用，電子計算機専門委員会資料 (1960年10月)
5) Leiner, Weinberger, Coleman; Using digital computers in the design and maintenance of new computers. IRE Trans. EC-10 No. 4 (1961-12) p. 680
6) C.W. Rosenthal; Computing Machine aids to a development project. IRE Trans. EC-10 No. 3 (1961-9) p. 400
7) Y.N. Chang, O.M. George; Use of high speed digital computers to study performance of complex switching networks incorporating time dely. AIEE Comm. **46** (1960) p. 982
8) 高島・津田・加藤・戸田・高山；論理構成のシミュレーション・プログラム通信学会計算機研究会 (昭37年10月25日)

附録 制御プログラムのマクロ命令表

(1) 入力用タイム・チャートの変換

CVIT *nY* (Convert input time-chart, $n \leq 30$) 任意個のグループからなる入力タイム・チャート (紙テープ) を読み，一連のグループ表と，それらグループの n クロック分の状態表に変換し，紙テープにさん孔する．

(2) 期初条件の設定，グループ表による状態の設定

RSLT. (Read sorted list) 分類された全素子の名前リスト (紙テープ) を読み，内部ドラム記憶装置の 1,000 番地以降に格納せよ．

RTAB (Read table) 1個のグループ表を読み，全素子の名前の表から番地を索引し，これらの番地からなるグループ表を外部ドラムの指定された位置に格納せよ，

CLWD (Clear working drum) 内部ドラムの0～1999番地を-1にセットせよ．

RSED (Read states to external drum) 1個の状態表 (紙テープ) を読み，指定されたグループ表 (外部ドラム) の各語の符号をセットせよ，

RSWD (Read states to working drum) 1個の状態表 (紙テープ) を読み，指定されたグループ表 (外部ドラム) の番地 k_i に従がって k_i 番地と $k_i+1{,}000$ 番地を状態表に従がって ±1にセットせよ．(iはグループ表全部について)

TSED *nY* (Transfer states from external drum to working drum) n 番のグループ表 (外部ドラム) を引きその表で指定された番地(k_i, $k_i+1{,}000$) の状態を表の符号に従がって ±1にセットせよ．

(3) 結果の印刷

PALS (Punch all states) 全素子の1回のシミュレーションの結果を1素子について0または1で全部紙テープにさん孔せよ．

PSTB *nY* (Punch states according to table) n 番の表に含まれる素子のみについて，PALS と同じことをせよ．

RNSS *nY* (Read N sequences of states, $n \leq 30$) PALS でさん孔したテープを n 回分読みとり，内部ドラム 0～999 番地の対応する位置に入れよ．

RSTS *nY*, *mY* (Read tabulor sequences of states) PSTB m 個を係つさん孔した紙テープを n 回分読み，RNSS と同様に内部ドラムに入れよ．

RSSS (Read single sequence of states) PALSでさん孔した紙テープを1回分よみ，内部ドラムの各番地を±1にセットせよ．

PTCA *nY* (Punch time-charts of all elements $n \leq 30$)全素子の n クロック分のタイム・チャートをさん孔せよ．(この前に RSLT と RNSS または RTSS を行なう)

PTCT *nY*, *mY* (Punch time-charts of table m) グループ表 m に属する素子のみについて，PTCA と同じことを行なう．

(4) マリロ命令の飛び越し，停止，繰り返し．

HALT 停止命令．restart ボタンで次の命令を実行する．

JUMP *nY*, または ***nX***. 無条件飛び越し．nY は絶対番地，nX はこの命令からの相対番地．(n は正または負)

REPT *nY mY*, または ***mX***. (Repeat) 繰り返し．(mY, mX は JUMP と同じ意味．) mY または nX から，この命令までを n 回繰り返し実行せよ．

CPSK *nY*. (Compare and ship) グループ表 n に対応する内部ドラムの状態がグループ表の符号部で表わされる状態と全部一致していればマクロ命令を一つ飛び越せ．

ACDR *nY*, *mY*. (Access directly)
$m=10$ または 11 のとき，n 番の素子のシミュレーションの結果が0または1ならばマクロ命令を一つ飛び越せ．
$m=0$ または1のとき，n 番地と $n+1{,}000$ 番地に −1 または +1を入れよ．

HBRK (Hanlt and braunchion key board) 一時停止．key board を ±nX と押せば，この命令から ±n 番目のマクロ命令へ飛ぶ．

(5) その他

SIML *nY*, *mY*. (Simulate) 外部ドラムの m番のブロックから始まるシミュレーション・プログラムを n 回繰り返せ，(実際上ははとんど $n=1$ である)

EXCT *nY*, *mY*, *lY* (Execute) 上記のマクロ命令で指定される以外の特殊サブ・ルーチンの実行，インデックス・レジスタ 1に n, 2に m を入れ，外部ドラムの l 番地 ($0 \leq l \leq 10{,}000$) を含む50語ブロック を C2 に読みこみ l に対応する命令から実行する．

(昭和38年2月8日受付)