ＦＰＬＡ書込支援システム

宮下弘　　杉山吉　　竹田忠雄

（日本電信電話公社　武蔵野電気通信研究所）

１．はじめに

ＰＬＡ（Programmable Logic Array）は，任意の論理関数を論理積（ＡＮＤ），論理和（ＯＲ）の２段構成により，アレイ状に規格化された構造上で実現する。そのため与えられた論理機能に対してゲート構成を考えることなく機械的に論理設計できる。この方式は、多品種少量生産の傾向を強め設計コストが増大しつつある論理ＬＳＩの一つの有力な実現法と考えられる。ＰＬＡはマスクにより工場で論理を書込むマスクＰＬＡとユーザー側で論理を書込むことのできるＦＰＬＡ（Field Programmable Logic Array）に分けることができる。前者ではより高度の微細加工技術を使用し、高速な論理回路の実現が可能である。一方後者のＦＰＬＡはユーザー側で手軽で安価に設計できることが最大の利点である。

ＰＬＡはアレイ論理の一種として比較的古くから提案されていたが，最近の集積回路技術の進歩により再び注目を集めている［１］。マスクＰＬＡを指向し１チップ上に１個のＰＬＡを搭載し高密度なＰＬＡを目指した研究［２］，あるいは大規模なＬＳＩ中の機能ブロックをＰＬＡで実現し設計コストの低減を狙った研究［３］［４］も行われている。これらの研究ではＰＬＡの構成とともに，設計手法の研究が重要な位置を占めている［３］。ＰＬＡの規格化された構造を有効に利用し設計の効率化を計るためには，設計自動化システムの開発が必須である。本稿ではＦＰＬＡを対象として論理入力からＦＰＬＡデバイスへの書込みまでの一貫した設計システムを構築し実用に供したので報告する。既に［５］において論理シミュレーション用ファイルを介してＨＳＬ（階層仕様記述言語）と結合したシステムについて発表した。本システムはＬＳＩ設計の中核となる設計言語であるＨＳＬと直接インタフェースがとられ、すでに設計されデータベースに蓄積された多くの設計資産を有効に利用できる。また各種ＦＰＬＡの設計に適用できるアプリケーションプログラムを開発し、ＦＰＬＡ書込みまでの設計時間の大幅な短縮化を達成した。

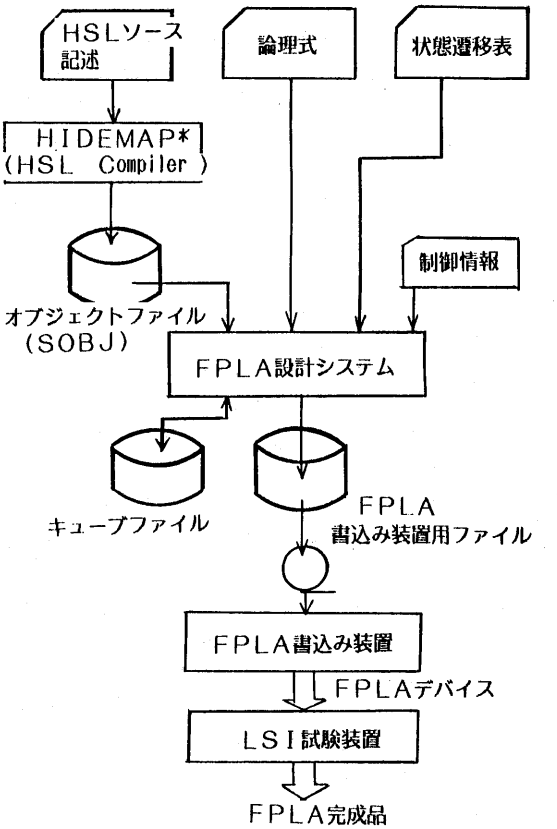


図１　ＦＰＬＡ書込みシステムのフロー

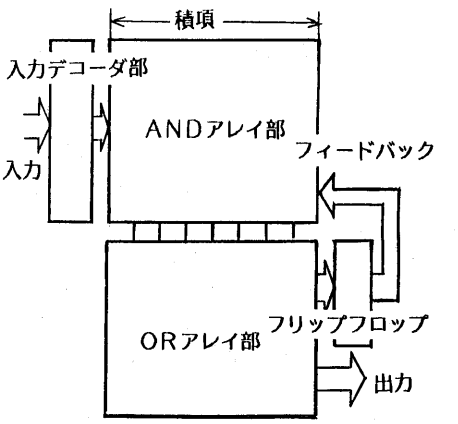


図２　　ＦＰＬＡの構成

２．システム構成

本ＦＰＬＡ書込みシステムは以下の諸点を目標と

*HIDEMAP(Hierarchical Design Data Base Manipulator)

して開発した。①ＦＰＬＡの設計期間の短縮化。②順序回路化ＦＰＬＡへの適用。③ＦＰＬＡ書込み装置，ＬＳＩテスタとの一貫したインターフェイスの確立。図１にＦＰＬＡ書込みシステム全体のフローを示す。入力データは①ＨＳＬソース記述，②論理式，③状態遷移表の３種類が可能である。①と②では一般にフリップフロップを含む順序回路の論理接続情報を記述する。また③では実現する順序回路の「状態」を抽出しその間の状態遷移を表現する。これらの入力形式の選択は制御情報によって指定する。ＨＳＬソース記述を入力とする場合は，ＨＩＤＥＭＡＰシステムにより，オブジェクトファイル（ＳＯＢＪファイル）に変換し，これを入力とする。

ＦＰＬＡ設計システムは上述の各入力に対して①論理の積和形式への変換，②フリップフロップの種類の変換，③積項数の削減を行いキューブファイルを作成する。このキューブファイルはＦＰＬＡ書込みのために必要なデータをすべて含む。また状態遷移表入力の場合は、指定された各状態に対して、状態割り当てを行う。更に、キューブファイルをもとにし，ＦＰＬＡ書込み装置用ファイルを作成する。対象とするＦＰＬＡは２種類あり、各々構成が異なり，対応する書込み装置用ファイルの形式も異なっている。このＦＰＬＡの種類の選択も制御情報によって行う。作成したＦＰＬＡ書込み装置用ファイルは磁気テープによりＦＰＬＡ書込み装置とインターフェイスがとられる。ＦＰＬＡ書込み装置により，ＦＰＬＡデバイスに論理が書込まれる。更にＬＳＩ試験装置によりテストを行いＦＰＬＡの完成品を得る。

３．機能概要

３．１　ＦＰＬＡの構成

本システムで対象とするＦＰＬＡの構成を図２に示す。ＦＰＬＡは①入力デコーダ部，②ＡＮＤアレイ部，③ＯＲアレイ部，④フリップフロップ部，に大別される。入力は入力デコーダを介してＡＮＤアレイ部へ入る。ＡＮＤアレイ部では入力の論理積（ＡＮＤ）がとられ積項が構成される。ＯＲアレイ部ではＡＮＤアレイ部で生成された積項の論理和（ＯＲ）をとり，出力および一部はフリップフロップを介してＡＮＤアレイ部へのフィードバックとなる。このような構成のＦＰＬＡにより任意の順序回路が構成される。入力デコーダは，１ビットデコード方式または２ビットデコード方式を採用し，同一ＦＰＬＡ内ではどちらか一方のみを使用できるものとする。

今，入力数ｍ，出力数ｎのＦＰＬＡを考える。簡単の為フリップフロップは使用していないと仮定する。入力デコーダとしてｋ（１≦ｋ≦ｍ）ビットデコード方式を採用し，このときＰ（ｋ）個の積項を必要としたとする。ＦＰＬＡのアレイ部の面積Ｓ（ｋ）は次の式で評価できる。

$$S(k)=\left(\frac{m}{k}2^{k}+n\right)P(k)$$

ＦＰＬＡの入力数ｍ，出力数ｎを固定しｋを変数と考える。一般に，任意の
ｋ（ｋ≧１）に対してＰ（ｋ）≦Ｐ（１）が成立する。またＡＮＤアレイ部への入力の本数を示す

$$I(k)=\frac{m}{k}2^{k}$$

は，ｋ＝１と２で最小値２ｍをとりｋ≧２では単調増加である。従って，①必要とする積項数は，１ビットデコード方式の場合と比較して増加しない，②ＡＮＤアレイ部への入力数は最小となるという２つの意味で２ビットデコード式が最適となる。このような理由から２ビットデコード方式ＰＬＡが多く使用されている［１］。
またフリップフロップはＪＫ型，ＳＲ型，Ｄ型，Ｔ型の４種類とし，同一ＦＰＬＡ内では１種類のみ使用できるものとする。

３．２　入力形式

（１）ＨＳＬ入力

ＦＰＬＡ化したい論理ブロックの論理接続情報をＨＳＬで記述する。ＨＳＬは任意の階層構造の記述が

表１　使用できる標準ファンクション

機能名	ファンイン数	ファンアウト数
AND	N	1
NAND	N	1
OR	N	1
NOR	N	1
Delay	1	1
NOT	1	1
Exclusive OR	N	1
Power	0	1
Ground	0	1
Wired AND	N	1
Wired OR	N	1
JK型 FF	5	2
SR型 FF	5	2
D型 FF	4	2
T型 FF	4	2

可能である［６］。本システムの入力としては、論理ゲートレベルの接続情報を必要とするので，ＨＩＤＥＭＡＰシステムによりＨＳＬソースを論理ゲートレベルまでマクロ展開したオブジェクトファイル（ＳＯＢＪ）を作成する。

表１に本システムの入力として許されるＨＳＬの標準ファンクションの一覧を示す。また一例として図３．１の論理ブロックのＨＳＬ記述を図３．２に示す。入力したＳＯＢＪより接続情報として等価な論理式ファイルを作成し、これより積和形式への変換を行う。またフリップフロップの種類を指定することによりＳＯＢＪ中のフリップフロップの種類の変換が可能である。この機能を使用し，ＦＰＬＡ上に準備されたフリップフロップを前提として，論理設計が出来る。

（２）論理式入力

論理ブロックの論理式を次の形式で記述する。

Ａ＝Ｂ；

ここでＡは出力変数名，状態変数名，中間変数名のいずれかである。状態変数名は頭１文字の指定により識別する。フリップフロップの入力は，フリップフロップ名の後に「・」を印しそのあとに端子名を記述して示す。例えばＤ型フリップフロップＱＡのＤ端子は「ＱＡ．Ｄ」で表わす。Ｂは入力変数名，状態変数名，中間変数名の論理式である。論理演算子としては／（ＮＯＴ），＊（ＡＮＤ），@（Exclusive　ＯＲ），＋（ＯＲ）を使用する。括弧の使用は自由である。図４に論理式記述の例を示す。

（３）状態遷移表入力

ＦＰＬＡ上で実現する順序回路の機能を記述する。いわゆるオートマトンとしての動作を次式で表現する。

Ｓ，Ｘ：Ｓ：Ｚ；

ここでＳは現在の状態名，Ｘは入力についての論理式であり，記述は（２）の論理式記述に準じる。Ｓは次の状態名を表わす。Ｚは出力の定義であり，１または Don´t　careとなる出力信号名を「，」でくぎって示す。

Don´t　careの信号名の前には「＊」をつける。他の出力信号はＯとする。指定された種類のフリップフロップを使用してＦＰＬＡが実現される。

図５に状態遷移表入力の記述を示す。

３．３　積和形式への変換

ＦＰＬＡの論理設計では任意の論理関数を積和形式に変換し，冗長な積項を削除し積項数の削減をはかる必要がある。このため［７］の手法を利用して積和形式変換をおこなう。この手法では任意の入力デコード方式に対して多出力論理関数を０，１のビット列で表現する。この表現はキューブ表現と呼ばれ入力，出力を区別することなく多出力論理関数を統一的に取り扱うことができる。例えば次の論理関数

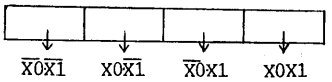
$$Z0=\times0\cdot\overline{\times1}+\overline{\times2}\cdot\overline{\times3}$$

$$Z1=(\times0+\times1)+\times2\cdot\times3$$

に対して次のようなキューブ表現が対応している。

$$F=\left\{\begin{matrix}0100 & 1111 & 01\\ 1111 & 1000 & 01\\ 0111 & 1111 & 10\\ 1111 & 0001 & 10\end{matrix}\right\}$$

ここで第１番目のデコーダに×０，×１，第２番目のデコーダに×２，Ｘ３が入力されているとする。ここで各行は３個のパートより構成されており，最後のパートは出力分に対応している。入力に対応するパートの各ビットは下に示すように論理最小項を表現している。



出力に対応する最後のパートの各ビットは各出力を表現する。同一デコーダ（パート）内では１の立ったビットに対応する論理最小項の論理和をとり，異なるパート間では論理積をとる。出力パートより見れば［１］の立っているビットに対応する積項（行に対応する）の論理和をとることによりその出力が実現されることを示している。上記Ｆのようなキューブの集合をカバーと呼ぶ。［７］で述べられた定理を次のように一般化する。

定理　入力変数は同じで（入力デコード方式も同じとする），出力変数がそれぞれ（f_1，f_2　…，f_n），（g_1，g_2，…，g_n）の２つのＰＬＡがあるとし，それぞれのカバーをＦ，Ｇとする。E_i を i 番目の出力変数の空間とすれば

$F=\bigvee_{i=1}^{n}E_i\cdot f_i$，$G=\bigvee_{i=1}^{n}E_i\cdot g_i$　となり次の関係が成り立つ。

（１）　$F\vee G=\bigvee_{i=1}^{n}E_i(f_i\vee g_i)$

（２）　$F\wedge G=\bigvee_{i=1}^{n}E(f_i\wedge g_i)$

（３）　$F=\bigvee_{i=1}^{n}E_i\cdot\overline{f_i}$

（４）　$F\oplus G=\bigvee_{i=1}^{n}E(f_i\wedge\overline{g_i})$

この定理より論理関数間の演算をカバー上の演算（∨，∧，－，⊕）に置換することができる。ここで⊕は［７］で定義されたＤisjoint Ｓharpと呼ばれる演算である。例えば

$F=\{X0\}$，$G=\{X1\}$　とすればＦ∨Ｇ，Ｆ∧Ｇ，$\overline{F}$，Ｆ⊕Ｇは，各々論理式Ｘ０＋Ｘ１，Ｘ０・Ｘ１，$\overline{X0}$，$X0\cdot\overline{X1}$　に対応するカバーとなる。この手法により多入力の論理関数を効率的に積和形式に変換

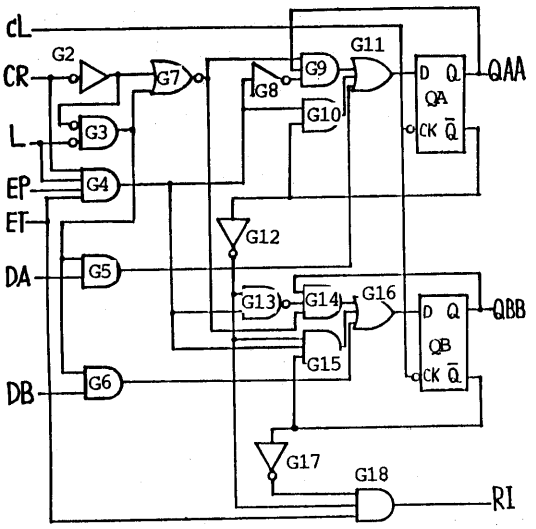
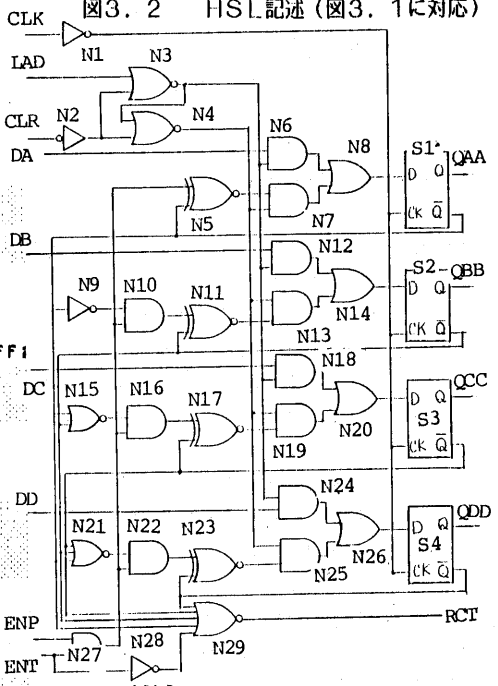


図3．1　論理図

```
NAME      :NAND2;
PURPOSE   :LOGSIM;
LEVEL     :END;
EXT       :1,2,3;
INPUTS    :.1,.2;
OUTPUTS   :.3;
FUNCTION  :NAND,2,1,DELND2;
DELAYS    :DELND2,1,1,1,1,1,1;
END;

NAME      :OR3;
PURPOSE   :LOGSIM;
LEVEL     :END;
EXT       :1,2,3,4;
INPUTS    :.1,.2,.3;
OUTPUTS   :.4;
FUNCTION  :OR,3,1,DELOR3;
DELAYS    :DELOR3,1,1,1,1,1,1;
END;
NAME      :DFF;
PURPOSE   :LOGSIM;
LEVEL     :END;
EXT       :1,2,3,4,5,6;
INPUTS    :.1,.2,.3,.4;
OUTPUTS   :.5,.6;
FUNCTION  :DEPE,4,2,DLDEPE;
DELAYS    :DLDEPE,1,1,1,1,1,1;
END;
```

図3．2　HSL記述（図3．1に対応）




```
IDENT     :TETS07;
VERSION   :MMM.01;
DATE      :80/10/25;
AUTHOR    :H.MIYASHITA;
NAME      :COUNT2;
PURPOSE   :LOGSIM;
COMMENT   :COUNT2;
LEVEL     :BLOCK;
EXT       :CL,CR,L,EP,ET,DA,DB,QAA,QBB,RI;
INPUTS    :.CL,.CR,.L,.EP,.ET,.DA,.DB;
OUTPUTS   :.QAA,.QBB,.RI;
TYPES     :NAND1,NAND2,NOR2,AND2,AND3,AND4,OR3,DFF;
NAND1     :G2,G8,G12,G17;
NAND2     :G13;
NOR2      :G3,G7;
AND2      :G5,G6,G10;
AND3      :G9,G14,G15,G18;
AND4      :G4;
OR3       :G11,G16;
DFF       :QA,QB;
N1        =.CL,QA.1,QB.1;
N2        =.CR,G2.1,G4.1;
N3        =G2.2,G7.1,G3.1;
N4        =.L,G3.2,G4.2;
N5        =.EP,G4.3;
N6        =.ET,G4.4,G18.3;
N7        =G3.3,G7.2,G5.1,G6.1;
N8        =.DA,G5.2;
N9        =.DB,G6.2;
N10       =G4.5,G8.1,G10.1,G13.2,G15.2;
N11       =G7.3,G9.1,G14.3;
N12       =G8.2,G9.3;
N13       =G9.4,G11.1;
N14       =G10.3,G11.2;
N15       =G5.3,G11.3;
N16       =G11.4,QA.2;
N17       =QA.5,.QAA,G9.2;
N18       =QA.6,G10.2,G12.1;
N19       =G12.2,G13.1,G15.1,G18.2;
N20       =G6.3,G16.3;
N21       =G15.4,G16.2;
N22       =G14.4,G16.1;
N23       =G16.4,QB.2;
N24       =QB.5,.QBB,G14.1;
N25       =G15.3,QB.6,G17.1;
N26       =G17.2,G18.1;
N27       =G18.4,.RI;
N28       =G13.3,G14.2;
END;

NAME      :NAND1;
PURPOSE   :LOGSIM;
LEVEL     :END;
EXT       :1,2;
INPUTS    :.1;
OUTPUTS   :.2;
FUNCTION  :NAND,1,1,DELND1;
DELAYS    :DELND1,1,1,1,1,1,1;
END;
```

```
N2=/CLR;
N3=/(LAD*N2);
N4=/(N2*N3);
N27=ENP*ENT;
N28=/ENT;
N9=/(/S1);
N5=N27@(/S1);
N10=N27*N9;
N11=/(N10@(/S2));
N15=/(/S2+/S1);
N21=/(/S1+/S2+/S3);
N16=N15*N27;
N17=/(N16@(/S3));
N22=N21*N27;
N23=/(N22@/S4);
RCT=/(/S4+/S3+/S2+/S1+N28);
N8=(DA*N3)+(N5*N4);
N14=(DB*N3)+(N4*N11);
N20=(DC*N3)+(N4*N17);
N26=(DD*N3)+(N4*N23);
S1.D=N8;
S2.D=N14;
S3.D=N20;
S4.D=N26;
QAA=S1;
QBB=S2;
QCC=S3;
QDD=S4;
```

図4　論理式記述例

できる。更に積項数削減の為のアルゴリズム［７］をプログラム化して各種の論理ブロックのＰＬＡ化に適用した。図６はＰＬＡのアレイサイズと処理時間の関係を示す。アレイサイズの２～３乗のオーダで処理時間が急激に増加していることがわかる。

本システムでは３．４で述べるように比較的小規模のＦＰＬＡを対象とするため上述のような積和形式変換手法の採用により，かなり積項数の少ない解を得ることが可能となる。そこで積項数削減の手法としては簡略化して積項の併合を採用し処理時間の短縮化をはかった。

３．４　ＦＰＬＡ書込装置用ファイルの作成

本システムが対象とするＦＰＬＡの構成を図７に示す。タイプＩは大容量のＦＰＬＡであり１６個のＪＫ型フリップフロップが用意されている［９］［１０］。等価ゲート数はほぼ２００ゲートである。タイプＩＩは小容量，高速のＦＰＬＡであり８個のＤ型フリップフロップが搭載されている。等価ゲート数はほぼ１００ゲートである。このＦＰＬＡでは４本の入出力共用のピンが用意されており，入出力数に対する融通性が増加している。

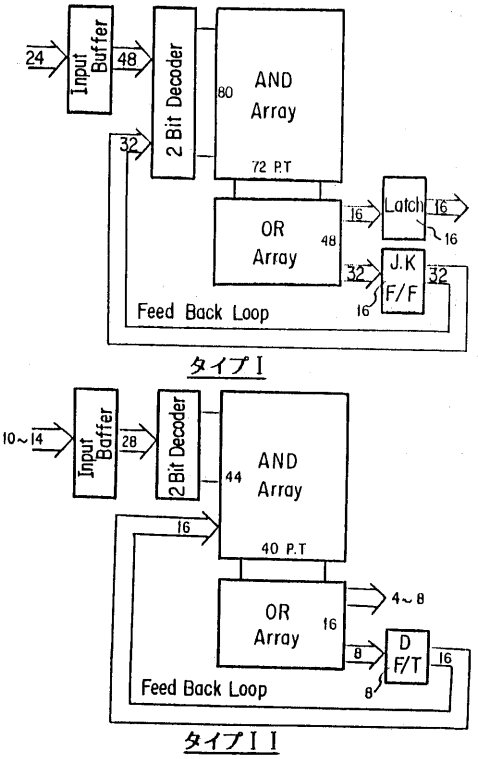
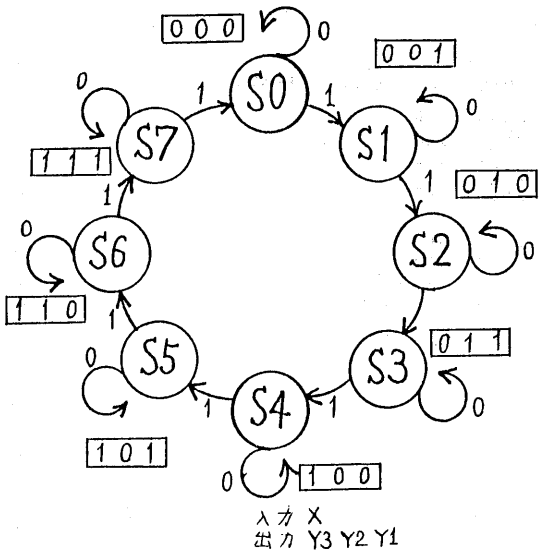


図７　対象とするＦＰＬＡの構成




```
 1      S0,X:S1:Y1;
 2      S0,/X:S0:;
 3      S1,X:S2:Y2;
 4      S1,/X:S1:Y1;
 5      S2,X:S3:Y1,Y2;
 6      S2,/X:S2:Y2;
 7      S3,X:S4:Y3;
 8      S3,/X:S3:Y1,Y2;
 9      S4,X:S5:Y1,Y3;
10      S4,/X:S4:Y3;
11      S5,X:S6:Y2,Y3;
12      S5,/X:S5:Y1,Y3;
13      S6,X:S7:Y1,Y2,Y3;
14      S6,/X:S6:Y2,Y3;
15      S7,X:S0:;
16      S7,/X:S7:Y1,Y2,Y3;
```

図５　状態遷移表入力記述例

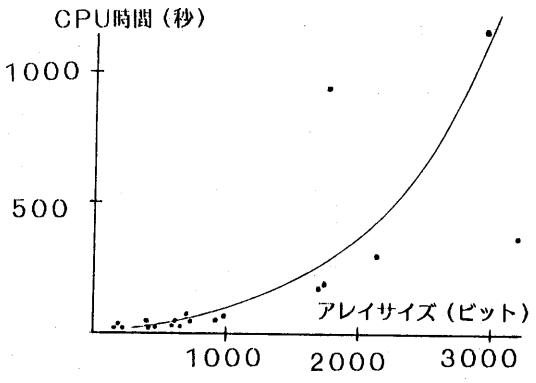


図６　アレイサイズと積項数削減処理時間

```
PIN   ASSIGNMENT   TABLE

X000   ACR
X001   AL
X002   ADB
X003   ADA
X004   AET
X005   AEP

Z000   AQAA
Z001   AQBB
Z002   ARI

JK00   SQB0002
JK01   SQA0001
```

```
                *****   PLA PATTERN   *****

                XXXXXX     JJ       ZZZ      J J
                000000     KK       000      K K
                000000     00       000      0 0
                012345     01       012      0 1

                2.2.2.     2.       ...      JKJK

 1              ......     1.       .1.      ....
 2              ......     .1       1..      ....
 3              ....1.     11       ..1      ....
 4              00....     ..       ...      ..1.
 5              10.0..     ..       ...      ...1
 6              100...     ..       ...      .1..
 7              101...     ..       ...      1...
 8              10.1..     ..       ...      ..1.
 9              .1....     00       ...      .1..
10              .1..NN     .0       ...      ...1
11              .1..NN     01       ...      .1..
12              .1..11     .1       ...      ...1
13              .1..11     11       ...      .1..
14              01....     10       ...      .1..
15              01..NN     .1       ...      ...1
16              01..NN     11       ...      .1..
17              01..11     .0       ...      ...1
18              01..11     01       ...      .1..
19              11....     10       ...      1...
20              11..NN     .1       ...      ..1.
21              11..NN     11       ...      1...
22              11..11     .0       ...      ..1.
23              11..11     01       ...      1...
```

図8　HSL入力実行例　（付録参照）

```
PIN   ASSIGNMENT   TABLE

X000   CLR
X001   LAD
X002   ENP
X003   ENT
X004   DA
X005   DB
X006   DC
X007   DD

Z000   RCT
Z001   QAA
Z002   QBB
Z003   QCC
Z004   QDD

D000   S1
D001     NA
D002   S2
D003     NA
D004   S3
D005     NA
D006   S4
D007     NA
```

```
                *****   PLA PATTERN   *****

                XXXXXXXX   DDDDDDDD     ZZZZZ   DDDDDDDD
                00000000   00000000     00000   00000000
                00000000   00000000     00000   00000000
                01234567   01234567     01234   01234567

                2.2.2.2.   2.2.2.2.     .....   DDDDDDDD

 1              ........   ......1.     ....1   ........
 2              ........   ....1...     ...1.   ........
 3              ........   ..1.....     ..1..   ........
 4              ....1...   1.......     .1...   ........
 5              ...1....   1.1.1.1.     1....   ........
 6              10....1.   ........     .....   ....1...
 7              10.....1   ........     .....   .....1..
 8              10..1...   ........     .....   1.......
 9              10...1..   ........     .....   ..1.....
10              11......   ....0.1.     .....   .....1..
11              11......   ...0.1..     .....   ....1...
12              11......   ..0.1.1.     .....   ....1..1
13              11......   0.1.....     .....   ...1....
14              11......   0.1.1...     .....   ....1...
15              11......   0.1.1.1.     .....   .....1..
16              11NN....   0.......     .....   1.......
17              11NN....   1.1.....     .....   ...1....
18              11NN....   1.1.1...     .....   ....1...
19              11NN....   1.1.1.1.     .....   .....1..
20              1111....   1.......     .....   1.......
21              1111....   1.0.....     .....   ...1....
22              1111....   1.1.0...     .....   ....1...
23              1111....   1.1.1.0.     .....   .....1..
```

図9　論理式入力実行例(付録参照)

```
STATE CODE TABLE

S0      000
S1      001
S2      010
S3      011
S4      100
S5      101
S6      110
S7      111

PIN   ASSIGNMENT   TABLE

X000   X
X001     NA

Z000   Y1
Z001   Y2
Z002   Y3

D000   #D001
D001     NA
D002   #D002
D003     NA
D004   #D003
D005     NA
```

```
                *****   PLA PATTERN   *****

                XX      DDDDDD     ZZZ      DDDDDD
                00      000000     000      000000
                00      000000     000      000000
                01      012345     012      012345

                2.      2.2.2.     ...      DDDDDD

 1              0.      0.0.0.     ...      .....1
 2              0.      0.0.1.     1..      ....1.
 3              0.      0.1.0.     .1.      ...1..
 4              0.      0.1.1.     11.      ..1.1.
 5              0.      1.0.0.     ..1      1.....
 6              0.      1.0.1.     1.1      1...1.
 7              0.      1.1.0.     .11      1.1...
 8              0.      1.1.1.     111      1.1.1.
 9              1.      0.0.0.     1..      ...1..
10              1.      0.0.1.     .1.      ...1.1
11              1.      0.1.0.     11.      ..1.1.
12              1.      0.1.1.     ..1      1.....
13              1.      1.0.0.     1.1      1...1.
14              1.      1.0.1.     .11      1.1...
15              1.      1.1.0.     111      1.1.1.
16              1.      1.1.1.     ...      ......
```

図10　状態遷移表入力実行例（付録参照）

4．実行例及び評価

本システムを用いて各種論理ブロックのＦＰＬＡ化を試みた。図８はＨＳＬ入力の実行例である。この例は図３．１に示した回路図のＦＰＬＡ化を試みたものである。ＦＦ＝ＪＫを指定し，もとの回路図中で使用されていたＤ型のフリップフロップをＪＫ型に変換している。２ビットデコード方式で２３個の積項を必要とすることがわかり，タイプＩのＦＰＬＡを用いてこの回路を実現することができる。ＦＰＬＡ書込み装置用ファイル作成までに必要なＣＰＵ時間は５．４秒程度であった。

図９は図４に示した論理ブロックを論理式入力によりＦＰＬＡ化した例である。この例ではフリップフロップは元の回路図中と同じＤ型を用いた。２ビットデコード方式で２３個の積項を必要とし，タイプＩＩのＦＰＬＡを用いることによりこの回路を実現できることがわかる。この場合ＣＰＵ時間６．９秒程度を必要とした。この例でＦＦ＝ＪＫを指定しフリップフロップをＪＫ型に変換した場合２ビットデコード方式で６６個の積項となり，タイプＩのＦＰＬＡで実現することができる。しかし，Ｄ型のフリップフロップの場合に比較して積項数は大幅に増加している。

図１０は図５の状態遷移表を入力した場合の本システムの実行結果である。Ｄ型のフリップフロップを使用し，２ビットデコード方式で，１６個の積項を必要とする。従ってタイプＩＩのＦＰＬＡで十分余裕を持って実現できることがわかる。又必要としたＣＰＵ時間は３．３秒程度であった。

各種の論理ブロックのＦＰＬＡ化の例を表２にまとめる。ＣＰＵ時間数秒から１０秒程度でＦＰＬＡ書込み装置用ファイル作成が完了している。実際のＦＰＬＡデバイスへの論理の書込みを含めても数分程度で終了しており，ＦＰＬＡの設計時間の大幅な短縮化が達成された。

図１１は３．３で述べた実験プログラムを用いて各種の論理ブロックをＦＰＬＡ化した例である。横軸はアレイサイズ，縦軸は論理ブロックのゲート数を示している。この例はフリップフロップを使用せず，単なる組み合わせ回路としてＰＬＡを使った場合のデータである。タイプＩ及びタイプＩＩのアレイサイズはそれぞれ９．２Ｋ，２．４Ｋビットであり図１１のグラフより等価ゲート数がそれぞれ，２００ゲート，１００ゲート程度であることが確認できる。このグラフでは少し等価ゲート数が低めにでているがこれはフリップフロップを使用していないデータであるためと考えられる。フリップフロップを有効に使用した場合はほぼ上記の等価ゲート数が得られると考えられる。

機　　能	入力数	出力数	P.T 数	F/F 数	等価ゲート	CPU (SEC)
7 Segment Decoder	6	7	16		43	5
4 Bit Full adder	9	5	19		44	9
4 Bit Counter	8	5	46	4	46	7
16-4 Priority Encoder latch	16	4	17	4	67	3
2-1 Mux X 11 latch	23	11	22	11	68	4
4 Bit Function Generator	15	4	50		69	9
16 Bit Prisetable Counter	18	16	64	16	177	19
4 Bit Comparator	11	3	11		33	5

表２　　各種論理ブロックのＦＰＬＡ化

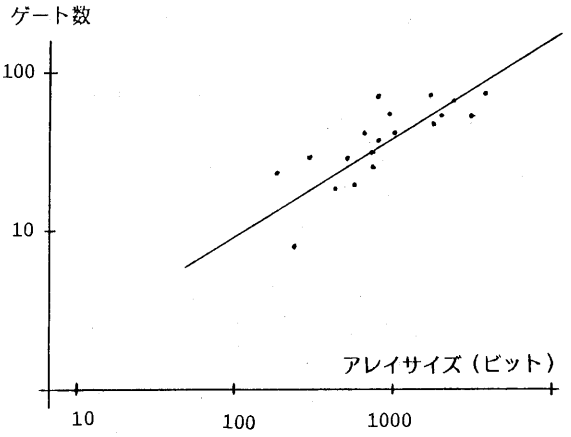


図１１　　アレイサイズとゲート数の関係

5．まとめ

ＦＰＬＡ設計をサポートするために開発したＦＰＬＡ書込みシステムの概要について述べた。このシステムはＬＳＩ設計の共通言語として開発された階層仕様記述言語ＨＳＬとのインターフェイスがとられ，ＦＰＬＡデバイスへの論理の書込みまでの一貫したシステムとなっている。各種装置の開発にあたり制御部などのランダム論理部をＦＰＬＡで置換える要求が随所に生じており，本システムはＦＰＬＡ設計期間の短縮化に大きな力となっている。

今後は大規模なＬＳＩのレイアウト設計に於てはチップ上にＰＬＡ，ＲＯＭなど規格化された機能ブロックを使用するアプローチが一般的となろう。そのためＦＰＬＡのみでなく，マスクＰＬＡによる機能ブロック（ＰＬＡマクロ）の設計自動化システムの検討も必要となる。本システムはこの検討のベースとなると

考えられる。

謝辞　最後に本研究を遂行するに当たり終始御指導を戴いた鈴木敏正集積回路研究部長ならびに須藤常太集積応用研究室長に深謝する。

参考文献

[1] H. Fleisher and L. I. Maissel, "An introduction to array logic," IBM J. Res. Develop. Vol. 19, pp. 98～109 (1975)

[2] R. A. Wood, "A High Density Programmable Logic Array Chip," IEEE Trans. on computers. Vol. C-28, No. 9 pp. 602-608 (1979)

[3] R. L. Golden, P. A. Latus and P. Lowly, "Design Automation and the Programmable Logic Array Macro," IBM J. Res. Develop. Vol. 24, No. 1pp. 23～31 (1980)

[4] B. Vergnieres, "Macro Generation Algorithms for LSI Custom Chip Design," IBM J. Res. Develop. Vol. 24, No. 5, September, pp. 612～621, (1980)

[5] 宮下, 佐久間, 石塚, "PLA設計自動化システムPLACAD", 昭和56年度電子通信学会総合全国大会410 (1981)

[6] 唐津, 星野, 須藤, "LSI設計記述処理システム," 昭和56年度電子通信学会総合全国大会S9-2 (1981)

[7] S. J. Hong, R. G. Cain and D. L. Ostapko, "MINI: A Heuristic Approach for Logic Minimization," IBM J. Res. Develop. Vol. 18, pp. 443～458 (1974)

[8] 宮下, 永谷, "アレイサイズの最小化について" 昭和52年度電子通信学会半導体部門全国大会 81. (1977)

[9] 鈴木, 竹田, 河原田, 宇佐美, "大規模FPLAの検討," 昭和54年度電子通信学会半導体部門全国大会67. (1979)

[10] 竹田, 鈴木, 宇佐美, "大規模FPLAの試作," 昭和55年度電子通信学会総合全国大会393. (1980)

付録　プログラミングポイントの表現法

AND ARRAY(2 bit decoder)	
11	ab
01	$\bar{a}b$
.1	b
10	$a\bar{b}$
1.	a
UU	$ab+\bar{a}b$
PP	a+b
00	$\bar{a}\bar{b}$
EE	$ab+\bar{a}\bar{b}$
0.	$\bar{a}$
NP	$\bar{a}+b$
.0	$\bar{b}$
PN	$a+\bar{b}$
NN	$\bar{a}+\bar{b}$
..	1

AND ARRAY(1 bit decoder)	
1	a
0	$\bar{a}$
.	1

OR ARRAY	
1	1(connect)
.	0(not connect)