計算機アーキテクチャ 45－4
（1982. 6. 1）

ＶＬＳＩ階層仕様記述言語と処理システム

星野　民夫　　唐津　修　　須藤　常太
（日本電信電話公社武蔵野電気通信研究所）

あらまし

ＶＬＳＩの実現には、数万ゲートにも及ぶ論理を、論理的物理的に誤りなく設計し確実に試験を行ない得る技術の開発が必須となる。そのため、ＶＬＳＩ設計全体を包括する設計システムを開発した。その特徴は以下の点にある。

（１）階層的構造化設計手法による大規模設計データの高能率処理。

（２）共通設計記述言語の設定による各設計段階間でのデータの共用。

（３）設計データベースの開発による一元的なデータ管理。

（４）配置配線をはじめとする各種処理の自動化。

（５）使い易いマン・マシン・インタフェースの設定によるコンピュータ処理と人手処理との有機的結合。

配置配線プログラムは階層を意識した自動レイアウトが可能であり、ＶＬＳＩの高品質なマスクパタンを効率的に設計することができる。本システムの利用により人手設計に比べ約一桁の設計工数の削減が実現される。

１．まえがき

ＬＳＩの大規模化化支える技術の一つのに設計技術がある。近年、ＬＳＩプロセス技術の進歩に伴って、数万－数十万素子／チップのＬＳＩの実現が日程に上がって居り、改めてこの設計技術の重要性がクローズアップされている。

近年、このような背景からＶＬＳＩ設計システムの研究が各所で行なわれている。主として設計データベースを含む設計システム全体に対する提案あるいは自動配置配線システムの検討が行なわれてる。しかしＶＬＳＩ設計全体を包括する設計システムを構築し数万ゲート／チップにも及ぶＶＬＳＩ設計に実際に適用した報告はない。

そこで２－３万ゲート規模を当面の目標に置き、その設計工数を人手に比べ１桁以下に削減するＶＬＳＩ設計システムを開発した。このシステムは階層的構造化設計手法を採用すると共に、各種設計段階で共用できる設計記述言語を設定したところに大きな特徴がある。また配置配線を始めとする各種処理の自動化を計った。配置配線プログラムは階層構造を意識したレイアウトを効率的に行なうことができる。本ＣＡＤシステムを２万ゲート相当の論理ＶＬＳＩに適用した結果、設計工数は人手設計と比較して約１／２０以下に減少し、設計品質（ゲート密度）は、人手に匹敵するレイアウトパターンを得た。本システムは、現在の構成でも５万ゲート程度のＬＳＩが設計でき、すこし拡張すれば１０万ゲート近くのＬＳＩ設計にも適用できる。

本文では、本設計システムの概要、共通設計記述言語、設計データベースについて述べ、ＶＬＳＩ設計システムの概要と効果を明らかにする。

２．ＶＬＳＩ設計システム

論理ＬＳＩの大規模化に伴い、その設計がボトルネックになってきている。これは人手で設計するには、余りにも回路規模が大きくなり過ぎたことによる。そのため、論理ＬＳＩの開発における経費の上昇や工程の長期化が益々深刻な問題となってきており、計算機によるＶＬＳＩ設計支援システムが不可欠である。

ＶＬＳＩ設計における問題点を整理すると以下のようになる。

（１）大規模化に伴い、ＬＳＩ設計データの絶対量が増加して来る。このため量の増大に伴うデータの作成、検証の困難度が加速度的に増大する。

（２）設計データの完成度が極めて厳しく要求される。プリント板の設計においては、多少の設計ミスも製造段階のボード配線で救済することが出来たが、ＬＳＩ上でのマスクパターンのミスは致命的である。

（３）高集積化の進展に伴って、単なる部品であった半導体が装置に近い機能を含みうる程になるため、デバイス的要因からシステム的要因までが一つのＬＳＩチップの中で複雑にからみあってくる。このため分野の異なる様々な境界条件を考慮にいれながら設計を行なう必要が出てくる。

（４）論理的要因と物理的要因を総合的に捉えて最敵化をする必要が生じる。例えば、チップ面積を最小化するために、素子技術を踏まえてワイアード・ファンクション等を大幅に導入する場合、論理的な素子の結線と物理的な素子の結線構造が異なってくるが、これらの間の統一的管理と最敵化が必要となる。

（５）ＬＳＩ製造技術が日進月歩である現状を考慮にいれる必要がある。使用プロセスや素子の基本性能が改良される度に設計全体をやりなおさなければならないのでは賽の河原の石積になる。

この様な問題を解決するために次の５点を念頭に置きＶＬＳＩ設計システムを開発した。

（１）大規模化するデータを効率良く設計処理するために無制限の階層があつかえる構造化設計手法をサポートする事。

（２）システム全体で共通に使用出来る共通設計記述言語を設定し、設計段階間でデータの共用をはかる事。

（３）設計データベースを開発し一元的にデータ管理を行なう一方、設計資産として蓄積再利用が計れるよう工夫すること。

（４）人手がデータに直接関与して、ミスの誘発することを避ける為、各種自動化アルゴリズムのプログラム化を進めること。

（５）設計システムによる設計結果が満足の行くものであるかどうか、設計者が容易に判断出来るよう使いやすいマン・マシン・インタフェースを設定し人間の経験が充分いかせるインタラクティブな切り口を設定する事。

本ＶＬＳＩ設計システムの全体構成図を図１に示す。システムの中央に設計データベースを配置してあり、共通入力言語であるＨＳＬ（Hierarchical Speceficati on Language）で記述されたデータはここに蓄積管理される。設計データのうち、各種ＬＳＩの設計に共通に用いられるセルの記述は、ライブラリとして用意されており、必要に応じて引用できる。データベースの周辺には、ＨＳＬ記述のコンパイラ、逆コンパイラ、階層展開プログラム、ブロック再分割プログラムやワイアード論理生成プログラム等のＨＳＬ処理ソフトウェア群、論理シミュレータ、タイミング・シミュレータ、回路シミュレータ等の各種シミュレータ群、その他テスト・パターン発生プログラムやインタフェース・プログラム等が配置されている。各種のシミュレータ等の支援ソフトウェアで検証された設計データは、自動配置配線プログラムによって自動レイアウトされた後、ルール・チェック・プログラムによる検査をへてマスクパタンジェネレータ等のデータとしても出力される。配置配線の結果が妥当かどうかの判断は、インタラクティブ・ターミナルやプロット図での判定のほか、配線結果を電気特性にフィード・バックさせて詳しいシミュレーションを行なう道が用意されている。

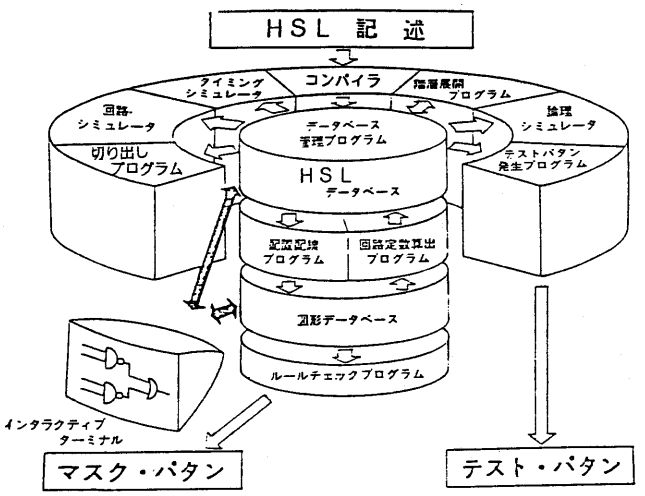


図１　ＶＬＳＩ設計システム

このように本ＶＬＳＩ設計システムにおいては、一度ＨＳＬによってデータを入力すれば、以降の処理は全てシステムによって自動的に行なわれる。人間は要所要所に於ける判定を行なってＧＯ／ＮＯＧＯの指示をだしさえすれば良い。本システムにおいては設計途中のデータをて手作業で修正したり、出力されたマスクパターンを人手で修正したりする事は一切ない。

設計データベースは、ＶＬＳＩ設計システムの中核を成すものであり、ＨＩＤＥＭＡＰ（Hierarchical Design Database Manipulator）システムとしてまとめられている。このＨＩＤＥＭＡＰにはＨＳＬ記述データのコンパイラ、逆コンパイラ、階層展開、切り出し処理、データベース入出力等の他に、各種検証プログラムへのインターフェイス処理を行なうプログラムも含んでおり、それぞれＨＩＤＥＭＡＰコマンドでアクセスできる。表１にＨＩＤＥＭＡＰコマンド一覧表を示す。

ＨＳＬコンパイラでは、ＨＳＬで記述した設計データ（ソースデータ）の文法誤りを検査した後、オブジェクト・コードに変換する。このオブジェクトは直接データベースに登録出来る。データベース上でのデータ管理は個人（ユーザ）ファイル、プロジェクトで共用出来るプロジェクト・ファイル、多数の人の共用出来るライブラリ・ファイルがあり、データの保護と設計資産の共用を計っている。すでに１０００を越える機能セルが標準ライブラリに登録されており、ドキュメントと共にユーザに提供されている。設計データはセル、ブロック、チップといった階層ごとのまとまりのある部分ごとに記述されたモジュールが単位となっている。

表１　ＨＩＤＥＭＡＰコマンド一覧

コマンド種別	コマンド名	機能概要
ＨＩＤＥＭＡＰ コントロールコマンド	＊ＨＩＤＥＭＡＰ	ＨＩＤＥＭＡＰシステムを起動する
	＊ＥＮＤ－ＨＩＤＥＭＡＰ	ＨＩＤＥＭＡＰシステムを終了する
ＨＳＬ処理コマンド	＊ＣＯＭＰＩＬＥ	ＨＳＬコンパイラを起動する
	＊ＥＬＩＰＭＯＣ	ＨＳＬ逆コンパイラを起動する
データベース アクセスコマンド	＊ＤＢＡＤＤ	データベースにデータを書込む
	＊ＤＢＲＥＤ	データベースからデータをよみだす
	＊ＤＢＤＥＬ	データベースのデータを削除する
	＊ＤＢＣＯＰＹ	データベース上でデータをコピーする
	＊ＤＢＳＴＡＴ	データベースの統計情報を出力する
	＊ＭＳＥＴＵＰ	データベースのデータを親子関係にしたがってよみだす
オブジェクト 操作コマンド	＊ＥＸＰＡＮＤ	マクロ展開を起動する
	＊ＷＧＡＴＥ	ドット処理等を行う
	＊ＳＣＩＳＳＯＲ	モジュールデータの切り出しを行う
	＊ＥＬＩＭＩＮＡＴＥ	出力先を持たないサブモジュールをリスト出力する。
	＊ＭＯＤＥＬ	配線容量と遅延の算出，ディレイ素子の付加を行う。
	＊ＲＥＤＥＦＩＮＥ	ＳＯＢＪデータの変更を行う。
	＊ＣＯＭＰＡＣＴ	展開後オブジェクトの整理圧縮を行う
アプリケーション インタフェースコマンド	＊ＴＥＧＡＳ３	ＴＥＧＡＳ３インタフェースを起動する
	＊ＳＰＩＣＥ	ＳＰＩＣＥインタフェースを起動する
	＊ＡＳＰＥＣ	ＡＳＰＥＣインタフェースを起動する
	＊ＬＯＴＡＳ	ＬＯＴＡＳインタフェースを起動する
	＊ＴＥＧＡＳ４	ＴＥＧＡＳ４インタフェースを起動する

こうして作られたデータベース内の設計データに対して、各種の検証処理や自動レイアウト処理を施す。ところが、検証処理や自動レイアウト処理ではモジュール単位でなくて、チップ全体や各ブロックごとの接続情報を用いる事になる。こうした情報は、階層展開プログラムを用いて準備することができる。又、デバイス技術の進歩にも、階層展開プログラムを用いることにより、比較的容易に追随できる。つまり同じ論理仕様のＬＳＩを新しいデバイス技術で設計する場合もブロック・レベル（時にはゲートレベル）のデータをそのまま残し、トランジスタ・レベルの回路の記述を書き替えるだけで済むからである。

切り出し処理プログラムも、階層展開処理プログラムと同様、階層化設計をうまく進めるのに大きな役割を果す。ＨＳＬの記述は論理的にまとまりのあるブロックやセル単位で行なわれる場合が多いがこの階層化構造を固定的にしてしまうと、足枷になる場合が生ずる。例えば、レイアウト処理においては、レイアウトの単位であるブロックが必ずしも論理的にまとまりがある必要はない。レイアウトに適したブロック構成を取るべきである。

設計データは、時には数Ｍバイトからのデータ量になり、モジュールの数も数百からなる場合がある。そのためにデータベースへのアクセスの容易さとそのパーフォーマンスが問題となる。

必要とするモジュールの集合を全てデータベースから読出す場合、そのすべてのモジュール名を指定することは繁雑で手間がかかる。そこで階層構造の頂上のモジュールのみを指定するのみで、あとは親子関係をたどりながら自動的にモジュール・データを集める＊ＭＳＥＴＵＰコマンドを用意した。またＣＯＬＬＥＣＴＩＯＮという項目を用意し、モジュールの集合の操作の便を計っているを示す。このような豊富な管理項目により、優先順位を項目ごに指定することで目的とするモジュールをデータ・ベースから容易に読み出せる。

データ・ベースのパフォーマンスを向上させるためには、データをブロッキングして、一度に大量のデータが読み出せるよう設計した。

表２　ＨＳＬ管理項目一覧

分　類	管理項目	内　容
データの所属	USER	各設計者
	PROJECT	プロジェクト名称
	LIBRARY	ライブラリ名称
データの属性	NAME	モジュール名称
	IDENT	チップ名称
	PURPOSE	使用目的名
	PROCESS	使用プロセス名称
	VERSION	版数名称
	LEVEL	階層レベル名称

３．階層化仕様記述言語ＨＳＬ

階層化仕様記述言語ＨＳＬは、ＶＬＳＩ設計システムの各種ＤＡプログラム用共通入力言語として開発された。従来の設計システムは設計段階に応じて各種の設計支援プログラムが存在し、それぞれが独自の入力形式を用いていることが多かった。そのため、設計のステップが進むたびに人手による入力データの変換等が必要となり、ミスの混入の元となっていた。またＬＳＩの大規模化に伴いデータの作成、検証の困難度が加速度的に増大してきたため、入力言語の一元化は必須となる。入力言語の一元化を計ることによりデータの再コーデングの工数の削減及びミス混入の防止を計ることがてできる。しかし、入力言語のを一元化するためには、以下の問題を解決する必要がある。

（１）配置配線プログラムは、回路の物理的構造データを必要とするが、論理シミュレータでは、論理的構造データが必要となる。これらはワイアード・ファンクションや相方向バス構造を用いた場合、同一構造にならない。
（２）論理シミュレータと回路解析プログラムで、ブロックレベルの記述は同一構造であるが、セルやゲートレベルでは構造が異なる。そこで同一の構造のデータは出来るかぎり共用できること。
（３）論理シミュレータでは電源ラインは不要であるが、回路解析プログラムでは必要となる。
（４）各種検証プログラムで必要とするゲートやトランジスタの解析パラメータをぞれぞれのプログラムごとに引渡せること。
（５）回路図入出力に対応して図面位置等を記述出来ること。
（６）端子が使用されない場合の終端処理が自動的に成されること。
（７）大規模なデータが効率良くあつかえる事が必要となる。

これらの問題を解決すべくＨＳＬ言語は以下の特徴を持たせた。
（１）システムを幾つかのブロックに分割しながら、階層的に記述出来る。そのために設計をトップダウン的またはボトムアップ的に容易かつ効率よく行える。
（２）一度設計した機能ブロックをライブラリ化することにより設計資産として蓄積再利用可能なように、データベース化のための管理情報用データもＨＳＬの中で木目細かく指定可能である。表２にＨＳＬで記述できる管理項目一覧を示す。
（３）ＬＳＩ設計ＤＡ用各種設計支援ソフトウェア（論理シミュレータ、回路シミュレータ、配置配線プログラム、テストパターン・ジェネレータ等）の共通入力とて使用出来るように工夫されており、データの書換によるミスの混入が避けられる。この結果、従来多くの労力と時間をついやした各設計段階間での計算機データ作成工数が１／４から１／１０にと大幅に削減できた。表３にデータ作成工数例を示す。
（４）ＶＬＳＩ設計における物理的構造と論理的構造との間のギャップを埋めるためにのデータが記述出来る。それらは（ｉ）ドット、相方向バスのある回路では、ネットワーク構造は物理

表3　HSLデータ作成工数

	HSL記述	カードパンチ	ファイル登録・修正	合　計	備　考
500ゲート	16人時	8人時	4人時	28人時	論理図 ↓ ファイル登録
4000ゲート	140人時	70人時	10人時	220人時	

	回路シュレータ用記述	カードパンチ	ファイル登録・修正	合　計	備　考
500ゲート	24人時	12人時	8人時	44人時	回路図→ファイル

	論理シュレータ用記述	カードパンチ	ファイル登録・修正	合　計	備　考
4000ゲート	280人時	140人時	40人時	460人時	論理図→ファイル

的構造を記述する。論理的構造はDEFAULT文に指定。よって指定された機能をもつ仮想ゲートが自動的に発生される。

（ii）電源ライン：回路シミュレータには必要で論理シミュレータには不要な電源ラインを、必要な場合のみ発生する事を指示する、電源ピンの宣言文（POWERS）が用意されている。

（iii）空きピン：入力端子が未使用で無結線のまま放置された場合の処置（入力の固定信号値）を指定できる。

（5）各種設計支援ソフトウェアで必要となるパラメータを記述出来る。

（i）論理シミュレータ、テストパターン・ジェネレータ用データ：論理ファンクション、遅延情報

（ii）回路解析プログラム：素子パラメータ、モデルパラメータ

（iii）タイミング・シミュレータ：論理ファンクション、モデルパラメータ

（iv）設計評価プログラム：最大許容負荷係数、最大許容ドット係数、入力負荷係数、遅延算出係数

（v）回路図入出力プログラム：図面位置、外形情報

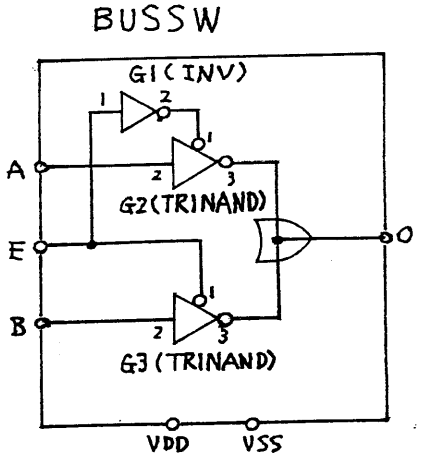
（vi）配置配線プログラム：端子位置、配線禁止領域、外形、端子の等価性

（6）VLSIの設計作業には企画段階から装置やシステムの設計者やユーザからVLSI製造担当者までの多くの専門家が参画する必要がある。これら組織や専門の異なる人達の間で必要な情報を正確に伝える事は極めて難しかったが、HSLを用いることによりVLSIの仕様を正確に伝える事が可能となる。

図2にHSLの記述例示す。この記述例では、ドットのある回路BUSSWについて、その記述が論理シミュレータ用の記述と回路シミュレータ用の記述に共用されていることを示している。図2（b）の回路に対応するHSL記述を図2（a）に示してある。記述の最初の部分（IDENTからCOMMENTまで）は、チップ全体の管理項目である。NAMEから最後のENDまでがモジュールBUSSWの記述である。PURPOSEでこの記述が論理シミュレータ（LOGSIM）、配置配線プログラム（ROUTER）、回路解析プログラム（SPICE）で用いることを示している。POWERS文で電源端子としてVDDとVSSがあることが示されている。しかし、電源ラインの結線はされていない。これは論理シミュレーションで邪魔になるからである。またドットがとられている結線はNETDOTで定義され、初期入力では仮想ゲートは挿入されていない。BUSSWで用いられている基本エレメントTRINANDは、論理シミュレータ用記述と回路シミュレータ用の記述が異なり、それぞれ図2（c）、（d）と（e）、（f）、（g）に示してある。論理シミュレータによう記述（c）には、論理ファンクション名、遅延定義等を記述してある。またDEFAULT文にドットで作られる仮想ゲートの種類を示すO（Wired－OR）が記述されているため、後の処理でゲートを自動的に生成する。回路シミュレーション用記述では、トランジスタ6個からなる回路接続が記述されている。そのトラン

```
IDENT     :EXAMPLE ;
VERSION   :VR01.00 ;
AUTHOR    :DENDEN TARO ;
PROJECT   :EXP082 ;
COLLECTION: EX001 ;
COMMENT   : THIS IS AN EXAMPLE OF DOTTING FUNCTION ;
"---------- END OF MANAGEMENT FIELD ----------"
NAME      :BUSSW ;
EXT       :A,B,E,O,VDD,VSS ;
PURPOSE   :LOGSIM,ROUTER,SPICE ;
LEVEL     :CELL ;
INPUTS    :.A,.B,.E ;
OUTPUTS   :.O       ;
POWERS    :.VDD,.VSS      ;
TYPES     :INV ,TRINAND   ;
INV       :G1             ;
TRINAND   :G2,G3          ;
NETA      =FROM(.A)       TO(G2.2);
NETB      =FROM(.B)       TO(G3.2);
NETE      =FROM(.E)       TO(G1.1,G3.1);
NETG1     =FROM(G1.2)     TO(G2.1)      ;
NETDOT    =FROM(G2.3,G3.3) TO(.O);
END;
```

（a）BUSSWのHSL記述



（b）BUSSWの回路図

図2　BUSSWのHSL記述例

れらの記述を用いてシミュレーションを行なう場合の例を図3に示してある。まず*MSETUP（モジュールセットアップ）コマンドを用いて、必要となるモジュールのデータを読みだしてくる。この際、どのような目的のデータを用いるかを、*PRIORITYコマンドを用いて指定する。図3の左側が論理シミュレータの時の例であり、右側は回路シミュレーションの場合である。論理シミュレーション用のデータには、自動的に仮想ゲートが挿入される。回路シミュレーション用データでは、自動的に電源ラインを結線する指定を行ないトランジスタ・レベルの回路記述とトランジスタ・モデルが回路シミュレータに渡される。この様な簡単な指定で、目的にあったデータの読みだし、変換ができデータの共用が可能となった。

4．適用例とその効果

本VLSI設計システムは、1979年より運用され、各種の通信用VLSIが設計された。その1例として、17Kゲート・ランダムロジックと2KビットのRAMから構成される32bit CMOS VLSIのプロセッサを図4に示す。HSL記述は、CELL、BLOCK、CHIPの3階層を用いて行なった。この3階層構成は、自動配置配線が効率良く成されるように、図5に示す7階層構成に、切り出しプログラムを用いて再構成した。RAMのブロックは人手で設計したが、他のランダム・ロジックのブロックは、全て自動配置配線プログラムを用いて設計した。設計工数は2人月であり，そのレイアウト結果は、542トランジスタ／㎜（全体のトランジスタ数は78K）であった。人手設計に比較して同等以上の集積度が得られた。

5．あとがき

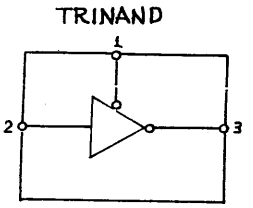
本VLSI設計システムは階層化仕様記述言語HSLを共通入力言語とし、データベースを中心にすえ、各種設計検証プログラム群、配置配線プログラム群、言語処理プログラム群を配した一貫システムである。設計工数は従来の1割以下となり、規模において1万ゲート以上のVLSIが高密度に設計できることが確かめられた。

謝辞

本研究の遂行に当り終始御指導いただいた渡辺誠研究開発本部副本部長及び鈴木敏正集積回路研究部長を始めとする関係各位に深謝します。


```
NAME      :TRINAND  ;
EXT       :1:3      ;
PURPOSE   :LOGSIM   ;
LEVEL     :END      ;
DEFAULT   :1,1,0    ;
INPUTS    :.1:2     ;
OUTPUTS   :.3       ;
FUNCTION  :TRINAND.2,1,D1;
DELAYS    :D1,6,5,7,4,3,6;
END;
```

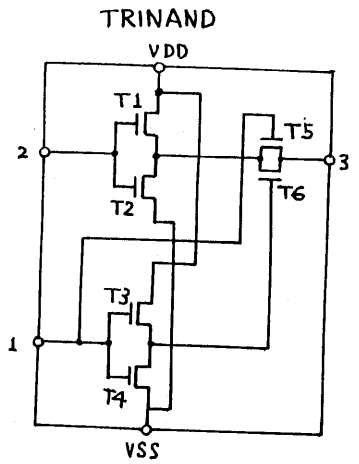
（c）TRINANDのHSL記述



（d）TRINANDの回路図

```
NAME      :TRINAND  ;
EXT       :1:3,VDD,VSS ;
PURPOSE   :SPICE,ROUTER ;
LEVEL     :CELL       ;
POWERS    :.VDD,.VSS  ;
INPUTS    :.1:2       ;
OUTPUTS   :.3         ;
TYPES     :PMOS,NMOS  ;
PMOS      :T1,T3,T5   ;
NMOS      :T2,T4,T6   ;
NET1      =.1, T1.G, T2.G      ;
NET2      =.2, T3.G, T4.G, T5.G      ;
NETVDD    =.VDD, T1.D, T3.D          ;
NETVSS    =.VSS, T2.S, T4.S          ;
NET01     =T1.S, T2.D, T5.D, T6.D    ;
NET02     =T3.S, T4.D, T6.G          ;
NET3      =.3,   T5.S, T6.S          ;
END ;
```

（e）TRINANDのHSL記述（回路解析用）



（f）TRINANDの回路図（回路解析用）

```
NAME      :PMOS;
PURPOSE   :SPICE     ;
LEVEL     :END       ;
PROCESS   :CMOS2U    ;
EXT       :D,G,S     ;
ELEMENT   :PMS3  ,15 ;
PARAMETER:BULK=5,VT=-.7,BETA=1.25E-5,MOB=99,CLM=1,
          GZI=5.34E-5,EO=4.11E4,EC=2.0E6,EOX=3.9,COX=9.87E-16,
          LATD=.295,GAMMA=.5,ALS=.436,GMD=.01,PHI=.32          ;
END;
```

（g）トランジスタのHSL記述

図2　BUSSWのHSL記述例

```
*MSETUP:LIST=(STR1,CTL3);
   *SELECT:NAME=BUSSW;
   *PRIORITY:PURPOSE=(LOGSIM,#);

        *****  MODULE CONSTRUCTION  *****

BUSSW(BUSSW)
   .G1(INV)
   .G2(TRINAND)
   .G3(TRINAND)

   'NAME    BUSSW
   'IDENT   EXAMPLE
   'VERSION VR01.00
   'DATE
'    THIS IS AN EXAMPLE OF DOTTING FUNCTION
MODE 2 ¥
READNET ¥
*LIST ¥
*PARTIAL ¥
*DELAYS ¥
D1              6       5       7       6       5       7 ¥
D2              6       5       7       4       3       6 ¥
DO              0       0       0       0       0       0 ¥
*TYPES ¥
INV         NOT             1       1    D1  ¥
TRINAND     TRINAND         2       1    D2  ¥
DOT00001    WOR             2       1    DO  ¥
PI          PI              0       1    DO  ¥
PPI         PPI             0       1    DO  ¥
PWR         PWR             0       1    DO  ¥
GRND        GRND            0       1    DO  ¥
PPO         PPO             1       0    DO  ¥
PRINT       PRINT           1       0    DO  ¥
*ELEMENTS ¥
G1          (INV)       'G1'  E  ¥
G2          (TRINAND)   'G2'  G1.1 A.2  ¥
G3          (TRINAND)   'G3'  E.1 B.2  ¥
GEN00001    (DOT00001)  'GEN00001'  G2(3) G3(3)  ¥
A           (PI)        '.A'  ¥
B           (PI)        '.B'  ¥
E           (PI)        '.E'  ¥
O           (PPO)       '.O'  GEN00001  ¥
VDD         (PI)        '.VDD'  ¥
VSS         (PI)        '.VSS'  ¥
*FILE ¥
*END ¥
ENDBLOCK ¥
```

図3（a）　論理シミュレータ用記述の発生例

```
*MSETUP:LIST=(STR1,CTL3),OUTSOBJ=SOBJ06;
   *SELECT:NAME=BUSSW;
   *PRIORITY:PURPOSE=(SPICE,#);

BUSSW(BUSSW)
   .G1(INV)
   .    .T1(PMOS)
   .    .T2(NMOS)
   .G2(TRINAND)
   .    .T1(PMOS)
   .    .T5(PMOS)
   .    .T3(PMOS)
   .    .T4(NMOS)
   .    .T2(NMOS)
   .    .T6(NMOS)
   .G3(TRINAND)
   .    .T1(PMOS)
   .    .T5(PMOS)
   .    .T3(PMOS)
   .    .T4(NMOS)
   .    .T2(NMOS)
   .    .T6(NMOS)

*   BUSSW
*   EXAMPLE
*
*   VR01.00
*    THIS IS AN EXAMPLE OF DOTTING FUNCTION
.OPTIONS     ACCT LIST NODE
.TRAN        1NS  100NS
.PLOT        TRAN V(13) V(14) V(15)
VIN          13       0       PWL(0.0.100N.0.120N.5V.300N.
.MODEL       PMOS                PMOS        (BULK=5.VT=-.7.BE
+       MOB=99.CLM=1.GZI=5.34E-5.E0=4.11E4.EC=2.0E6.EOX=.
+       COX=9.87E-16.LATD=.295.GAMMA=.5.ALS=.436.GMD=.01.
.MODEL       NMOS                NMOS        (BULK=5.VT=-.7.BET
+       MOB=99.CLM=1.GZI=5.34E-5.E0=4.11E4.EC=2.0E6.EOX=3.
+       COX=9.87E-16.LATD=.295.GAMMA=.5.ALS=.436.GMD=.01.P
M1      18      13      14      0       PMOS        15
M2      14      13      19      0       NMOS        15
M3      18      14      16      0       PMOS        15
M4      16      14      19      0       NMOS        15
M5      18      11      17      0       PMOS        15
M6      17      11      19      0       NMOS        15
M7      16      11      15      0       PMOS        15
M8      16      17      15      0       NMOS        15
M9      18      13      20      0       PMOS        15
M10     20      13      19      0       NMOS        15
M11     18      12      21      0       PMOS        15
M12     21      12      19      0       NMOS        1
M13     20      12      15      0       PMOS
M14     20      21      15      0       NMOS
.END
```

図3（b）　回路シミュレータ用記述の発生例

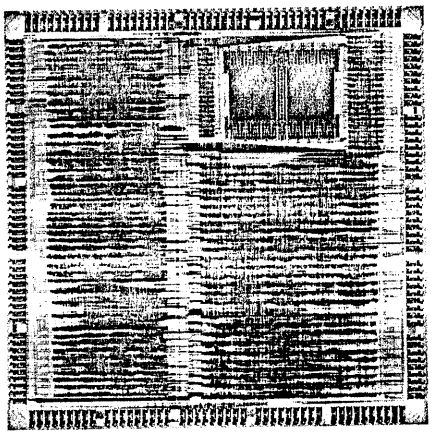


図4　32bit　CMOS　VLSIレイアウト図

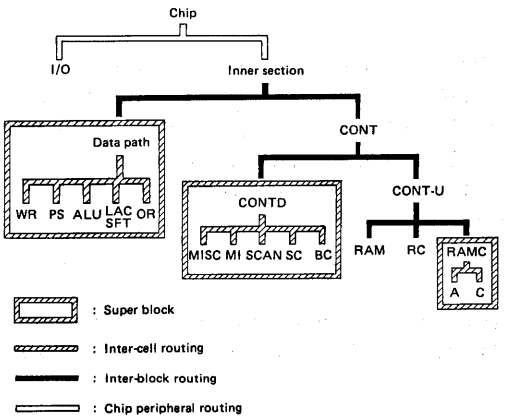


図5　レイアウト用階層構成

参考文献

1）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講演論文集、S9－2、1981年

2）永谷、杉山、堀口、須藤、「機能セルによるVLSIレイアウトプログラム」、同上、431、1981年

3）唐津、須藤、「LSI階層仕様記述言語（HSL）」、電気学会システム・制御研究会SC81－12、1981年1月

4）van Cleenput，W. M.，"A Hierarchical Language for the Structural Description of Digital Systems"，Proc．of 14th D. A. Conf．pp. 272－279, June 1977.

5）Y. Sugiyama，M. Suzuki，Y. Kobayashi and T. Sudo，"A Subnano second 12K gate Bipolar 32bit VLSI Processor"，Proc．of Custom IC Conf．，pp. 73－77, 1981.

6）S. Horiguchi，H. Yoshimura，R. Kasai and T. Sudo，"An Automatic Designed 32b CMOS VLSI Processor"，ISSCC82，pp. 54－55, 1982.