設 計 自 動 化 46－6
（１９８９．２．２１）

統合化セルライブラリデータベースＹＩＬＤ

押切 実　　田村 明穂

ヤマハ（株）　半導体研究所

スタンダードセル方式ＬＳＩ設計システムにおける、セルライブラリのデータベース化について報告する。スケマティックキャプチャ、論理シミュレータ、セル自動配置配線等のツール群から構成されるこの設計システムでは、各ツール毎にセルライブラリが必要となり、それらの総数は数千種類にもなる。そこで、設計システムを効率よく運用していくうえで、これらセルライブラリの管理方式が重要であるとの認識から、我々は統合化セルライブラリデータベースＹＩＬＤを開発し、運用している。ＹＩＬＤとは、各ツールのセルライブラリ情報を統合化しデータベース化したもので、当社のスタンダードセル方式ＬＳＩ設計システムの中核となっている。

Integrated Cell Library Data Base YILD

Minoru OSHIKIRI,　Akiho TAMURA

YAMAHA CORPORATION Semiconductor R & D Laboratories
203 Matsunokijima Toyooka-mura Iwata-gun Shizuoka-ken 438-01 Japan

A database system is implemented to centralize cell library information management for various CAD/CAE tools, thus to facilitate overall standard-cell based LSI design flow. In the conventional standard-cell based LSI design system, cell library information amounts to few thousand parts for a process while they must be consistent through schematic capture, logic simulation, automated placement & routing and so forth. Recognizing the need for and the importance of foolproof library management system, we have implemented YILD, our version of Integrated cell Library Database, which now plays central role in our standard-cell based LSI design system.

1．はじめに

スタンダードセル方式ＬＳＩ設計システムは、実用システムとして現在広く使われている。この設計システムは、自動配置配線ツールを中心にスケマティックキャプチャ、論理シミュレータ、レイアウト検証等のツール群から構成されている。その運用に際しては、各ツール毎にセルライブラリを用意する必要がある。

ここで各ツールの実行に際して必要となるセル情報の主なものを表１－１に示す。

ツール	必要となるセル情報
スケマティックキャプチャ	シンボル
論理シミュレータ	論理ネットワークモデル
セル自動配置配線	セルの外形、端子属性
レイアウト論理照合	Ｔｒネットワークモデル
レイアウトデータ合成	セル内部のレイアウト

表１－１　セル情報

スタンダードセルはひとつのプロセスあたり数百種類程度ある。またプロセスの種類も２μ、１．５μ、１．２μ …と数シリーズを同時に管理する必要がある。

ところで、あるプロセスのあるツール用のセルライブラリ数百種類を短期間で完備することは困難で、一般には半年から一年程度かかって順次そろえていくのが普通である。またその間に修正もかなり生ずる。

この様な背景から、セルライブラリの管理方式は、設計システムを効率よく運用していくうえで重要であると考えられる。

この問題に対処するために、我々は統合化セルライブラリデータベースＹＩＬＤ（Ｙamaha Ｉntegrated cell Ｌibrary Ｄata base）を開発し、運用している。

ＹＩＬＤとは、数種類のプロセス各々が持つ数百種類のスタンダードセルに対する各ツール毎のライブラリ情報を統合化しデータベース化したもので、当社のスタンダードセル方式ＬＳＩ設計システムの中核となっている。

本稿ではまず従来のセルライブラリの管理上の問題点を述べる。

次に今回開発したＹＩＬＤのデータベースシステムとしての機能を説明する。

そして設計システムにおけるＹＩＬＤの運用形態、アプリケーションにおける利用形態を説明し、最後に今後の課題を述べる。

2．セルライブラリ管理上の諸問題

セルライブラリを管理するにあたり、第一章で述べたようなその種類の多さからくる煩雑さの他に、以下で述べるような運用上の問題も存在していたが、ＹＩＬＤの導入によりこれらを解決することができた。

従来のセルライブラリの管理方式では、各ツール毎のフォーマットで、各プロセス毎に全セルのライブラリを一つのファイルにまとめて、あるディレクトリに置いてあった。この方式だと各ツール毎、各プロセス毎にライブラリが独立しており、全体的な登録状況の把握等が困難だった。

ＹＩＬＤではツール別、プロセス別のセルライブラリを統合化することにより、全体的な管理が可能となった。

また、これはモラル上の問題ではあるが設計システムのユーザの規模が人数的にも、地理的にも広がってくると、その中にはこのマスタのライブラリファイルを自分のディレクトリにコピーしてきて、そのローカルなライブラリファイルに対し追加、修正をしながら使っている場合が見受けられた。

これに対しては、ネットリスト変換等のアプリケーションソフトの実行時に、ＹＩＬＤの中の最新のライブラリ定義を読み出し、ツールのフォーマットでセルライブラリを生成し、これを利用する方式とした。

ところでプロセスのセルライブラリの中で、特定のユーザにしか使われないセルの集合がある。これらはそのプロセスの標準的なセル集合以外に、自分達の回路設計用に特化したカスタムセルを開発しライブラリ化したものである。このような汎用性の無いセル集合を、標準的なセル集合と同一のファイル内で管理すると不都合なことがある。例えば一般のユーザにとってはツールの実行時に参照するセルライブラリが不必要に増大したり、またセルライブラリの保守の面で繁雑さが増大する。

そこでＹＩＬＤでは各プロセス毎に標準的なセル集合と、特定のユーザが使用するセル集合を別けて管理し、アプリケーションの実行時にこれらの組み合わせで参照できるようにした。

3．統合化セルライブラリデータベース ＹＩＬＤ

3．1　分類方式

ＹＩＬＤにおけるセルライブラリの分類方式として、図３－１に示す木構造を採用した。まずデザインルールによる分類項目『プロセス』があり、次にひとつのプロセスの中に用途の異なるセルの集合『グループ』の分類があり、またグループを構成する『セル』、そして各セルに対して『ツール』の分類がある。最後にそのツールの下に具体的なセルライブラリの定義データが登録してある。なおグループの分類方式としては、あるプロセスにおける標準的なセル集合のグループを考え、これをそのプロセスの『プライマリグループ』と呼ぶことにし、グループの名前はそれが含まれるプロセスの名前と同一とする。それ以外の、特定のユーザが使用するグループを、『スーパーポーズグループ』と呼ぶことにする。

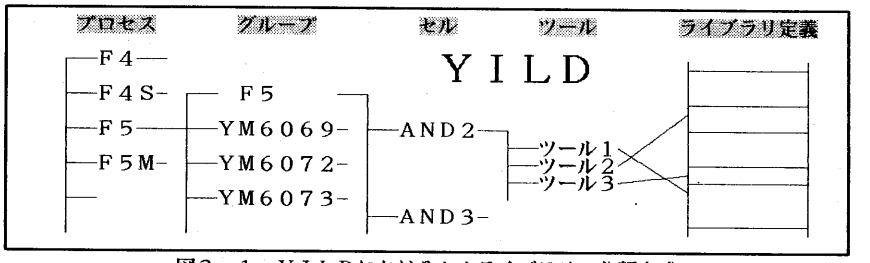


図３－１　ＹＩＬＤにおけるセルライブラリの分類方式

3．2　ＹＩＬＤの構成

上記の分類方式をインプリメントしたＹＩＬＤの構成を図３－２に示す。ＹＩＬＤは五つのダイレクトアクセスファイルから成る。これらは Header File, Process Table File, Group Table File, Cell Table File, Data File である。ここで Tool Table に相当するものとしては、Cell Table の１レコードの中を５バイトずつ区切って使用することで、各フィールドでツールの分類を表現している。Data File の各レコードはツール毎に固有の書式で登録され、ＹＩＬＤはそのセルライブラリの定義データの意味については関知しない。

なお Process, Group , Cell の各テーブル内の項目はアルファベット順にソートして登録してあり、検索の際の高速化をはかっている。

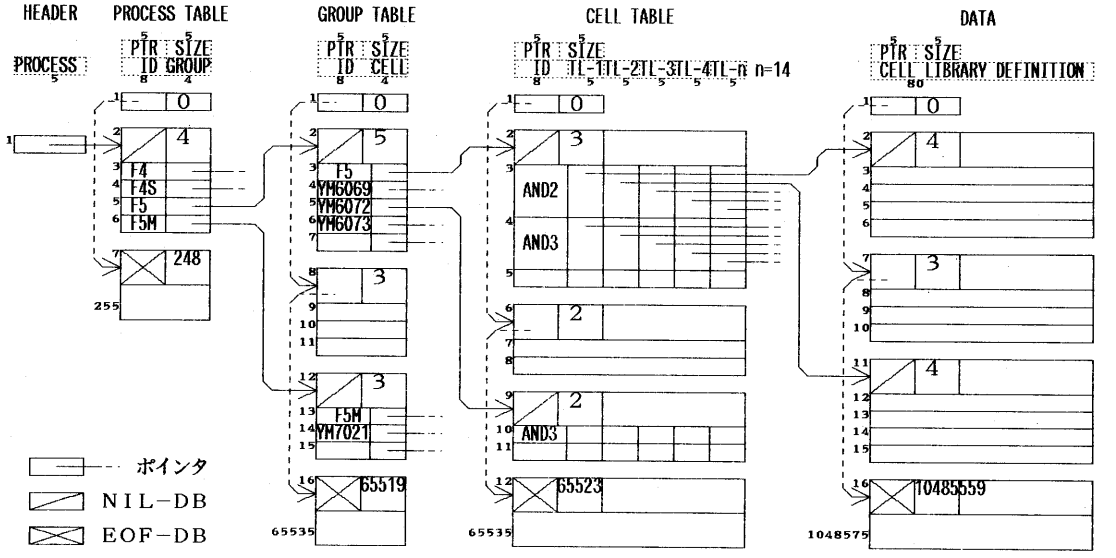


図３－２　ＹＩＬＤの構成

３．３　登録方式

ＹＩＬＤへのセルライブラリの登録は、以下に示す方式によりおこなわれる。

まずセルライブラリの『定義ファイル』を用意する。この定義ファイルはツール毎に様々な方法により得られる。例えば論理シミュレータ用ならば、ターゲットのシミュレータの持つプリミティブからなるネットリストをスケマティックキャプチャでセル単位に作成する。それらは数セル分まとめられ、ホストコンピュータへ転送される。あるいはセルの自動配置配線ツール用ならば、セル内部のレイアウト設計の際に、未使用のレイヤを利用してセルの外形寸法、端子位置、属性等を入力したデータから、必要な情報を抽出する。そしてそれらは同様にホストコンピュータへ転送される。

次にこれらの定義ファイルを、ツール毎のプリプロセッサに通し、ＹＩＬＤ登録用の『規格化ファイル』に変換する。

なおここでプリプロセッサを実行させる際に、プロセス名とグループ名を指定する。

このようにして得られた規格化ファイルの構成を、図３－３に示す。

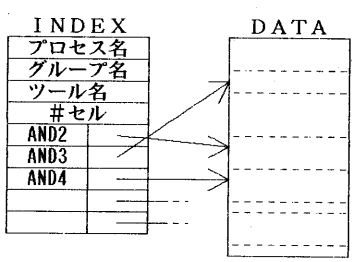


図３－３　規格化ファイル

そして全てのツールに共通の『ＹＩＬＤ登録プログラム』に、これらの規格化ファイルを入力する。ＹＩＬＤ登録プログラムは、規格化ファイルの中で指定されたプロセス、グループ、ツールの分類に従い、セル定義を１セル分を読みとっては、ＹＩＬＤの Cell Table へセル名を登録し、セル定義の数レコード分をそのままのイメージでＹＩＬＤの Data File へ登録する。

なおここで、そのセルが既に登録済みの場合には、古いデータは解放され、新しいデータが登録される。

３．４　参照方式

各アプリケーションソフトからのＹＩＬＤへの参照は、以下に示す方式によりおこなわれる。

各アプリケーションソフトの実行に先立ち、全ツール共通フォーマットである、ＹＩＬＤ参照用の『インデックスファイル』の生成がおこなわれる。

まずアプリケーションソフトの起動コマンドのパラメータの一部としてプロセス、グループの指定がおこなわれる。それらの指定に従って、全ツール共通の『ＹＩＬＤ参照プログラム』がＹＩＬＤ内を検索し、該当する Cell Table を得る。

そこでそのテーブルの各セルを表すレコードにおける、該当ツールのフィールドの内容、すなわちＹＩＬＤのData File へのポインタをセル名とともに書き出したものがインデックスファイルである。

ここでインデックスファイルの構造と、Data File との関係を図３－４に示す。

なおこのインデックスファイルは、各アプリケーションソフトの実行に先立ち作成され、その処理が終了すると削除される。

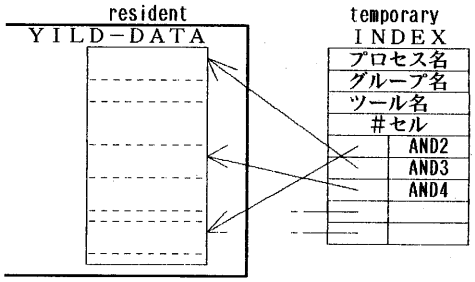


図３－４　インデックスファイルとＹＩＬＤの関係

この様にして得られたインデックスファイルを用いて、各アプリケーションソフトから直接ＹＩＬＤの Data File へリードアクセスをおこない、処理に必要なセルライブラリの定義データを得る。

ここで、プロセスＰ、グループＧ、ツールＴで指定される、ＹＩＬＤ上のセルライブラリの定義データの集合Ｌを、次のように表すものとする。

$$L = \{P, G, *, T\}$$

つまり先に述べたインデックスファイルは、この集合Ｌを表していると考えられる。

実は上で述べたインデックスファイルを生成するＹＩＬＤ参照プログラムの説明の中で、便宜上内容を一部単純化した箇所があり、実際には次のようになっている。

ＹＩＬＤ参照プログラムはプロセスＰ、グループＧの指定を受けてＹＩＬＤの検索をおこない、プロセスＰの中のグループＧのセルテーブルのみではなく、これとプロセスＰの中のグループＰのセルテーブルを組み合わせた内容のインデックスファイルを生成する。

この方法により２章の最後で述べたような、各プロセスの標準的なセル集合と、特定のユーザの使用するセル集合を組み合わせたセルライブラリの参照が可能となる。

なおこのセル集合の組み合わせ方法は、特定ユーザ用のグループＧ（スーパーポーズグループ）のセルを、標準セル集合のグループＰ（プライマリグループ）のセルに対し、優先して使用するものとする。

すなわち、例えばネットリストの中にＡＡというセルが使われていて、スーパーポーズグループとプライマリグループの双方にＡＡというライブラリセルが登録されている場合には、スーパーポーズグループの方を使用する。

そこでインデックスファイルの内容は、結局次の式で表されるものになる。

$$LP = \{P, P, *, T\}$$

$$LS = \{P, G, *, T\}$$

$$L = LS + LP \cdot \overline{LS}$$

これらの関係を、図３－５に例示する。

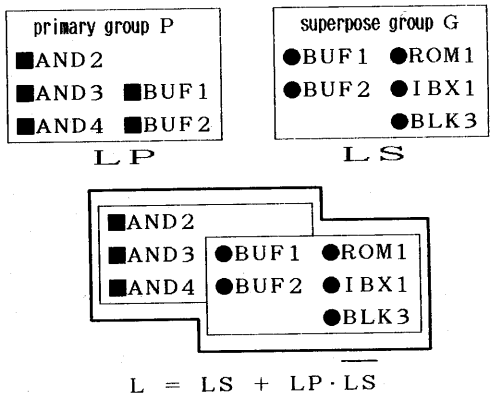


図3－5　インデックスファイルの内容

３．５　その他の基本機能

ＹＩＬＤには登録、参照の他に、以下に示すようないくつかの基本機能がある。

○削除
ライブラリ定義を、
ＹＩＬＤから削除する。

○サマリーリポート
ＹＩＬＤの登録状況や、セルライブラリの登録、更新日付の一覧表を出力する。

○ガーベジコレクション
各ファイルにテーブルを割当てたり削除する過程で生ずる、隣接した空き領域を連続化する。

○コンパクション
新規ＹＩＬＤ用ファイルに、空き領域を除いて、有効データをコピーする。
これは半年に一度程度、実行する。

○定義ファイル逆変換
バックアップ用に、ＹＩＬＤからプロセス、グループ、ツール単位で定義ファイルを生成する。
万一ＹＩＬＤがダメージを受けた場合には、ＹＩＬＤを初期化後これらのファイルを再登録すれば復旧可能である。

4．ＹＩＬＤの運用形態

4．1　登録作業のフロー

現在運用中のセルライブラリのＹＩＬＤへの登録作業について、そのフローを図４－１に示す。

主にパソコンＣＡＤ上で作成されたセルライブラリの定義ファイルは、ホストコンピュータへ送られた後に、ターゲットの各ツール毎にプリプロセッサを通し規格化ファイルに変換される。

次にこれらのファイルを、プロセス、グループを指定しながら、ＹＩＬＤ登録プログラムに入力し、ＹＩＬＤへ登録する。

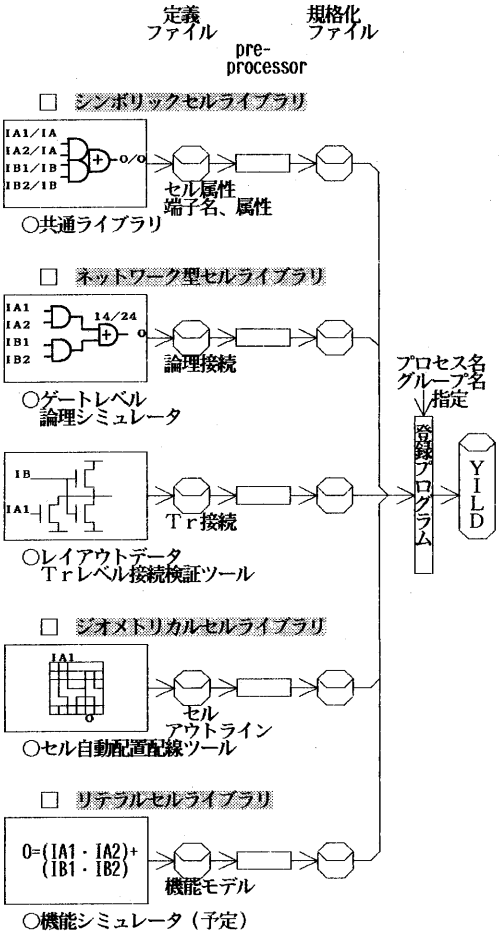


図4－1　ＹＩＬＤ登録作業フロー

4．2　アクセス形態

ＹＩＬＤへのアクセス形態は、以下のようになっている。

まず一般ユーザのＹＩＬＤへの参照は自由とする。

そしてＹＩＬＤの内容の書き替えを伴う登録作業は、グループのカテゴリにより次のように分かれている。

すなわち各プロセスのプライマリグループには、各ツールの管理者が登録をおこなうものとする。

一方スーパーポーズグループは、一般のユーザが登録許可の申請をおこないＹＩＬＤへの登録アクセス権を得た後に、自分で登録をおこなうものとする。

4．3　現在の登録状況

平成元年１月現在でのＹＩＬＤ登録状況を、表４－１に示す。

プロセス	グループ	セル数	ツール数	小計
Ｆ４	F4	188	1	188
	F4TEST	180	1	180
	GOMI	9	1	9
	YM3412	4	1	4
	YM3432	3	2	6
	YM3433	3	1	3
Ｆ４Ｓ	F4S	200	4	800
	TESTF4S	197	1	197
	YM3432	10	2	20
	YM7116	1	1	1
Ｆ５	F5	354	3	1062
	F5TEST	308	1	308
	TESTF5	51	1	51
	YM1804	3	1	3
	YM1813	6	3	18
	YM6000	29	1	29
	YM6069	4	2	8
	YM6072	1	2	2
	YM6073	3	2	6
	YM6093	7	2	14
	YM6096	1	3	3
	YM6601	1	3	3
Ｆ５Ｍ	F5M	444	4	1776
	TEST	443	2	886
	TESTF5M	369	1	369
			合　計	5946

表４－１　ＹＩＬＤ登録状況

5．アプリケーションにおける利用形態

アプリケーションソフトにおけるＹＩＬＤの利用形態のいくつかをみてみる。

5．1　論理シミュレータ用ネットリストフォーマッタ

スケマティックキャプチャで作成されたネットリストを、ある論理シミュレータＺ用に変換するアプリケーションにおいて、ＹＩＬＤがどのように利用されているかを説明する。

まず論理シミュレータＺ用のセルライブラリ定義データの一例を図５－１に示す。

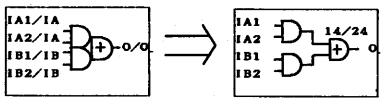


図5－1　Z用セルライブラリ定義データの例

このような内容のセルライブラリ定義データが、すでにＹＩＬＤ上に図５－２で示す構造で登録されているものとする。

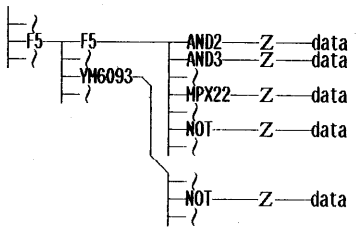
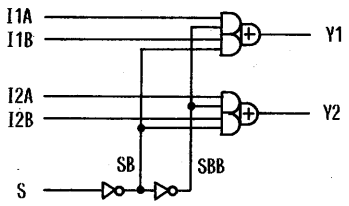


図5－2　ＹＩＬＤの分類例

そしてスケマティックキャプチャで図５－３に示す回路を設計し、そのネットリストファイルが得られたものとする。




```
ZJ157: Y1, Y2, I1A, I1B, I2A, I2B, S;
  S_1, NOT: SB, S;
  S_2, NOT: SBB, SB;
  S_3, MPX22: Y1, I1A, SBB, I1B, SB;
  S_4, MPX22: Y2, I2A, SBB, I2B, SB;
  EOM;
```

図5－3　回路図とそのネットリスト

ここで使用するＹＩＬＤ上のライブラリは、プロセスがＦ５、スーパーポーズグループはＹＭ６０９３とする。

そこで次に示すネットリスト変換コマンドにより、処理が開始される。

```
ZNET   F5   YM6093   NT6093
                       └ソースファイル名
                  ──グループ名
             ──プロセス名
        ──コマンド
```

以下に、処理の過程を説明する。

まずＹＩＬＤ参照用のインデックスファイルを作成する。これを式で表すと次のようになる。

ＬＰ＝〔F5, F5　　, ＊, Ｚ〕
ＬＳ＝〔F5, YM6093, ＊, Ｚ〕

$$L = LS + LP \cdot \overline{LS}$$

次にソースファイルを読み込みながら、ネットリストで使用されているセルの集合を求める。この例の場合

（　ＭＰＸ２２，　ＮＯＴ　）

となる。

そしてこれらの使用セルに対し、インデックスファイルを用いてＹＩＬＤから定義データを読みだし、シミュレータＺのフォーマットに変換してファイルに出力する。その後、ネットリストを変換し出力する。

このようにして得られた、論理シミュレータＺ用のネットリストファイルを、図５－４に示す。

```
$ YILD library cell( MPX22    )
.MACRO MPX22 O IA1 IA2 IB1 IB2
   X0000 .AND IA1 IA2
   X0001 .AND IB1 IB2
   O .OR 14 24 X0000 X0001
  .EOM
$ YILD library cell( NOT      )
.MACRO NOT ON I
   ON .INV 4 4 I
  .EOM
$ end of used cell set
.MACRO ZJ157 Y1 Y2 I1A I1B I2A I2B S
  <S_1 NOT SB S
  <S_2 NOT SBB SB
  <S_3 MPX22 Y1 I1A SBB I1B SB
  <S_4 MPX22 Y2 I2A SBB I2B SB
  .EOM
```

図5－4　Z用ネットリストファイル

5．2　複数の論理シミュレータにおける単一セルライブラリの利用

もともと論理シミュレータＺ用のプリミティブで構成されたＹＩＬＤ上のライブラリ定義（図５－１、図５－２参照）を、アプリケーションレベルで、それらのプリミティブを別のＶという論理シミュレータのプリミティブで再定義することにより、論理シミュレータＶ用のライブラリ定義としても使用している。この方法により、論理シミュレータＶ用のライブラリ定義を、新たにＹＩＬＤ上に登録せずに済んでいる。

実行例として、この方法で得られた図５－３の回路に対する論理シミュレータＶ用のネットリストを図５－５に示す。

```
/* YILD LIB. CELL( MPX22    ) */
module MPX22( O, IA1, IA2, IB1, IB2 );
  input IA1, IA2, IB1, IB2;  output O;
  supply1 vdd;  supply0 vss;
    VAND2 x0000 ( X0000, IA1, IA2 );
    VAND2 x0001 ( X0001, IB1, IB2 );
    VOR2  #(14,24) x0002 ( O, X0000, X0001 );
  endmodule
/* YILD LIB. CELL( NOT      ) */
module NOT( ON, I );
  input I;  output ON;
  supply1 vdd;  supply0 vss;
    VINV  #(4,4) x0000 ( ON, I );
  endmodule
/* end of used cell set */
module ZJ157( Y1, Y2, I1A, I1B, I2A, I2B, S );
  input  I1A, I1B, I2A, I2B, S;  output Y1, Y2;
  supply1 vdd;  supply0 vss;
    NOT s_1 ( SB, S );
    NOT s_2 ( SBB, SB );
    MPX22 s_3 ( Y1, I1A, SBB, I1B, SB );
    MPX22 s_4 ( Y2, I2A, SBB, I2B, SB );
  endmodule
```

図5－5　V用ネットリストファイル

5．3　一つのアプリケーションで二つのツールのライブラリを同時に参照

一つのアプリケーションソフトにおいて二つのインデックスファイルを用いて、二つのツールのライブラリ定義を同時に参照する場合がある。

例えばプロセスＰ、グループＧが指定されると、次式で示す内容のツールＴＡとツールＴＢ用のインデックスファイルが得られ、アプリケーションで同時に参照される。

$$LAP = 〔P,\ P,\ *,\ TA〕$$
$$LAS = 〔P,\ G,\ *,\ TA〕$$

$$LA = LAS + LAP \cdot \overline{LAS}$$

$$LBP = 〔P,\ P,\ *,\ TB〕$$
$$LBS = 〔P,\ G,\ *,\ TB〕$$

$$LB = LBS + LBP \cdot \overline{LBS}$$

5．4　ライブラリ更新時における新旧バージョンの使い分け

論理シミュレータ等のライブラリの検証では、実際にツールにかけて実行させてみる必要がある。そのためには、ライブラリがＹＩＬＤ上に登録されていなければならない。

既にＹＩＬＤ上に登録済みのセルライブラリを更新したい場合、それまでのセルライブラリはユーザが参照できる状態のままで、新しいセルライブラリの検証作業をおこなう必要がある。

そこで、新しいバージョンのセルライブラリをスーパーポーズグループに登録しておけば、アプリケーションソフト実行時のグループ指定により新旧のバージョンを使い分けることが可能である。

例えばプロセスＰ１のプライマリグループＰ１の中のいくつかのセルライブラリを更新する場合、更新したいセルライブラリをスーパーポーズグループＴＥＳＴに登録することで、次のように実行時に使い分けることができる。

```
command P1 P1    source (旧バージョン)

command P1 TEST source (新バージョン)
```

そして更新内容を検証後に、スーパーポーズグループＴＥＳＴの内容をプライマリグループＰ１に登録する。その後スーパーポーズグループＴＥＳＴを削除する。

6．今後の課題

現在、プラットフォームの違いから、スケマティックキャプチャのシンボルライブラリと、レイアウトデータ合成用ＣＡＤのレイアウトライブラリはＹＩＬＤ化されていない。

今後ネットワークレベルにＹＩＬＤの機能を拡張し、すべてのセルライブラリを統合化して管理していきたい。