社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2004－ＳＬＤＭ－114　(13)
2004／3／19

# 商用ソフトを用いないSFLベースのLSI試作の試行

飯田　佳洋†　　清水　尚彦††

† 東海大学大学院工学研究科 〒259-1292 神奈川県平塚市北金目 1117
†† 東海大学電子情報学部コミュニケーション工学科 〒259-1292 神奈川県平塚市北金目 1117
E-mail: †{3aepm001,nshimizu}@keyaki.cc.u-tokai.ac.jp

あらまし　我々は今までにSFLを用いたLSI設計とVerilogによるシミュレーションをシームレスに行う設計環境を提案、構築してきた。今回、さらにフリーなEDAツールであるAlliance VLSI CAD Systemを組合せ、商用ソフトを用いずにレイアウトを作成、VDECを利用したLSI試作を行っている。SFL、Verilog、Allianceを用いたLSI試作環境を提案し、この設計試行の詳細を説明する。

キーワード　EDA, VDEC, Alliance, SFL, Verilog

# A trial of SFL based LSI fabrication without commercial software

Yoshihiro IIDA† and Naohiko SHIMIZU††

† Graduate School of Engineering, Tokai University, 1117 Kitakaname, Hiratsuka, Kanagawa, 259-1292 Japan
†† Dept. Communications Engineering, Tokai University, 1117 Kitakaname, Hiratsuka, Kanagawa, 259-1292 Japan
E-mail: †{3aepm001,nshimizu}@keyaki.cc.u-tokai.ac.jp

**Abstract**　We have proposed LSI development environment based on SFL and Verilog. We propose a new development environment with SFL, Verilog, Alliance for layout and routing to fabricate real chips. The TEG developed with this environment is now under the fabrication with VDEC. In this paper, we describe the details of this development environment.

**Key words**　EDA, VDEC, Alliance, SFL, Verilog

## 1. はじめに

これまでに我々はSFL[1]によるHDL設計とこれをVerilogに変換してシミュレーションを行う開発環境を提案・構築してきた。これは高抽象度のRTL設計、Verilogシミュレーション、そしてFPGAへの実装をシームレスに行うことができるものである。今回我々はこれと共にAlliance VLSI CAD System[2]を用いてLSIチップレイアウトの自動生成、そしてVDECを利用したチップ試作を行った。これらは全て非商用ソフトを用いて開発し、様々なソフトをリンクさせて構築した。本報告ではSFL、Verilog、Allianceを用いたLSI試作の詳細を説明する。

## 2. SFL/Verilog開発環境

これまで我々はSFLの高抽象度による設計効率の良さに着目し、開発言語としてSFLを使用してきた。しかしそれは一般的なVerilog/VHDLを用いた開発環境とのギャップが大きく、実装までを考えた上では効率的とはいえなかった。そこでVerilogへのRTL変換ソフトウェアを開発し[3]、シミュレーション効率を向上、そして実装までをシームレスに行うことができるものとした。

## 3. Alliance VLSI CAD System

Alliance VLSI CAD Systemはフランスの Pierre et Marie Curie Universityにて開発されたライセンスフリーのVLSI設計統合開発環境である。AllianceはVHDLやEDIFネットリストを入力とし、GDS、CIFといったフォーマットでパターンレイアウトを出力することができる。また、論理圧縮コンパイラを持っており、回路規模/出力遅延の最適化を行うことができる。Allianceには90万弱ゲート規模のマイクロプロセッサの開発実績もあるが、元来CMP等EU圏内での使用を想定しているため、日本のVDECや米国のMOSISでの試作実績は無かった。そこで我々はAllianceを用いて設計し、VDECで試作を行う環境の構築を目指した。

Allianceを用いた設計・LSI試作を日本で行う上での問題点は、VDECやMOSISで利用可能な企業のデザインルールファイルがAllianceには無いため独自で作らなければならない点である。また、入力可能なVHDLはIEEE標準互換ではないた

め、他のツールとの連携に問題があった。これにより、Alliance の VHDL 設計環境で数百・数千ゲート規模の設計を VLSI 設計教育に導入するには難易度が高く、実現性に乏しいという問題があった。

## 4. 提案する設計環境

### 4.1 設計環境の構築

これまでの SFL/Verilog 環境では、Verilog シミュレーション用に Icarus Verilog [4] を用いてきた。Icarus Verilog は Verilog コンパイラとしてネットリストを生成する機能を持っており、我々はこれに Alliance で用いる sxlib (注1) に対応した EDIF ネットリストを生成するドライバを開発し、Verilog 記述から EDIF へ変換を行うことで SFL による設計から Alliance への入力を可能とした。

Alliance は多くの Alliance 内部ツールに対する総称であり、今回我々は表 1、表 2、表 3 に示す記述フォーマット、ツールを用いて論理設計、論理合成、論理圧縮、自動配置配線を行った。

表 1 使用するファイルフォーマット

| 記述ファイル | 記述レベル | 拡張子 | 使用ツール |
|---|---|---|---|
| SFL | RTL | .sfl | sfl2vl |
| Verilog | RTL | .v | iverilog |
| EDIF | ネットリスト | .edi | x2y |
| VHDL Structure | ネットリスト | .vst | flatbeh, ocp |
| VHDL Behaviour | ビヘイビア | .vbe | boom, boog |
| Alliance 仮想配置 | Alliance 内部 | .ap | nero, s2r |
| CIF, Alliance 実配置 | CIF テキスト | .cif | dreal |
| GDS, Alliance 実配置 | GDS データ | .gds | dreal |
| パッド配置 | Alliance 内部 | .rin | ring |

表 2 使用したツール

| ツール名 | 入力 | 出力 | 機能 |
|---|---|---|---|
| sfl2vl | .sfl | .v | Verilog 記述への変換 |
| iverilog | .v | .edi | EDIF ネットリストの論理合成 |

表 3 使用した Alliance 付属ツール

| ツール名 | 入力 | 出力 | 機能 |
|---|---|---|---|
| x2y | .edi | .vst | vst ネットリストへの変換 |
| flatbeh | .vst | .vbe | vst をビヘイビア記述に変換 |
| boom | .vbe | .vbe | 論理圧縮 |
| boog | .vbe | .vst | vst への合成 |
| ocp | .vst | .ap | セルの自動配置 |
| nero | .ap | .ap | 自動配線 |
| ring | .ap,.vst,.rin | .ap | 電源リング、パッドの自動配置 |
| s2r | .ap | .cif, .gds | 仮想配置から実配置を生成 |
| graal | .ap | - | ap ファイルエディタ・ビューア |
| dreal | .cif, .gds | - | CIF, GDS ファイルエディタ・ビューア |

回路の記述は SFL で行い、Icarus Verilog(iverilog) にてシミュレーション・EDIF ネットリストの合成を行う。これを Alliance 付属の x2y を用いて VHDL Structure ネットリスト記述へ変換し、論理圧縮を行う場合は flatbeh にて VHDL ネットリスト記述を VHDL ビヘイビア記述に変換後に boom、boog を用いる。ocp は圧縮後もしくは未圧縮の VHDL ネットリストから sxlib の仮想配置を行い、セル・入出力ピンの自動配置を行う。そして nero を用いてレイアウト内の自動配線を行う。ここで、設計した回路のレイアウトと入出力パッドとの接続を VHDL Structure ネットリストで記述し、またパッドの配置順序を指定する .rin ファイルを作成し、ring を用いてパッドを含めたレイアウト全体の仮想配置を行う。

### 4.2 デザインルールの記述

ここまではプロセスデザインルールに依存しないが、Alliance では仮想配置から実配置を生成するときに RDS と呼ばれる .rds ファイルにてデザインルールを細かく指定する。表 4 に今回作成した RDS ファイルのパラメータの概要を示す。

これらのパラメータを、利用するプロセスのデザインルールに合わせて記述した。以下に主なパラメータの説明を記す。

PHYSICAL_GRID は実配置の最小グリッドを指定し、各レイヤの最小幅をこれよりも細かくしたときにエラーとする。LAMBDA は実配置の大きさの基準値であり、Alliance では 1 セルの幅が 5 λ 単位、高さが 50 λ 単位となっているため、λ が小さい程面積が小さくなる。1 セルの大きさを小さくしても後述する各レイヤの最小幅は別に指定するため、セル内の最小間隔等ルールを満たす最小の λ を設定した。設定した λ を用いたインバータセルを図 1 に示す。

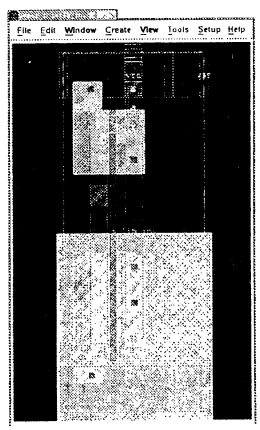


図 1 最適値の λ を適用したインバータセル

MBK_TO_RDS_SEGMENT は RDS ファイルの主な項目であり、λ を基準とした各レイヤの最小幅を設定する。ここでは例えばゲート上のポリシリコンとセル内配線用ポリシリコンとを区別するように用途による最小幅を設定する。MBK_TO_RDS_VIA ではメタル 1–VIA– メタル 2 というようなレイヤ間の接続ルールを設定する。

S2R_POST_TREAT、S2R_OVERSIZE_DENOTCH、S2R_MINIMUM_LAYER_WIDTH では、同レイヤの間隔が狭い時の結合に関する設定を行う。レイヤの結合を行う最大間隔と最小幅を指定し、最

(注1) : sxlib は Alliance 付属のサブミクロンプロセス用のスケーラブル CMOS ライブラリである。

小間隔のデザインルールエラーを無くすことができる。これは最小間隔を指定するわけではないので、異電位の配線など結合を行わないレイヤに対する最小間隔エラーは $\lambda$ を大きくすることでのみ解決を行う。

CIF_LAYER、GDS_LAYER では出力する CIF または GDS のレイヤ名の指定、そして Alliance 処理系が使うレイヤとの対応関係を記述する。

## 5. Test Element Group の試作

### 5.1 TEG の設計

Alliance を用いた自動配置配線レイアウトの評価を行うために VDEC を通してローム社 0.35 $\mu$ m プロセスを用いて TEG の試作を行った。TEG に実装した回路を表 5 に示す。まず sxlib には基本セルと複合ゲートセルがあり、基本セル全ての動作確認用回路を実装、動作周波数の性能評価のための 30 段インバータ+起動用 NAND のリングオシレータ、基本セルのみを用いた小規模順序回路として 4 ビットカウンタを実装した。また、複合ゲートも含めた中規模順序回路として AES 暗号用 Sbox 回路と RS-232C コントローラを実装し入出力を含めた順序回路のテスト回路とした。そして入出力にクロック入力の異なる 1 段ずつのラッチを用い、Alliance が生成したクロックリングによるクロックスキューの評価回路を実装した。

これらの回路は全て SFL で記述し、Alliance で配置配線を行ったレイアウトを図 2 に示す。

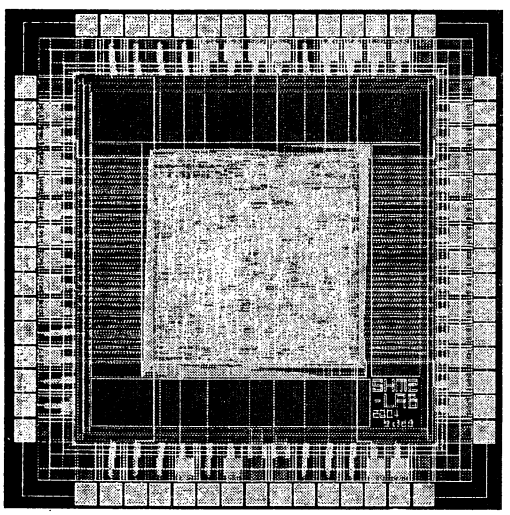

図 2 TEG チップのレイアウト - dreal で表示

### 5.2 ライブラリの修正

Alliance には sxlib の他にパッドライブラリ、データパスライブラリや ROM/RAM ライブラリ等が付属されており、今回は sxlib と共にパッドライブラリを使用いた。しかしパッドはチップ組み立てベンダに依存するため、Alliance 標準のパッドライブラリを修正した。修正を行ったパッドライブラリは、I/O 用 VDD、I/O 用 VSS、コア VDD、コア VSS、クロックパッド、及び I/O パッドである。

### 5.3 Alliance 生成レイアウトの評価

今回は Alliance でレイアウトを生成した回路をはじめて VDEC を通して試作するので、生成レイアウトの確認のため Alliance で生成したレイアウトを hspice 及び Cadence 社 icfb、Dracula を用いて評価を行った。

その結果デザインルール違反が 10 件程発見され、Alliance 用に記述したデザインルールファイル RDS ファイルの修正、パッドライブラリの修正を行った。その後 SFL からの論理合成、論理圧縮、自動配置配線の結果を hspice で動作確認、さらに Dracula でデザインルールエラーが無いことを確認した。設計した TEG を Virtuoso で表示したものを図 3 に示す。

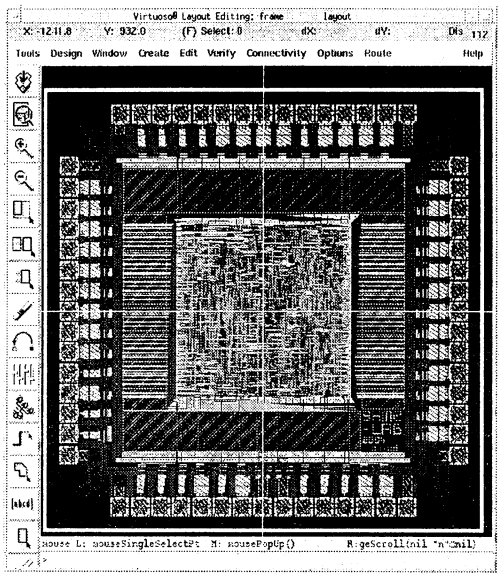


図 3 TEG チップのレイアウト - Virtuoso で表示

## 6. 評　　価

ここでは Alliance を用いた設計全体を通した評価を述べる。

### 6.1 レイアウトの生成

まずデザインルールの適用は、前節で説明したように RDS ファイルの作成のみでデザインルールエラーの無いレイアウトを生成することができる。しかしながら Alliance 開発者陣も言うように電源・クロックリング及びパッド自動配置を行う ring エンジンは柔軟性に乏しく、パッドライブラリをチップ組み立てベンダごとに修正しなければならない。また、斜め配線が未対応であるため、電源リングのコーナーが直角になってしまう点が問題である。

### 6.2 hspice との連携

hspice との連携に関しては、Alliance で生成する CIF においてラベルの定義文 4N を 94 に置換し、コーナーのラベルを削除、そして電源電圧ラベル VDD、VSS をそれぞれ VDD:P、VSS:G に置換することで VHDL Structure ネットリストで指定した

表 4　RDS ファイルのパラメータの概要

| パラメータ | 内容 |
|---|---|
| PHYSICAL_GRID | 最小グリッドの指定 [μ m] |
| LAMBDA | 実配置の大きさの基準値 [μ m] |
| MBK_TO_RDS_SEGMENT | 各レイヤの最小幅のルールの設定 |
| MBK_TO_RDS_CONNECTOR | ライブラリ間を接続するコネクタの大きさ |
| MBK_TO_RDS_REFERENCE | 自動配線を行う基準点の大きさ |
| MBK_TO_RDS_VIA | レイヤ-VIA-レイヤにおけるの各レイヤのルール |
| MBK_TO_RDS_BIGVIA_HOLE | 電源リング等で使う広い面積の配線レイヤ間の VIA のルール |
| MBK_TO_RDS_BIGVIA_METAL | 電源リング等で使う広い面積の配線レイヤ間のメタルのルール |
| S2R_POST_TREAT | 同レイヤの間隔が狭いときに結合するかを指定 |
| S2R_OVERSIZE_DENOTCH | 同レイヤの間隔が狭いときに結合する間隔を指定 |
| S2R_MINIMUM_LAYER_WIDTH | 同レイヤの間隔が狭いときに結合する最小幅を指定 |
| CIF_LAYER | RDS ファイル内でのレイヤと出力する CIF のレイヤ名とのリンク |
| GDS_LAYER | RDS ファイル内でのレイヤと出力する GDS のレイヤ名とのリンク |

表 5　TEG チップに実装した回路

| 回路 | テスト内容 |
|---|---|
| sxlib の基本 14 セル | 基本セルライブラリの動作確認 |
| リングオシレータ | 動作周波数の性能評価 |
| 4 ビットカウンタ | 基本セルのみの小規模順序回路の動作確認 |
| AES 暗号回路用 Sbox 順序回路 | 中規模順序回路の動作確認 |
| RS-232C コントローラ | Sbox の動作確認、入出力付き順序回路の動作確認 |
| 異クロック入出力ラッチ | Alliance 生成クロックリングによるクロックスキューの確認 |

入出力ピンのラベルを用いてシミュレーションを行うことができる。

### 6.3　論理圧縮の効果

Alliance の論理圧縮プログラム boom ではビヘイビア記述に変換した VHDL 記述を BDD を用いた圧縮エンジンや Boolean 圧縮を行うことで最適化を行うことができ、TEG 用に設計した AES 暗号用 Sbox 回路では未圧縮の状態と比べ 20%強の回路規模の削減、出力遅延を 30%削減した。また、圧縮したビヘイビア記述を boog によって複合ゲートへと合成することで更に回路規模、出力遅延の削減を行うことができる。boog は sxlib への合成後の回路規模及び出力遅延を評価することができ、これを用いて次の評価を行った。

Alliance 付属の論理圧縮ツール boom の性能評価のため、TEG 用に設計した AES 暗号用 Sbox 回路に対して論理圧縮の有無で回路規模を比較すると、圧縮後の結果が 20%弱使用セル数が少なかった。また、AES 暗号用 Sbox は森岡氏らによる合成体アルゴリズムを用い、オリジナルの実装とを比較すると、森岡氏らによる Sbox [5] が 354 ゲートであるので、boom による Boolean 圧縮は効果が高いといえる。これらの回路規模、出力遅延のの比較を表 6 に示す。出力遅延に関してはプロセスの違いがあるため比較はできないが、回路規模に関しては boom による論理圧縮と boog による複合ゲートへの合成が大きく効いている。

### 6.4　開発効率

Alliance のツールの中で最も処理時間を要するものは nero による自動配線であり、ocp による自動配置で指定するセル間のマージン設定によって大きく異なった。設計した TEG は約

表 6　論理圧縮の効果

| 回路 | 回路規模 | 出力遅延 | 備考 |
|---|---|---|---|
| boom 圧縮前 | 344 セル | 14.338[ns] | 0.35 μ m プロセス |
| boom 圧縮後 | 280 セル | 10.730[ns] | 0.35 μ m プロセス |
| 森岡氏らの Sbox | 354 ゲート | 2.19[ns] | 0.13 μ m プロセス |

1500 セルであり、この回路規模のレベルでは ocp の標準マージン 0.2 では自動配線することができず、マージン 0.6 で数分の処理時間を要した(注2)。

## 7.　本開発環境の実行例

ここでは、提案した開発環境を用いた簡単な回路の設計例を説明する。

図 4 に示す 4 ビットカウンタ回路の SFL 記述を sfl2vl で Verilog へ変換する。

```
sfl2vl test.sfl -icarus↵
```

次に出力された図 5 ファイルを Icarus Verilog を用いて EDIF ネットリストを次のように生成する。

```
iverilog -tfpga -parch=sxlib test.v -o test.edi↵
```

この EDIF ネットリストを Alliance のネットリストコンバータ x2y を用いて VST へ変換する。Alliance では入出力ファイルの形式を環境変数で設定し、プログラム実行時には拡張子の指定を行わない。

```
x2y edi vst test test↵
```

ここで論理圧縮を行う場合は、VST ファイルを VBE に変換

(注2)：Intel(R) Pentium(R) III CPU family 1133MHz 環境下にて。

後に圧縮を行い、sxlib 用にゲート圧縮を行い再度 VST を合成する。

```
flatbeh test test↵
boom -nT test test↵
boog test↵
```

boom、boog では回路規模:出力遅延の割合を変えて最適化を行うことができる。また、boog は合成した回路で使用したセルの内訳と面積、出力遅延を評価する。得られた評価を図 6 に示す。この段階で論理圧縮された VST が合成され、次に配置配線を行う。ocp でセルの自動配置と入出力ピンの生成を行い、nero でセル間の配線を行う。

```
ocp -ring test test↵
nero -V test test↵
```

配置配線後の仮想配置を s2r を用いて実配置を生成する。

```
s2r -nr test↵
```

実配置を行った CIF ファイルを dreal で表示させたレイアウトを図 7 に示す。

電源・クロックリング、パッドの自動配置配線を行う場合は s2r の前に ring を用い、パッドと設計した回路との接続情報を記述したファイル (ここでは top.vst) と、パッドの配置順序を記したファイル top.rin を用意し、次のように実行する。.rin ファイルの記述例を図 8 に示す。

```
ring top top↵
```

ここでパッド等を含めた仮想配置が生成され、実配置のレイアウトのファイルフォーマット (CIF もしくは GDS) を環境変数で設定し、

```
s2r -nr top↵
```

と実行して最終的なレイアウトデータが出力される。

```
module test {
  output f<4>;

  reg_wr r<4>;

  par {
    r := r + 0x1;
    f = r;
  }
}
```

図 4　test.sfl

## 8. ま と め

### 8.1 商用ソフトを用いない設計

商用ソフトを用いない設計の利点として、ライセンス費用を必要としない、機能拡張が容易、ソフトの導入・評価が容易、であることが挙げられる。今回我々は Icarus Verilog に対し sxlib 対応 EDIF 合成機能を追加し、複数のツールの連携を行った。今回は Alliance の生成レイアウトのデザインルール違反の有無

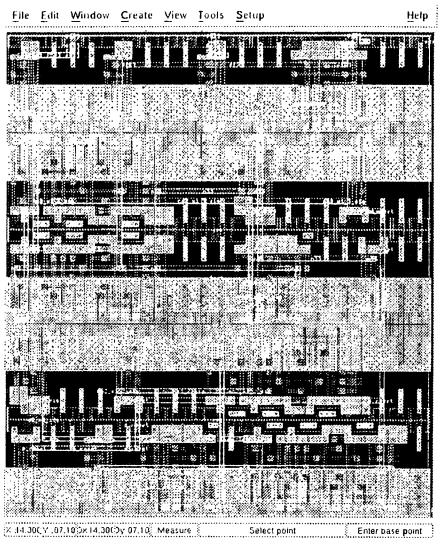


図 7　4 ビットカウンタ回路の自動配置配線後のレイアウト

```
north( p_vdde  p_vsse       )
east ( p_f0    p_vddi  p_f1 )
south( p_reset m_clock      )
west ( p_f2    p_vssi  p_f3 )
```

図 8　.rin ファイルの記述例

の評価を行うために Casence 社のツールを使用したが、レイアウトの生成まで全てを商用ソフトを用いずに設計を行った。

### 8.2 SFL、Verilog、Alliance の連携

提案した開発環境では高抽象度の RTL 記述からシームレスにシミュレーション、レイアウト生成を行うことができる。さらに hspice シミュレーションを行うまで、人手による作業を SFL 記述のみとした自動合成環境を構築した。また、論理の最適化も可能であり、回路規模・出力遅延のトレードオフを行うことができる。

現段階では小・中規模の LSI 設計において、すべてフリーソフトによる開発を行うことができることを実証した。フリーであるだけでなく、これにより LSI 設計の難易度を低くすることができ、VLSI 設計教育にも貢献できるものと考えている。

### 8.3 今後の課題

設計した TEG チップは VDEC を通して試作中であり、試作完了次第動作確認等の評価を行う。

現在の設計ではチップレイアウト全体の配置配線を一度に自動生成するため、数千・数万ゲート規模では処理時間が莫大になる。今後はフロアプラン設計を行い各ブロック毎にそれぞれ自動配置配線を行い、その後にブロック間の配線を自動で行う方法についてを模索し、大規模な開発を行えるよう環境を構築、そしてマイクロプロセッサ等の大規模な LSI 試作を行う予定である。

```
module test ( p_reset , m_clock , f );
 input p_reset; m_clock;
   reg [3:0] r;

output [3:0] f;
   assign f = r;
always @(posedge m_clock  )
  begin
  r <= (p_reset)? 4'b0000: (r)+(4'b0001);
end
/* The sfl2vl by Naohiko Shimizu generated this module. */
endmodule
```

図 5　test.v

```
Quick estimated critical path (no warranty)...1890 ps from 'n33' to 'n30'
Quick estimated area (with over-cell routing)...35250 lambda
                                                          Details...
        buf_x2: 4
        sff1_x4: 4
        no2_x1: 4
        nxr2_x1: 3
        a2_x2: 2
        Total: 17
```

図 6　boog による回路評価 (抜粋)

また、Alliance 付属のデザインルールチェッカ druc のみを用いたレイアウトの評価を行い、完全な非商用ソフト環境での LSI 試作を目指す。Alliance 付属 ring の電源・クロックリングに関しても設計の柔軟性を持たせ、斜め配線への対応など機能の向上も提案していきたい。

さらに、SFL から VHDL への変換ツールを評価中であり、Icarus Verilog を用いない合成手法を構築していく予定である。他ツールとして HP 社の Tsutsuji [6] を起源とするスペースソフト社の SLDS [7] のネットリストと SFL との相互変換機能を開発済みでありデータパス系の合成に SLDS を利用することでさらなる効率的な LSI 開発が可能となる。

## 謝　辞

本研究は東京大学大規模集積システム設計教育研究センターを通し、ケイデンス株式会社の協力で行われたものである。

## 文　献

[1] http://www.onlab.ntt.co.jp/mn/parthenon/pack.html
[2] http://www-asim.lip6.fr/recherche/alliance/
[3] Naohiko SHIMIZU, "Design of *sfl2vl*:SFL to Verilog Converter Based on an LR-Parser," IEICE Trans. Fundamentals., vol.E86-A, no.12, pp3225-3229, 2003
[4] http://www.icarus.com/
[5] 森岡澄夫, "数学いらず AES 暗号 SubBytes 設計ガイド", Design Wave Magazine, pp.152-157, 2004 年 1 月号
[6] W. Bruce Culbertson, Toshiki Osame, Yoshisuke Otsuru, J. Barry Shackleford, Motoo Tanaka, "The HP Tsutsuji Logic Synthesis System", Hewlett-Packard Journal, August 1993
[7] http://www.spacesoft.co.jp/EDA/slds/
[8] Mead, Conway, "Introduction to VLSI systems", Addison Wesley, 1977
[9] Neil H. E. Weste, K. Exhraghian, "Principles of CMOS VLSI design", Addison Wesley
[10] *sfl2vl*, http://shimizu-lab.dt.u-tokai.ac.jp/
[11] 石黒健史, "Alliance の Scalable CMOS Library によるチップ試作の研究", 東海大学工学部卒業研究論文, 2003