マイクロコンピュータ 48－12
計算機アーキテクチャ 69－12
（１９８８．１．２１）

『ＡＳＩＣ手法を活用したシステムＶＬＳＩの開発』

白石 肇、　西尾 誠一、　伊藤 聡
（株）東芝 半導体システム技術センター、総合研究所、半導体技術研究所

本報告では、ユーザ・システム毎の高度な目的に適したシステムＶＬＳＩを比較的短期間に実現する開発手順を提案する。 我々半導体製品開発者は、常により高度で有用な機能をデバイスの中により多く取込もうと努力し続けてきた。一方、高付加価値のシステムＶＬＳＩに対するユーザの需要は非常に高い。しかしながら、従来の開発方法のみでそれを実現するのは難しい。そこで、我々は、短期間でユーザ応用システムの仕様に合ったＬＳＩを提供出来るＡＳＩＣ（Application Specific Integrated Circuit）の開発手法をシステムＶＬＳＩの開発に応用する事を試みた。特に、機能記述言語の使用、機能検証の利用、自動論理合成の適用、階層レイアウトの活用等が有効であった。これらの活用方法と開発手順の事例を示し、更には、概念設計も含めた全体のシステムＶＬＳＩ開発手順と今後の方向につい提案したい。

THE DEVELOPMENT PROCEDURE FOR SYSTEM VLSI BY APPLYING ASIC METHOD

Hajime SHIRAISHI, Seiichi NISHIO, Satoshi ITOH

Semiconductor System Engineering Center;
R & D Center;
Semiconductor Device Engineering Laboratory;
,Toshiba Corporation
580-1, Horikawa-cho, Saiwai-ku, Kawasaki, Kanagawa, 210 Japan

We propose a development procedure which we can make some VLSIs applied to the each object of high level user-systems in shorter period. We, semiconductor device developer, have been making efforts to put higher, more effective function into devices as much as we can. On the other side,many users have a great demand for very effective system oriented VLSI devices. However, we could not develop such devices without new challenge. So, we tried applying an extended ASIC(Application specific Integrated Circuit)developing method by which we can provide some LSIs applied to each user-systems specification.Especially, we found it very useful to write in functional description language, to apply function simulator, automatic logic synthesis system and hierarchical layout system. On this report, we will show an example of application method for these support system and development procedure. We would also like to propose whole development procedure for system oriented VLSI including conceptual design from now on.

１．　はじめに　（市場からの期待）

システム指向、ソフトウェア指向のアーキテクチァを持つＶＬＳＩの需要が急増している。

以前は、品質の良い、出来るだけ標準的な製品を安く、大量に生産しユーザに供給する事がメーカの使命であった。特に、半導体業界ではその傾向が強かった。即ち、民生用製品群や産業用制御機器群等を中心にひととおりの要求が満たされた現在、より高度で機能的かつ個性的な商品が求められている。別の表現をすれば、ユーザが必要なものを、必要なときに、必要なだけ作れなければならないと言うことになる。メーカは、業界ごとの、或いは、目的とする用途毎に最適な商品をユーザに提供出来るようになる必要がある。即ち、システム毎に対応して行くことになる。これは、半導体製造業にとって広い意味のＡＳＩＣ製品が、その重要度を増すことになる。

ユーザ・システムの行動を直接、柔軟に表現した（システム指向、ソフトウェア指向のアーキテクチァ）ＶＬＳＩが多く求められている。例えば、ＦＡ機器、ＯＡ機器において、今や１６ビット・マイクロ・コンピュータの７割近くはシステム埋め込み形とされており、ソフトウェアの面では、流通ソフトよりもむしろシステム制御を直接に表したものが、またハードウェアの面では、汎用のＭＰＵよりはむしろ周辺Ｉ／Ｏポートや特殊Ｉ／Ｏ回路も一体となったＭＣＵが強く求められている。ＭＰＵを使っている場合でも、周辺回路はほとんど目的に合ったＡＳＩＣ化がなされている。最も需要が大きいシステムＶＬＳＩとして、画像処理用、情報通信用、データベース用（ファイル・サーバを含む）、知識処理用等が考えられる。これらに共通している点は、複雑だが体系的なプロトコルを持ちしっかりしたデータ構造を必要としていることである。しかるに、専用の高機能回路ブロックを用いずに、全てを、従来の汎用アーキテクチァのみで実現しようとすれば、低いレベルの回路ブロックを直接利用する機械語でプログラミングをして、高機能を実現しようとするため、その記述効率は低く、性能の飛躍は期待が薄くなる。これらの観点からも、応用システム指向のＶＬＳＩに望むところは大きい。

２．　ＶＬＳＩ作りに期待される事

いま最も期待されているのは、多くの異なるユーザの要求仕様毎に合わせたＶＬＳＩを短期間に作ることや、逆にユーザの需要を引き出す優れたシステム・アーキテクチァを創造し、それを実際のＶＬＳＩの形で実現する事である。

新たな飛躍を引き起こす市場の需要動向を述べてきたが、ここで半導体メーカとして工夫して、準備しておかなければならないことを提案したい。後述するシステムＶＬＳＩ開発の実際例で、ＶＬＳＩ　ＣＡＤをベースとした開発手順と開発戦略を示す。

開発工程は幾つかの大きなブロックに分けられる。工程の下流から、ＶＬＳＩのユーザ・システム評価・検証、ＶＬＳＩチップ評価・検証、試作、マスク製作、ＶＬＳＩフロアプラン＆マスク設計、回路設計・検証、論理設計・検証、詳細機能設計・検証、基本設計・検証、概念設計・検証、外部要求仕様設計・検証等に分割して考える事ができる。

これらの局面（フェーズ）のうち、従来は論理検証や、マスク製作等の局面が自動化され実施されていた。今回示す実施例は、更に自動論理合成（論理設計・製図）と詳細機能記述・検証、ＶＬＳＩフロアプラン＆マスク設計が加わった。従って、今回の開発手順実行によって詳細機能記述以降の工程は、かなり効率向上が約束されよう。合理化は、常に新しい問題点を浮き彫りにする。即ち、下流工程の改善によって詳細機能記述するための入力としての詳細機能設計手順の明確化、基本設計・検証、概念設計・検証、外部要求仕様設計・検証、等上流の諸局面が次の課題となってくる。ユーザの目的にあった高性能・高機能のシステムＶＬＳＩの実現に、１００万トランジスタ位の素子数がどうしても必要である。この事から、プロセスももっと進歩して、０．６ミクロン幅ぐらいにならなければ思い切った良いシステム設計は望めない。また現在のＶＬＳＩフロアプラン＆マスク設計の戦略の飛躍的改善も必要となる。ＣＡＤ、ＣＡＥなどの設計支援ツールは、ゆっくりだが確実に進歩している。しかしながら、一番大きな問題は、仕事の進め方や考え方といった、いわゆるカルチァ（風土、習慣）を環境の改善と同期をとりながら確実に変革していく必要があると言うことである。ソフトウェア工学で危機を救うために実施されつつある主要な概念とその実現手法である構造化分析・設計が、またＶＬＳＩの開発でも今後最も必要なアプローチとなっていくものと考えている。

これらの諸問題も多くあるが、システム・アーキテクト、システム設計者、応用技術者が自ら進んでユーザの要求仕様を満たし、また反対にユーザの需要を引き出す目的指向の優れたシステムＶＬＳＩの開発手順と、概念の導入に力をいれることが大切である。

３．　実例

今回の開発事例で最も効率向上に寄与したＣＡＤサブシステムの一つとして、自動論理合成に焦点を絞って以下に報告したい。更に、実際にＡＳＩＣ手法を活用したシステムＶＬＳＩの開発事例を紹介し、その手順と活用戦略をのべる。

（１）自動論理合成システム

本節では、自動論理合成システム[1]の概要と今後の方向付けについて述べる。本自動論理合成システムの構成を図１に示す。

本自動論理合成システムは、ハードウェア設計言語Ｈ² ＤＬ（Hierarchical Hardware Design Language)の内部仕様記述[2]と呼ぶレジスタ転送レベルの言語で書かれたＬＳＩの機能仕様を入力し、予めライブラリに登録されている回路素子（セル）からなる論理回路を自動的に合成するシステムである。処理手順としては、まずトランスレータが、Ｈ² ＤＬ内部仕様記述に一対一に対応した形の初期の回路構造（ＵＮＥＴ）を生成する。ここでの回路は、ＡＤＤ等の関数は、そのまま一つの回路構成要素となっており、またゲート類もファンイン数、ファンアウト数に制限がない等、実際のセルとは対応しない抽象的なものである。なお、ＵＮＥＴは、このような抽象レベルから具体的なセルレベルまでを統一的に扱えるようにした回路表現である。続いて、論理合成エンジンが、ルールベースに蓄えられた設計知識に基づき、展開（関数をゲートの組み合わせに展開する等）、最適化（論理的に冗長な部分を除去する）、セル割り付け（ライブラリに登録されているセルを割り付け、ファンアウト調整を行う等）の処理を行う。最後に、ＵＮＥＴ接続記述変換が、論理シミュレータやレイアウトの入力となる接続記述を生成し、合成の処理を完了する。また、この他に、合成処理の任意の時点での回路状態を図面化するツール、及び、合成された回路のマニュアル修正を可能とするツールも用意されている。

現在本自動論理合成システムは、次節で紹介する３２ビットグラフィック専用プロセッサを始めとする幾つかの実製品

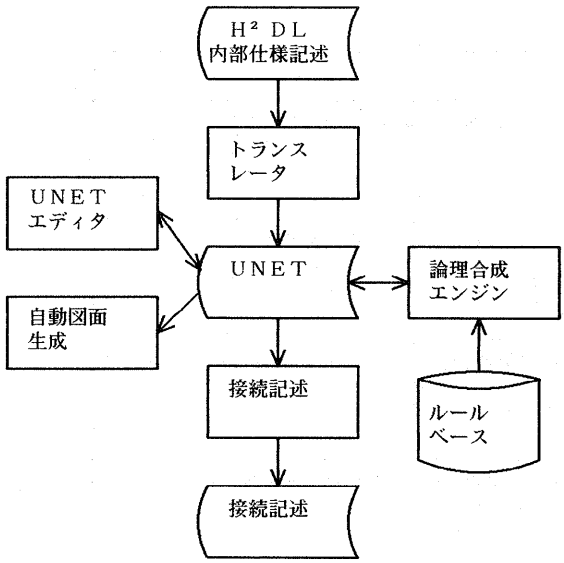

図１．　自動論理合成システムの構成

の開発に利用されており、論理設計の効率向上に大きく寄与している。今後の課題としては、タイミング設計のサポート機能の強化等により合成回路の質の向上を図ると共に、合成過程への設計者の介入をより容易にすることが重要と考えている。後者の一例としては、サブモジュール化支援の機能があげられる。一般にＬＳＩの設計では、ＰＬＡの使用部分、（既設計または未設計の）人手設計部分等を効果的に切り分けることが重要である。現在の本自動論理合成システムを用いた設計では、H² ＤＬ内部仕様記述の段階でこれらを意識してブロック化した記述を行う必要がある。しかし、内部仕様記述の段階では、回路の詳細については分からないため、上記の切り分け作業を適切に行うことは困難な場合が多い。さらに、トップダウン設計により生成したサブモジュール化部分と、レイアウトのフロアプラニングの際のブロックとは効率を考えると必ずしも一対一に対応しない場合も多い。このため、サブモジュール化、サブモジュールの展開を自動論理合成を進めながら、適宜回路を評価し、柔軟に行えるようにすべく検討を進めている。

（２）　システムＶＬＳＩ開発の実際

伝統的な集積回路の設計手法は紙と鉛筆に基いた人手によるものが中心であった。設計する回路の規模が小さかった従来の回路に対しては、これでも設計することが可能であったしかし、プロセス技術が進んできて、集積される回路の規模が大きくなるにしたがって、機能／論理／パターン設計とこの検証に莫大な時間がかかるようになり、現実的には開発が困難な状況となってきた。これを打破するために、かなり以前より様々なＣＡＤシステムが開発され、実用に供されている。これまでのＣＡＤシステムは従来人手で行ってきたものを計算機に肩代わりさせることに重点が置かれてきたが、勿論これでも大幅な設計効率の向上が実現されたものの、更に一歩進めて、これらのＣＡＤシステムを総合的に使用する事で、これまでとは異なった設計手法が考えられる。本節ではそうしたアプローチの一つを提案したい。

従来、ＬＳＩは次のような設計の段階をふんでいた。まずおおまかな概念設計の後にこれを詳細化する機能設計を行うこの結果を検証するためにブレットボードを作製し、機能の確認を行い、必要に応じて、機能設計の修正を行う。次に機能設計／検証を元に論理設計／検証を行う。この局面を支援するＣＡＤとしては例えばＤＡＩＳＹのＬｏｇｉｃｉａｎ等がある。論理回路的にも動作を確認した後、必要ならば回路シミュレータ（ＳＰＩＣＥ等）で回路の電気的特性を検討／設計し、次にパタン設計に入っていく。パターン設計は特にＣＡＤの出現によって大きく変わったものの一つであるが、古くは全面手描きにより処理していたが、最近ではワークステーション上でのアートワークや、特にＡＳＩＣの場合はスタンダードセル方式の自動配置配線がよく採用される。この様に、これまでの設計手順は逐次設計を進めていくものであった。従ってある所からは後戻りできないし、敢えてこれをおこなえば開発期間の大幅な延長を余儀無くされる。

ところが、前節に示したような高機能／高性能のＣＡＤシステムの出現はこの状況を大きく変えようとしている。　ここでは、比較的回路規模の大きい論理ＶＬＳＩの開発を例として、この状況を示す。このＬＳＩは３２ビットグラフィック専用のプロセッサであり、凡そ１３万トランジスタを集積している。実際の設計は以下のようにしてなされた。まず、これまで通り、ＬＳＩの仕様検討と概念設計を行う。この段階で、ビヘィビアモデルを用意し、幾つかの機能ブロックに分け、基本的なタイミングチャート等を作成しておく。次に機能の詳細設計にはいる訳だが、これは階層的ハードウェア設計言語（H² ＤＬ）によってその機能を記述することがそ

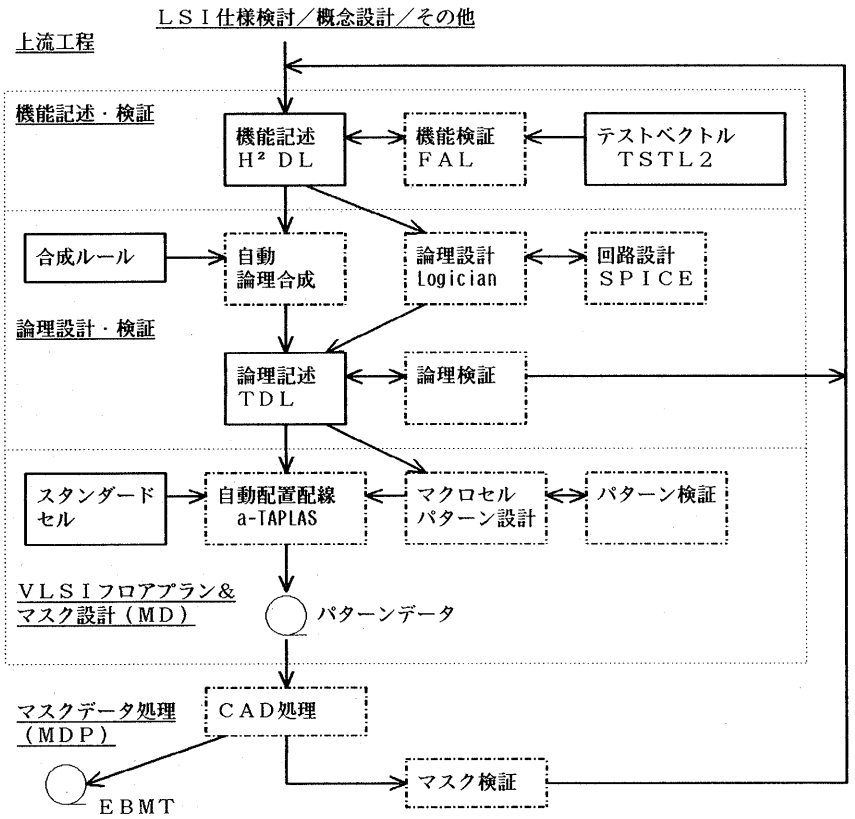

図２．　ＣＡＤシステム全体の流れ図

れに相応する。H² ＤＬで機能記述されたものはこれを解する機能シミュレータで検証される。この際必要となるテストシークエンスは、この後の論理検証や最終的なマスクパータンの検証の際にも使えるように工夫されている。検証の済んだH² ＤＬ記述は自動論理合成システムによって、適当なスタンダードセルライブラリに基く論理回路の接続記述（ＴＥＧＡＳ Ｄｅｓｉｇｎ Ｌａｎｇｕａｇｅ：ＴＤＬ記述）に変換される。このＬＳＩでは通常のＣＭＯＳ論理のスタンダードセルライブラリを用いたが、スタンダードセルライブラリはどの様なものであっても勿論構わない。或いはどのようなスタンダードセルライブラリを使うかによって、そのＬＳＩの性質が決まってくると言っても良い。そして、この自動論理合成システムはルールベース方式であるので、どの様なセルライブラリを用いる場合でも、ルールベースの変更により柔軟に対応でき、短時間のうちにスタンダードセルを割付けたＴＤＬ記述を得ることが出来る。この際、回路的に特殊なものを採用する場合や、ＲＯＭ／ＲＡＭ等はマクロセルあつかいとし、別途、論理／パタン設計することも出来る。実際に高速ＡＬＵやＲＯＭ／ＲＡＭ等でこの方法を採用しており、また加算器等でＣＬＡ方式を使い高速化をはかった部分はＴＤＬ記述のみ設計者が作成し、これに基き自動配置配線を行うことも可能である。自動論理合成システムは原理的には正しい回路を合成するが、回路として適切かどうかはまた別問題である。すなわち冗長なゲートを発生してクリティカルパスを作っていないか、タイミング的に無理をしていないか、スパイクが発生しやすくはなってはいないか等、回路的なチェックは必要である。この為、ＴＤＬ記述を扱い得る論理シミュレータ（例えばＺＹＣＡＤ）で検証する必要がある。これによって、機能検証では見付からなかったバグが発見されることがある。このときは必ずH² ＤＬで書かれた機能記述に立ち返って修正するよう心掛けて、設計しなければならない。H² ＤＬ記述の書き方次第で、大抵の場合、期待されるようなＴＤＬ記述を自動論理合成システムで得られることが分かっている。常にH² ＤＬ記述に戻って修正するということは、ＬＳＩ設計をその上流側で押さえようという一貫した方針であり、万一ＬＳＩとしてのリファインが必要になったときも、これによって比較的短期間でリファインし得るものと思われる。ＴＤＬ記述が得られたならば、スタンダードセル方式の自動配置配線システムによってレイアウトを実行する。最後に作成されたマスクパターンデータから再度、回路を抽出してアフターレイアウトシミュレーションを行い、万全を期した。以上の作業をまとめると図２のように成る。

こうした総合的なＣＡＤシステムの利用は設計者による設計の局面をなるべく機能設計側に引き寄せようとするものである。この事はH² ＤＬ記述を書くことがＬＳＩ設計の大きな要因を占めるものである事を意味する。設計者はH² ＤＬ記述を機能設計するつもりだけではなく、これが回路設計にもなっていることに留意しなければならない。一般にある機能を実現するためのH² ＤＬ記述の書き方は複数存在する。この中で論理回路的にもレイアウト的にも優れているものは、恐らくただ一つである。これを機能記述にいかに反映するかが、ハイパフォーマンスＬＳＩを実現する際の鍵である。ところが、ＬＳＩ仕様を詳細化しはじめた段階では、とてもレイアウトにまでは考慮できないのが普通である。しかし一度H² ＤＬ記述が得られると、自動論理合成と自動配置配線を使ってパターンデータがすぐに得られる。実際、１３万トランジスタ程度でもH² ＤＬ記述からレイアウトまでは数日あれば十分可能である。従ってH² ＤＬ記述の完成度が十分でなくともレイアウトしてみることで、ＬＳＩとしての具体的な姿が容易に得られる。設計の早い段階からレイアウトの試行が出来るので、この結果有効なフロアプランを立てることが可能である。ＬＳＩの設計でフロアプランニングの重要性は改めて指摘するまでもないが、ＶＬＳＩの規模が大きくなるにつれてスタンダードセル方式のレイアウトであっても、チップ全面を一つのブロックとするのは得策ではない。機能的あるいは回路的に纏められるところは纏めて、ブロック化したほうが効率良く、高性能のものが開発できる。フロアプランの出来のよしあしがそのＬＳＩの運命を左右するといっても言い過ぎではない。従って完成度の高いフロアプランを行っておくことは決定的に重要であり、ＣＡＤシステムを総合的に使用することで、これを効率良く行える。このようにこれまでの設計法が逐次的なものであったのに対して、機能設計／検証がある程度進んだ段階で論理設計／検証に取りかかり、検証し終わっていないＴＤＬ記述で自動配置配線を試みることができる。この様な並列化設計手法によって、常にフィードバックのかけられる状態で設計を進めて行くことが可能である。設計すべきＬＳＩの規模が大きくなり、またシステム的にも複雑になっている現状では、バグを全く含まない形にまで機能設計を仕上げるのは、それだけでかなりの時間を費やさなければならない。勿論ＣＡＤシステムによってこれにかかる時間は相当削減されてはいるが、それ以上に機能設計／検証をやっている片手間に、それ以後の詳細設計に役立てるべく、ＣＡＤシステムを十分に活用して試行錯誤をある程度行う必要がある。　機能設計段階を十分に支援できるＣＡＤシステムの出現はファームウェアの設計／検証についても、従来の方法と異なる開発法が考えられる。従来は機能仕様に合せてＬＳＩのハードウェアの設計とは別に開発されることが多かった。この為エミュレータを作成してデバックしていた。今回の３２ビットグラフィック専用プロセッサ用のファームウェアの開発に当っても、念の為、計算機上にエミュレータを開発し一応のデバックを行ったが、最終的には機能／論理シミュレータによって細部の検証を行った。きわめて汎用性の高い論理ＶＬＳＩであれば、十分な時間をかけてエミュレータを開発し、その後にファームウェアを開発しても良いかもしれないが、ＡＳＩＣ的な性格の強いものに対しては、一々エミュレータを開発している時間的余裕は無い場合がほとんどである。従って、このやり方を押し進め、エミュレータなしでファームウェアを開発する手法もあり得るし、むしろ今後の主流になるものと予想される。

以上解説したように、ＣＡＤシステムを総合的に使用し、論理ＶＬＳＩをスタンダードセル方式に基いて、並列化設計手法により開発した結果、従来例に比較して大幅な設計期間の短縮が可能となった。尚、今後の論理ＶＬＳＩの設計手法的な課題としては、有効なテストシークエンス及びその期待値の効率の良い作成法と、詳細な機能設計以前に行われるＬＳＩの概念設計で必要となるビヘィビアモデルの妥当性を支援するＣＡＤシステムの必要性を特に指摘しておく。

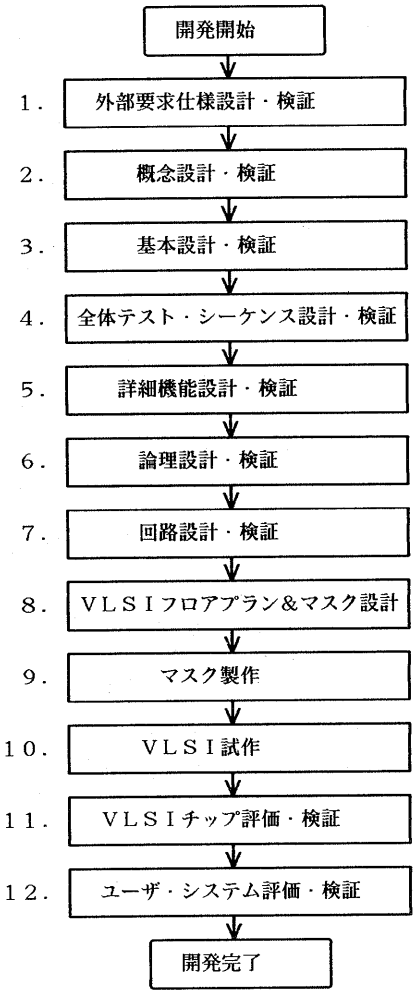

図３．　システムＶＬＳＩ開発工程案

４．　まとめ　（結論ー効果、今後の方向）

以上の展開から、結論として、付加価値の高いシステムＶＬＳＩを短期間で比較的容易に設計・試作出来る開発戦略とそれを実行する手法、及び開発手順を提案する。

ＶＬＳＩの規模と複雑さ、及び高速化に伴って、もはや回路を汎用のロジックＩＣを用いてエミュレートする、いわゆるブレッドボードでは機能と論理と実時間の検証をすることは、極めて困難になってきた。また膨大な論理図を正確に短期間に作成する事も難しい。システムＶＬＳＩの場合は、ほとんど不可能である。

開発事例にその効果が示されている通り、ＡＳＩＣ手法の活用により従来の開発手順に比べて、著しく設計効率が向上した。その中から、特に画期的な点は、機能記述言語（Ｈ²ＤＬ）の採用と機能シミュレータ（ＦＡＬ）による機能検証前述の自動論理合成、スタンダードセル・ライブラリの適用階層化自動配置・配線等の活用である。その他の効果として機能記述を厳密に行っていく為に、システム設計の仕方が系統だって出来、文書かが整然と実施される点も大きい。

以下に、今後の課題も含め、高付加価値のシステムＶＬＳＩを短期間で比較的容易に開発するための幾つかの戦略を提案したい。

（１）　図２のシステムＶＬＳＩ開発工程案の中から、１～３は、構造化分析・設計技法を採用すべきである。システム＆ソフトウェアに適用しているＳｏｆＴｅｃｈ社のＳＡＤＴ（Ｓｔｒｕｃｔｕｒｅｄ　Ａｎａｌｙｓｉｓ　＆　Ｄｅｓｉｇｎ　Ｔｅｃｈｎｉｑｕｅ）を応用すると良い。動作毎のリストとその操作の対象となるデータのリストを洗いだし、それらを機能的強度の高いグループに分けることから作業を始め、機能フロー図の形にまとめる。また、状態の遷移の定義、詳細化、検証には、ペトリネット図を応用することを勧めたい。勿論、この図も平坦な状態数の羅列としてはいけない。抽象化レベルを設けて、階層的に配置しなければならない。

（２）　４、１１、１２の局面では、複雑に絡み合ったテスト項目とテスト条件とそれらの相互作用を明らかにして、最適なテスト項目数に減らす工夫をしなければならない。

（３）　大規模なシステムを機能分割し、複数の階層に分けるのは非常に広い経験と、深い洞察力を必要とするので、誰でもできる仕事ではない。この仕事を比較的やり易くするために、ＡＳＩＣのスタンダードセル・ライブラリに目的機能実行が出来る、高度に洗練されたスーパ・マクロセルを特定機能モジュールとして、開発・登録していきたい。これにより、比較的階層の上位レベルで、早いうちに、具体的な見える回路に辿り着くことができる。更に、このセルをソフトウェア・マクロではなく、ハードウェア・マクロとして、即ちトランジスタ・レベルの組み合わせで工夫して、高性能の回路・パターン設計しておく事が大切である。

（４）　システムＶＬＳＩを正しく動作させるために、ＶＬＳＩフロアプランの概略をシステム設計者が自ら行う必要がある。電源配線の特性や、ブロック毎の分離の問題は、難解である。信号線間の相対遅延時間も重要な検討項目である。自分のＬＳＩのブロック配置は、性能に直接的に影響があるからである。

（５）　トップダウン設計で、概念、基本、詳細機能のどのレベルでも、全体とそれを構成するモジュール毎に起動時刻と実行時間（遅延時間）を設計時に見積もったり、目標値を割り当てたり出来なければならない。少なくとも、論理図が完成し、セルが割り付けられてからしか見積もれないと言うことだけは、避けたい。つまり、開発の上流工程こそ設計と

その検証ツールに、時間を表す変数群が利用出来なければならない。ここで言う時間とは、大小、包含関係、時刻の前後関係、独立した並行動作関係、同期を含む相互依存関係、親子関係等の抽象化したレベルの表現も含む。

（６）　詳細設計以上の開発上流工程と、システムＶＬＳＩチップの試作後のチップ評価や、システム評価・検証局面にも、広い意味でのＡＳＩＣ手法の活用を進めていく必要がある。工程上のそれらの局面こそ、ＡＩ技法の活用の場が与えられている。多面的でダイナミックな戦略そのものをを柔軟に記述しなければならないので、システム設計者自身が戦略を立てて、それを直接ＡＩ言語で記述していくべきである。何故ならば、汎用のエキスパート・システムでは大切な、仕事の流れの全てを連繋して使うことは困難であると判断出来るからである。利用できるエキスパート・システムがあるとすれば、それは次のあらゆる意味でオープン・システムでなければならない。従来のＶＬＳＩ－ＣＡＤシステムとコミュニケーションが出来ること。即ち、データの互換性が必須である。他の汎用システムの応用ソフトウェアから利用できる。ＥＷＳ（Ｅｎｇｉｎｅｅｒｉｎｇ　Ｗｏｒｋ　Ｓｔａｔｉｏｎ）の環境だけの開放のみならず、他の言語とのリンク、とりわけＡＩ言語（Ｐｒｏｌｏｇ、Ｌｉｓｐ等）で自由に記述したものと導入しようとしているエキスパート・システムがリンク出来なければならない。

結論として、付加価値の高いシステムＶＬＳＩの開発で、最も活用すべきものは、広くとらえたＡＳＩＣ手法である。即ち、専門家自ら主になって、図３．に示したシステムＶＬＳＩ開発工程案のそれぞれの局面毎のサブ・システムを、工程の流れ（手順）に最適で、かつ開放的なものとするように、大勢の協力者と一緒になって構築していく必要がある。

文献

［１］宮田、増渕、西尾：”ＬＳＩ論理設計エキスパートシステム”、東芝レビュー、42巻5 号、p.367 ～370, 1987.

［２］西尾、宮田、山崎：”階層的ハードウェア設計言語Ｈ² ＤＬの言語仕様ー内部仕様記述とハードウェア・プロセス記述ー”、情処会　設計自動化、22-2, 1984.

［３］　国友　義久、”効果的プログラム開発技法”、近代科学社、1979

［４］Grady Booch, SOFTWARE ENGINEERING WITH Ada, United Stated Air Force Academy, 1983

［５］Philip W. Metzger, MANAGING A PROGRAMMING PROJECT, Prentice-Hall, 1973

［６］Glenford J. Myers, RELIABLE SOFTWARE THROUGH COMPOSITE DESIGN, Mason/Charter Publishers, 1975

［７］Glenford J. Myers, The Art of Software Testing, John Wiley & Sons, 1979

［８］William C. Hetzel, PROGRAM TEST METHODS, Prentice-Hall, 1973