第17回 LSI設計技術

厚木エレクトロニクス　代表
サクセス インターナショナル 取締役
加藤俊夫

サクセス インターナショナル 技術顧問
逸見文明

今まで，半導体のプロセス，デバイスの話題を取り上げて来ましたが，今回は話題を変えて，半導体の設計です。そこで，長年，LSIの設計に携わってこられた逸見さんに助太刀をお願いし，今回の原稿の99％は同氏に書いて頂きました。設計と言ってもデバイスの種類はデジタルからアナログまで大変な種類がありますので，ここでは代表的なデジタルICの設計の話です。

半導体の設計の階層構造

半導体の基本単位はトランジスタで，特にデジタルICの場合はCMOSです。そのCMOSを組み合わせれば，デジタルIC中の論理回路の部分はどんな回路であっても合成できます。つまり設計とは「CMOSをどのように組み合わせるか，CMOS間の配線をどうするか，を決めることである」と言えます。論理回路以外の部分で重要なのはメモリで，SRAMやDRAMの設計の仕方は，また少し理屈が異なりますので，ここでは省略します。

さて，設計の対象はマイクロプロセッサやシステムLSI，携帯電話機の中の各種デジタル回路等々ですが，これらの設計と，半導体のCMOSの組み合わせをどう結び付けるか，そこがわかって来れば，半導体の設計の仕組みも理解できるはずです。

半導体の回路規模が小さい時は，CMOSの組み合わせを人間が直に考えて簡単な回路を設計していましたが，微細化・大規模化の勢いは激しく，3年で4倍の規模に膨れ上がってきました。そんな時代に，どうやって対応してきたのでしょうか。

これは設計を階層化することです。人間ひとりの頭で考えられることは限りがあります。自動車を作る場合も，一人で設計することはできません。全体の構成を決める人，個々の部品を設計する人，部品間のつなぎを考える人，全体の重量を決める人など，分業が行われます。LSIの設計でもいくつかの専門家集団が自分たちの得意な領域を明確にし，それぞれの部分について責任を持って設計する形態を取るようになりました。その仕事の流れを図1に表しました。

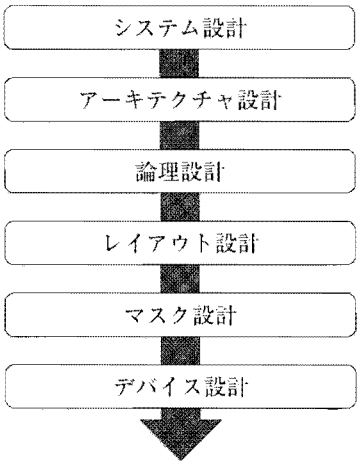


図1　設計の階層構造

一番上の「システム設計」とは，そのICの全体の構造を設計する段階で，おおまかな仕様を決定する部分と言えます。次の「アーキテクチャ設計」とは主にICの機能をソフトウェアで実現するか，ハードウェアで実現するか，その分担を考える段階です。

次は「論理設計」で具体的に各機能を実現する回路を考えていきます。この段階を過ぎると，回路で使うCMOSの種類や，CMOS間の接続はほぼ決定されます。「レイアウト設計」は，各回路をチップのどこに配置するか，配線をどう構成するかを決定する段階です。「マスク設計」では，そのレイアウト構造を実現するために，

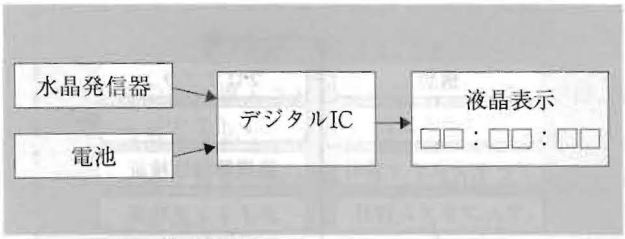


図2　デジタル時計の構造

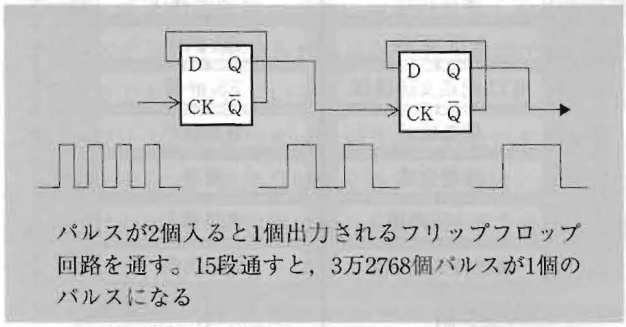


図3　フリップフロップによるカウントの仕組み

半導体の各プロセスでどのように描画を行うか，そのマスク・パターンを設計する段階です。ここまでは，新規のICの設計の場合，必ず一度は通らなければいけない階層になります。

一番下の「デバイス設計」と言うのは，CMOSをどのような構造とするか，100nmか80nmか，Al配線かCu配線かなどのデバイスを決めます。半導体メーカーが担当するところは，これ以外にもデバイスの設計のためにプロセスでの試作をするとか，信頼性の評価をするとか，この上の設計階層とうまく繋がるようにコンピュータを使ったCADシステムを構成するとか，様々な仕事がありますが，ここでは，簡単に一つの階層で表しています。

このようにして各階層の中では，その専門家集団が自分たちの得意な分野に集中して大規模な設計に取り組むことができます。階層間の約束事をきちんと決めておけばお互いの責任も明確になり，仕事の効率も上がります。こうして，例えば論理設計者はCMOSのことを意識せずに論理のことだけ考えて設計できますし，デバイス設計者は論理のことを考えずにデバイスのことに集中できます。自動車の設計者は，エンジンの細かい構造は知らなくても，出力，重さ，燃費などのデータさえあれば自動車を設計できるようなものです。このようにして半導体の設計は大規模化に対応してきました。こうした設計の階層化は半導体に限らず，現在の巨大システムの設計で

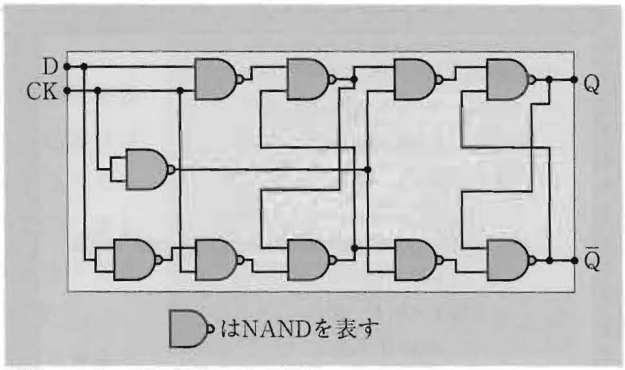


図4　フリップフロップの内部

は広い範囲で行われていることと思います。

簡単な回路例で

さて，具体的に設計とはどんなことをするのか，簡単な回路の例で説明してみましょう。図2を参照下さい。これはどこにでもあるデジタル時計です。電池がある限り，自分で時をカウントし，液晶表示に何時，何分，何秒と表示しますが，中心部分はデジタルICです。これを設計していく時，どのように進めていくでしょうか。

時計を作るときの基準になる「時」は水晶発振器か，国内で常に電波で送られている標準電波か，そのような安定な周波数源が使われています。ここでは，32,768Hzの水晶発振器の例で考えてみましょう。まず，1秒間に3万2768回の波を発振器から受取り，それを3万2768回数えれば，1秒の信号を作り出せます。数える論理回路はカウンタと呼ばれていて，これはデジタル回路技術の中では良く知られた手法です。カウンタの中身はフリップフロップと言う回路の集合です。フリップフロップが一つあれば，2を数えることができます。図3は，2個のパルスがフリップフロップに入り，1個のパルスが出て行く様子を表しています。二つあれば4を，三つあれば8を数えられ，そのあと16，32・・・と考えていけば，3万2768を数えるにはフリップフロップが15個あればできることになります。

種明かしをすれば，32,768Hzと言う周波数はフリップフロップが15個あれば，ちょうど1秒になる都合の良い周波数だったのですね。1秒に1回の信号を作り出せれば，それを前の数に足していけば1，2，3・・・と秒の表示はできます。これで1秒の位の表示は可能になりました。9まで数えたら桁が足りなくなりますので，上位

```
module CT16 (clk, rst, clr, hex);
  input          clk, rst, clr;
  output [3 : 0] hex;
  reg [3 : 0]    hex;

  always@ (posedge clk or negedge rst) begin
    if (! rst)
      hex <= 4' d0;
        else if (clr)
          hex <= 4'd0;
    else begin
      if (hex >= 4'd15)
        hex <= 4'd0;
      else
        hex <= hex + 4'd1;
    end
  end
endmodule
```

図5　16進カウンタのHDL記述

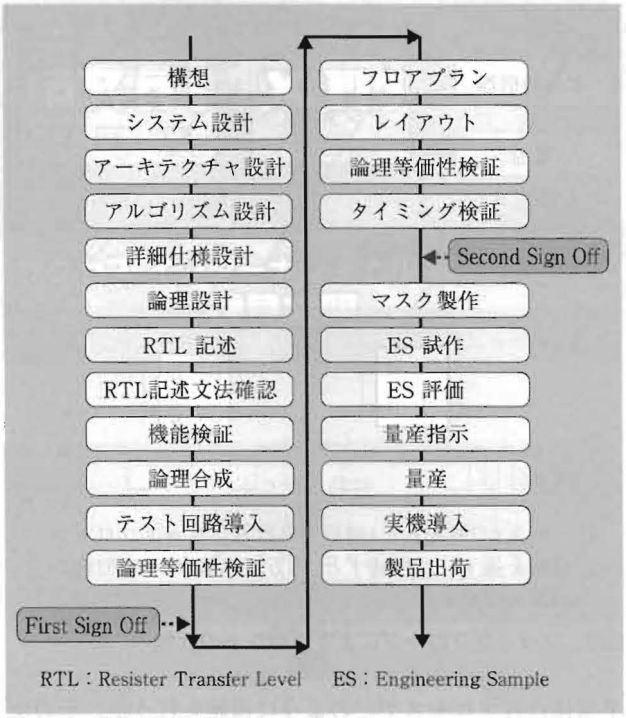


図6　一般的なデジタルICの設計フロー

の10秒の桁に桁上がり信号を出し，自身は0に戻します。10秒の位の表示も，秒の位からきた桁上がり信号を数えていけばできます。後は，分の位，時の位も同様に考えれば良いわけです。このようにデジタル時計はカウンタがあれば実現できることがわかります。

カウンタの中身はフリップフロップでした。そのフリップフロップはどうやって実現するかと言いますと，図4のような回路で実現できます。ここにある“蒲鉾の断面に小さな丸をつけたようなマーク”は論理回路のNANDの回路を表しています。詳しい説明はここでは省略しますが，NAND一つはCMOSの二つ分で構成できます。

このようにデジタル回路は各要素を分解していくと，全てCMOSになります。よってCMOSを組み合わせていけば，あらゆる回路が合成できるのですね。組み合わせと言うのはCMOS間の配線で実現できますから，結局は設計とはCMOS間の配線に帰結することになります。

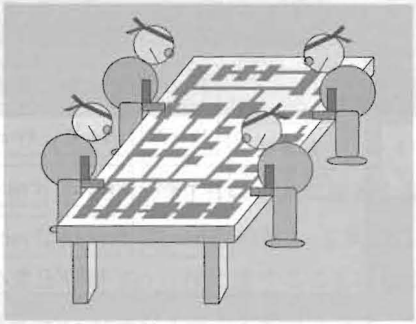
デジタル IC の設計フロー

さて，前節ではデジタルICの内部がどのように構成されていて，それがどう設計されるかを概念的に見てきました。それでは実際の半導体設計ではどのようなフローで設計されるか，次にそれを説明しましょう。

実際の設計では，一々回路図を書いているでしょうか。答えは「ノー！」です。回路図による設計は80年代ぐらいまでは主流でしたが，ICの規模が大きくなるにつれ，回路図ではページ数が増え，全体を把握できなかったり，複数の設計者で進めると回路図間の接続が複雑になりすぎるなどの問題で，次第に使われなくなってきました。代わりに登場したのがHDLと言う言語を用いた設計手法です。HDLとはHardware Description Language の略ですが，例えば，16進のカウンタでは図5のような記述になります。

まるでソフトウェアの書き方と同じなのですが，まさにソフトウェアのような扱いができるため，HDLは前記の問題を解消して行きました。このHDLの発達に欠かせなかったのが論理シミュレータです。論理シミュレータはCADシステム上で動くソフトウェアで，図5のようなHDL記述を受け取って，コンピュータ上でシミュレーションを行い，設計者が意図している動作になっているかを検証します。この手法のおかげで設計者はハードウェアを組み立てることなく，論理動作を確認でき，設計の効率が格段に上がりました。このようなHDLを使って論理設計を行い，事前に検証を進めて設計信頼性を上げ，

＜ちょっと脱線＞一発完動で感動


▲マスク・パターンの検査

上流設計が導入される前のIC設計の職場では，紙の上に回路図を書いて設計を進めていましたが，上流設計が導入されると，設計者はコンピュータ端末に向かい，そこにHDLを入力し，サーバにあるシミュレータを動かしてシミュレーションを行う，と言う形態に変化しました。大規模なLSIの設計の職場では，コンピュータ端末がずらりと並び，多くの設計者がそれに向かってもくもくと仕事をしています。

ステッパが開発される以前の露光装置では，写真撮影したネガをマスクとして用いていました。回路パターンは，ルビリスと呼ばれる赤いフィルムをカッターで切り，それをベテラン検査員が数人で1日中目視検査をして誤りがないか検査をします。そして，試作品が完全動作した時は，ビールで乾杯です。不幸にして間違いが発見されると，直ぐに再試作にかかり，お客様に謝りに走ります。現在のLSIでは，人間の目視が不可能なのでソフトウェア的に検査しますが，それでも一発で完動した時は嬉しいものです。

それからハードウェア（この場合はICチップ）を製造することを上流設計手法と呼んでいます。この設計手法は90年代になってから急激に設計現場に浸透し，今ではごく当たり前に行われています。その上流設計手法での設計フローを図6に示します。

あまり聞き慣れない専門用語が並んでいますが，それらの説明は別の機会に譲ることとし，要するにこの図は図1で示した設計の階層構造を実現したものだ，と言うことをご理解いただきたいと思います。

この中で特に重要な設計工程は論理合成です。それは，図5のようなHDLの記述を実際の回路素子の繋がりに置き換えて行く作業です。図5の段階ではあくまでもコンピュータ上に存在する記述でしかありません。半導体にするためには，回路の各素子に何を使い，それぞれの素子を配線で結ぶために，どの素子とどの素子をどう結ぶかを決めていく必要があります。素子の選択の際に重要になるのが半導体メーカー側の用意するライブラリです。ライブラリ中には，図4の中で示したNANDとか，あるいはフリップフロップとか，CMOSを組み合わせた具体的な回路（これをセルと呼んでいます）のデータが入っています。論理合成を行うのは論理合成ツールと言うCADシステム上で動くソフトウェアですが，この論理合成ツールは図5の記述と，その回路を動かす電流条件などからセルを選択して行き，それらのセル間の接続データを作り出します。このようにして作り出されたセルとセル間の接続情報はネットリストと呼ばれています。

ネットリストは，次の段階でタイミング検証を受けます。それは，実際に回路を動かした時にセル同士を信号が時間の制限通りに伝わって行くか，チップ全体では設計の要求通りの速度で動くかどうかをチェックするものです。

こうしてネットリストは正しい論理か，正しいタイミングで動くか，をチェックされ，マスク行程に渡されて行きます。これが図6で示したSecond Sign Offと言うイベントで，それまではハードウェアが作られることは一度もありませんでしたが，これから先は，マスク，半導体サンプルの試作と言うようにハードウェアが作られて行きます（それもかなり金額が高い）。と言うことで，このイベントは「これから半導体の金をかけてもいいぞ！」と言う設計者からの意志表示になり，これはこれで随分と責任の重いイベントとなります。

大規模化への対応

今回のまとめとして，今後のこと，将来のことを考えてみましょう。半導体の規模は3年で4倍の規模に拡大を続けています。これは設計者にとっても大変なことで，もし，設計者数が変わらず，手法も変わらないと，3年経つと今までの4倍の仕事量になっていることを意味します。これが今，大問題になってきています。何かこれからの大規模化に耐えられるような方法はないか，それ

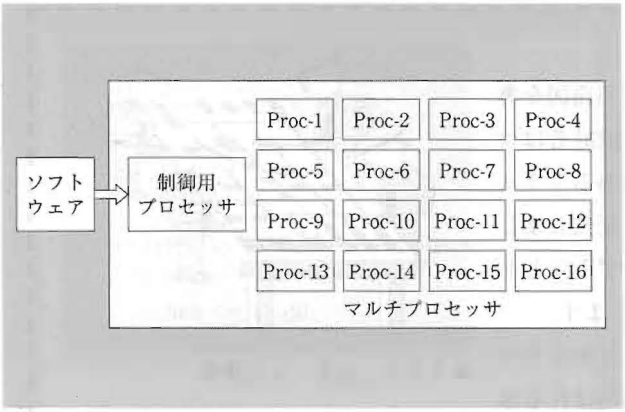


図7　ソフトウェア制御とマルチプロセッサの概念図

を今，設計者は真剣に考えています。その中でも注目に値するのは“ソフトウェアによる制御とマルチプロセッサ”でしょう。その概念図を図7に示します。

ソフトウェアによる制御とは，ICチップ内部にソフトウェア制御部を取り込み，そこに外部から組込みソフトを入れることでチップ全体の制御を外部から自由に変更できる仕組みのことを言っています。今まで，すべての制御をハードウェアに頼っていたため，IC設計に一つでもミスがあると，それを直すために，もう一度，最初から作り直すしかなかったのですが，この方法を取れば，外部から入れるソフトウェアを組み直すことで制御の範囲内であれば対応できます。

マルチプロセッサとは，MPUならばコアを一つでなく複数にする，他の信号処理プロセッサでもプロセッサを複数設ける手法です。画像処理のように並列演算の多い処理にはこの手法は大変に適しています。ハードウェア設計者は一つのプロセッサの設計をし，それを複数並べるので，負担が軽減できます。

この手法と先述のソフトウェア制御の手法を組み合わせ，それぞれのプロセッサの制御を外部から入れるソフトウェアで変更できるようにすると，柔軟な制御が可能になりますし，設計リスクを減らすことができ，かなり大規模なLSIの設計ができます。

以上，設計について概論を簡単に述べました。ここに書かれていることが少しでも半導体の設計についての理解の助けになれば幸いです。書いていて感じたことは，設計について，まだまだ言い足りないことが多くあることです。それらのテーマは，また何かの機会に紹介させていただきます。