連　載

第5回

論理合成可能なRTLモデルを作る

渡辺　玲

Verilog-HDL によるコンピュータ設計

前回はパイプライン・プロセッサのRTLモデルを設計して，その動作を試験し，性能を評価しました．

今回はRTLの設計フェーズからもう一段進めて，ロジックの設計フェーズに移ります．ここでは論理合成ツールが登場します．

ロジック設計の変遷

これからロジックの設計をするのですが，ここで少し筆者がたどったロジック設計の歴史を振り返ってみたいと思います．

●大昔のロジック設計

昔々，筆者が大学を出て初めて会社に入社したころ(約15年前)，最初にみんなが買ったのはANDやORの論理回路が載っているテンプレート定規でした．そのころすでにいくつかのHDLは存在していましたが，あまり普及してはいませんでした．それでチップを設計するといえばまず回路図を描くところから作業は始まったのです(一部ではPASCALやCでまず動作を確認していた．筆者の会社でもCADグループがシミュレーション用ライブラリのようなものを提供していた)．

そのころの回路図入力には特別な機械が必要でした．カルマなどが有名で，ハードとソフトが一体となった「CADシステム」として売られていました．とても高価なもので社内にあまり台数がなく，CAD(グラフィック)ルームと呼ばれる部屋に数台まとめておかれていました．

回路を設計するエンジニアは，まずテンプレートを使って方眼紙の上に鉛筆と消しゴムで回路を設計しました．鉛筆で描いては消しゴムで消してを繰り返して，やっと求める回路が完成したらそれをCADルームに持っていき，自分で回路図を入力するか，ディジタイザと呼ばれるディジタイズ専門の人にお願いして入力してもらうのです．

●回路図入力の意義

ちなみに回路図をコンピュータに入力する利点は，人間向けのドキュメントとしての意味だけではありません．それだけならドローイング・ソフトやワープロ・ソフトでも代用可能です．

本当の意義は，入力された回路図を設計データとして活用するところにあります．

歴史的に見て，回路図を設計データとして使うツールとして最初に登場したのがレイアウトと回路図をくらべるツールです．LVS(レイアウト vs.スケマティック)などと呼ばれています．

その次に登場したのが，回路図をSpiceなどのサーキット・シミュレータのモデルや，MOSSIMなどのロジック・シミュレータのモデルに変換するツールです．これのおかげで，HDLが登場する以前でも回路のシミュレーションはできました．

そして，最後に登場したのが，レイアウト作業を革命した自動配置配線ツールです．回路図さえあればもうレイアウトはいらない(？)とまでいわれました．

もっとも，人間の観点ではなく，ツールの観点ではEDIFなどのネットリストこそが真の回路の設計データであり，回路図はネットリストを作るための一つの手段にすぎません．実際，論理合成を行うようになり，論理合成ツールがネットリストを直接出力するようになってからは，回路図なしでもチップ設計のすべての行程を進めることができるようになりました．

●昔のロジック設計

入社後何年かたち，筆者がチップの設計を本格的に始めたころ(約10年前)は，HDLが普及してきて，まずRTLモデルを設計し，動作と性能の確認をして，そのあとに論理回路を設計するといった作業工程(デザイン・フロー)になりました．

すでに機能面での設計はRTLモデルを設計した段階で終了していますので，論理回路の設計はかなり単純化されたものになりました．

(1)ゲート(スタンダード・セル)にマップすること．
(2)回路を小さくする最適化をすること．
(3)回路を速くする最適化をすること．
(4)可能なかぎりリソース・シェアをすること．
(5)適正なサイジング(ファン・アウト)をすること．

などなどです．

具体的には，できたRTLモデルを目の

前に置いて，それを1行ずつ論理回路に落としていく作業です．

そのころは使い勝手のいい回路図エディタのソフトが出ていました．昔と違い，特別な機械の上ではなく，自分の机の上に置かれたワークステーションの上で回路図の入力ができるのでたいへん便利でした．ちょうどPCに向かいプログラムをテキスト・エディタで作る感じで，ワークステーションに向かい，ロジックを回路図エディタで作っていました．

筆者がCPUメーカにいたときは，自社製の回路図エディタをDECのマイクロVAXの上で走らせていました．HDLも自社製です．そのあと現在の会社に移ったあとは，Cadenceの回路図エディタEdgeをSUNのSPARCの上で走らせていました．HDLはVerilog-HDLです．

現在は，回路図エディタとしてはCadenceのOPUSを使っています．ただ最近は論理合成ツールを使うようになったので回路図エディタはあまり使っていません．小さなバグをメタルのパッチをあててフィックスするときに回路図エディタを開くくらいです．

●最近のロジック設計

筆者も約5年ほど前から，論理合成ツールを使うようになりました．職場の同僚はそれより前から使っていたのですが，その当時，筆者の受け持っていた回路が比較的小さかった(約5Kゲート)のと，筆者はこう見えても(？)先頭を走るのが嫌いなたちなので，ツールが安定して，環境が整うのをひそかに待っていました．

しかし，FPUのコントロールを設計するときにとうとう覚悟を決めました．…さてやってみた感想はというと，大きな問題もなく意外とすんなりいきました．

もともと筆者のRTLモデルのコーディング・スタイルは，次に論理回路を手作業で設計することを前提にして書いていましたので，かなり論理回路に近いものでした．

このときも，その延長線上でRTLモデルを書いていました．それで論理合成ツールはあまり混乱することなく素直に論理回路を生成したのだと思います．エラーも問題となるワーニングもなかったのです．

しかしいくつかの遅延時間は見事にバイオレーションしてしまいました．論理合成ツールが回路をとことん最適化しても間に合わなかったのです．ここからは人間様の出番です．クリティカル・パスのあるデコード回路などいくつかの回路をパイプラインの前段や後段に移動することに決め，RTLモデルを書き直しました．変更後，動作試験で動作を確認し，再度論理合成をかけましたら，今度はうまくいきました！

●論理合成ツールの強み

パイプライン・レジスタをまたぐ設計の変更はかなり大きな変更なので，論理回路を自分で作って，回路図も自分で入力したあとに，もしこれをやらなければならなくなったらそれは涙です．何週間分の労力がむだになるのは悲しいことです．そのうえ，さらにやり直しに数週間必要だと上司に報告しなくてはならないことを考えると，まったく，仕事を捨てて山にこもりたくなります．

筆者は長い経験で論理回路の最適化やサイジングには自信がありました．「初心者ならいざ知らず，自分が論理合成ツールを使う利点は果たしてどれほどあるのだろうか？」と思っていました．しかし，きびしいスケジュールにもかかわらずドラスティックな変更ができたことを経験して，「論理合成ツールは便利なツールなんだ」と実感しました．

HDLや論理合成ツールはチップ設計の作業工程の短縮に寄与すると一般にいわれますが，筆者の見方では作業工程の短縮より設計品質(性能)の向上に寄与するほうが大きいと思います．

直接回路図を描かずにHDLで設計すれば，性能改善のための作り直しが現実的になります．ビヘイビア・レベルとかレベルが上位であればあるほど作り直しは簡単です．モデルの性能評価はなるべく上位レベルのうちに行います．もしモデルが条件に満たなければモデルを作り直します．作っては壊し，は性能を改善するには必要な作業です．最初から回路図では作った第一号で良くも悪くもおしまいです．

また，たとえ下位のレベルまで下がって問題(よくあるのはタイミング関連)がみつかっても，論理合成ツールがあればRTLモデル(やさらに上位のモデル)から作り直すことが可能です．

何かを作って，ただ動作すればいいというレベルの話であれば別ですが，それではアマチュアの仕事です．プロとして，限られた条件で最高の性能を求めるのであれば作り直しは避けられません．動作試験だけでは不十分で，性能評価が必要なのです．性能が出なければ正しく動作していても意味がありません．

論理合成可能なRTLモデルとは？

論理合成ツールを使うのには，RTLモデルを論理合成可能なRTLモデルに作り直さなければなりません．

●守るべき三つの条件

そもそも「論理合成可能なRTLモデル」とはいったい何なのでしょうか？どうしてそういうものが必要なのでしょうか？

筆者は，論理合成可能なRTLモデルを記述する上で必要な条件を大きく三つに分けて考えています．

(1) 論理合成ができる記述はHDLの記述全体の部分集合であること．よって論理合成不可能な記述を取り除くこと．
(2) レジスタやマルチプレクサのインファ(推論)を手助けする必要があること．そのために適切な論理合成埋め込み指示文を挿入すること．
(3) 論理合成ツールの知能には限界があること．よって，ツールが最適化をやりやすいようにコードを書き直すこと．

実際は(2)と(3)はなくても論理合成は一応できます．そういう意味では論理合成可能なRTLモデルの必要条件は(1)の条件を守ることだけです．しかし広い意味