論理合成のトラブル・シューティング(前編)

—“Zen of RTL”

Jack Marshall

論理合成ツールを使用する場合，HDLの文法にしたがって記述しただけでは適切な回路は生成されない．動作速度の遅い回路が生成されたり，規模の大きな回路が生成される場合も珍しくない．RTL設計のスキルを身につけるには，ある程度の修練が必要となる．本稿では，回路を合成した結果，所望の仕様を満たしていなかった場合の対処法について解説する．米国Synopsys社の論理合成ツール「Design Compiler」を例に説明するが，それ以外の論理合成ツールを利用している方にとっても，本稿は参考になるだろう．　（編集部）

ここでは米国Synopsys社の論理合成ツールである「Design Compiler」を利用する際に発生したトラブルを解消する方法について紹介します．Design CompilerはASICやシステムLSIの設計に標準的に利用されている論理合成ツールです．

まず，タイミングに関するいくつかのトラブル対策について説明していきます．

1．スネーク・パスが最適化処理を妨害していないか？

Design Compilerのタイミング・レポートを見てください．クリティカル・パス（信号伝播遅延時間が大きいパス）が複数の階層の境界にまたがっていませんか？ タイミング・レポートの左側の列（Point）を見れば，それがわかります（**図1**）．モジュールのポートに対して，ここに複数の階層パスが示されていれば，そのパスは階層の境界をまたいでいることになります．

クリティカル・パス（始点となるフリップフロップから終点となるフリップフロップまでのパス）が論理階層全体にまたがっている場合，これを「スネーク・パス」と呼びます（**図2**）．

スネーク・パスがあると，論理合成ツールの最適化処理が有効に働きません．RTLコードを論理合成ツールに入力すると，論理レベルの最適化が行われます．すなわち，多くのブール代数式を組み合わせて適切な論理構造を実現し，タイミングの目標を達成します．Design Compilerでは，デフォルトの設定のまま実行すると，階層内のモジュールのすべてのポートの定義が維持されます．そのため，ほかのモジュールの中の論理と一つにまとめるため，階層の境界をまたいで回路を移動させるといったことは行われません．結果として，クリティカル・パスが階層をまたいでい

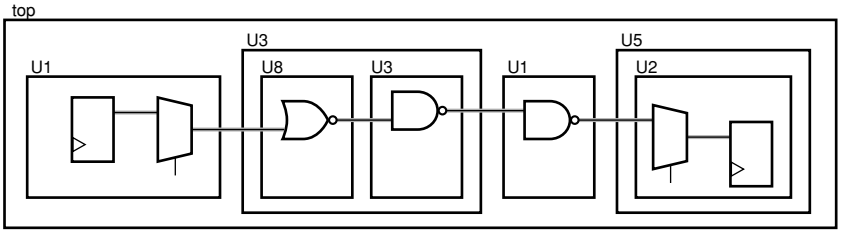
タイミング・レポート

Point		Incr	Path
U1/s_reg[21]/Q（DFFH4）		0.25	0.25 f
U1/U1119/Y（MX2X4）	ライブラリ・セル	0.18	0.43 f
U1/State（ACT）		0.00	0.43 f
U3/State（CNT）	階層の境界	0.00	0.43 f
U3/U8/State（ENB）		0.00	0.43 f
U3/U8/U814/Y（NOR3X6）		0.15	0.58 r
U3/U8/Stall（ENB）		0.00	0.58 r
U3/U3/Stall（STT）		0.00	0.58 r
U3/U3/U3429/Y（NAND2X8）		0.16	0.74 f
U3/U3/Stall（STT）		0.00	0.74 f
U3/Stall（CNT）		0.00	0.74 f
U1/Stall（ACT）		0.00	0.74 f
U1/U950/Y（NAND2X6）		0.13	0.87 r
U1/Go（ACT）		0.00	0.87 r
U5/Go（FIN）		0.00	0.87 r
U5/U2/Go（AA）		0.00	0.87 r
U5/U2/U737/Y（MX2X6）		0.14	1.01 r
U5/U2/g_reg[0]/D（DFFH4）		0.00	1.01 r
…			

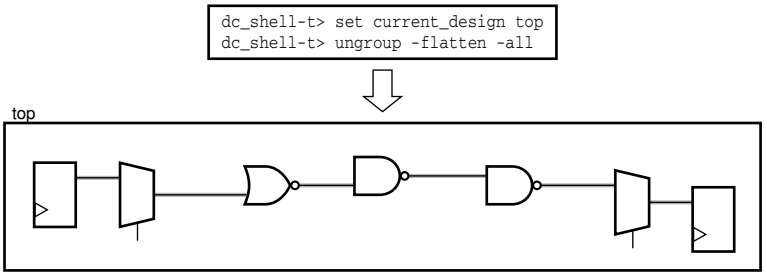
トップ・レベルのセル名が違っているのは，クリティカル・パスが階層をまたいでいることに対応する

〔図1〕スネーク・パスのタイミング・レポートの例

Pointの列に複数の階層パスが示されていれば，そのパスは階層の境界をまたいでいる．ここではタイミング・レポートの一部を示した．



〔図2〕
スネーク・パスの階層ブロック図
四角の箱は階層を表現している．



〔図3〕
階層を取り除く
階層を壊してフラットにした．もはやスネーク・パスは存在しない．

る限り，Design Compilerのタイミング最適化の能力を十分に引き出すことができません．

この問題にはいくつかの解決策があります．以下に示す二つの対策では，クリティカル・パス中の階層の境界を取り除くことで対処しています．

●すべての階層をなくし，フラットにする

もっとも単純な解決策は，Design Compilerの`ungroup -flatten -all`コマンドを使用することです．これによって`current_design`の中の階層がなくなり，フラットになります．スネーク・パスの問題は必ず解消します(**図3**)．しかし，この手法にはいくつかの問題点があります．設計階層をすべて取り除くと，ゲート規模やワイヤ・ロード・モデルの精度，デバッグのしやすさなどが大きく変化します．

規模の大きい設計データ(100万ゲート以上)の場合，階層を取り除くと，設計データの取り扱いは非常にやっかいになります(例えば，コマンドの処理が遅くなる)．また，タイミング最適化の処理に長い時間がかかる可能性があります．

ワイヤ・ロード・モデルは，与えられた回路規模とファンアウトごとの平均ネット(配線)長に基づいて計算されます．回路規模が大きくなるにしたがって，平均ネット長に基づくワイヤ・ロード・モデルの見積もり誤差が大きくなります．これは，ネット長がチップ内部の物理的なセル配置の影響を受けるためです．例えば，ファンアウト数が1のパスがすぐ隣のブロックに接続されていると，ネット長は短くなりそうです．しかし，実際の配置では，そのネットがチップ上の端から端まで横断することになるかもしれません．

20万ゲートを超えるフラットな設計の場合，タイミングや寄生容量/抵抗の制約を満足させるため，なんらかの方法でレイアウト情報をバックアノテーションし，論理合成処理に反映させる必要があります．また，100万ゲートのフラットな設計では，デバッグ時に回路図の中から特定のインスタンスを見つけることは容易ではありません．さらに，論理合成はシングル・ビット・レベルで適用されるので(つまり，一度に1ビットずつ最適化される)，非常に細かい論理構造の中から特定のパターンを見い出すことは容易ではありません．

したがって，スネーク・パスの問題を解決するためにすべての設計階層をなくすのは，小規模な回路(例えば20万ゲート未満)の場合に限るべきです．

●一部の階層だけを壊す

RTL記述やDesign Compilerの操作だけで対処できる別の解決策は，階層の一部分だけを取り除く方法です．**図4**はその一例で，設計者は特定のブロック，あるいは階層ツリーの枝を指定します．

理想は，ほとんどの階層構造を維持しながら，最小限の変更でスネーク・パスを取り除くことでしょう．この方法