特集1 第2章 FPGA/PLD トラブル・シューティング

設計ツール活用時のトラブルと解決法

宮部秀行，山口耕作

設計ツールを活用している際に起こったトラブルと解決法を解説する．ここで紹介するものは，FPGA/PLDのアーキテクチャやデバイス内蔵機能に直接関係のない事例である．（編集部）

1 期待どおりの動作周波数が得られない

宮部秀行

設計ツール
Altera社Quartus-Ⅱ

トラブル度 ③

トラブルの状況

VHDLで階層設計を行い，論理合成後にQuartus-Ⅱで配置配線を行っています．ある設計で，最大動作周波数f_{max}が要求仕様にどうしても届かないことがありました．下位モジュールの機能を考えると，実現可能と予想される周波数なのですが…．もちろん，論理合成ツールやQuartus-Ⅱの設定をいろいろと変えてみましたが，f_{max}を満足するには至りませんでした．

〔図1〕フロアプランナで配置状況を調べる
青い部分は使用セルを意味している．

このような場合，HDLソース・コードを見直し，クリティカル・パス（もっとも遅延が大きいパス）の論理を浅くすることで，f_{max}の改善を図ります．しかし今回はクリティカル・パスをいくらつぶしても解決できません．問題なかったはずの別のパスがクリティカル・パスになってしまうのです．そのたびにHDLソース・コードを修正するはめになりました．この作業に数日を使いましたが，クリティカル・パスはなくならず，きりがありません．かりに一度だけうまくいっても，デバッグ工程における修正や機能追加の際にすんなりf_{max}を満たせるのかという不安を残します．

解決方法

●原因の究明

まずはレポート・ファイルをチェックしてみました．そして，クリティカル・パスがいくつかのモジュールに集中していることに気づきました．

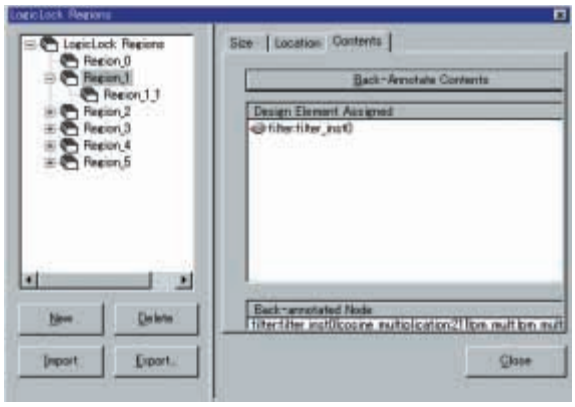
次にフロアプランナで配置状況を調べてみると，どのモジュールも散らばって配置されています（図1）．モジュールの配置をくふうできれば，HDLソース・コードの変更は少なくてすみそうです．

●LogicLockの活用

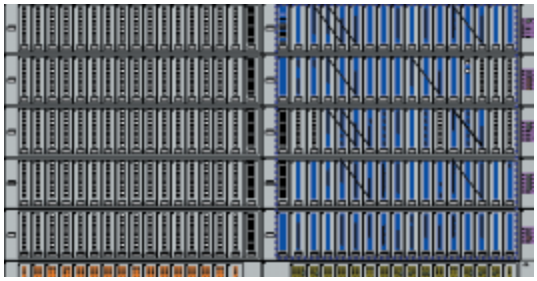
そこで，Quartus-ⅡのLogicLock機能を使用して，モジュールを固めて配置することにしました．

LogicLockは，設計者が意図するとおりにモジュール

〔図2〕LogicLock機能 ―配置領域の作成
Sizeタブで領域の大きさを指定し，Locationタブで領域の配置箇所を決める．



〔図3〕階層状の領域
Region_1の下にRegion_1_1という領域が配置されている．



〔図4〕LogicLock機能による改善後の配置のようす
モジュールを固めて配置して，クリティカル・パスをつぶした．

を配置するための機能です．配置制約の作業は敷居が高いと感じていましたが，LogicLockは設定が簡単なので，手軽に使用できます．

Quartus-Ⅱのメニューから「Tools」→「LogicLock Regions」を選択すると，配置領域設定のウィンドウが開きます．ここで[New]ボタンを押すと，配置領域を作成できます(図2)．Sizeタブで領域の大きさを指定し，Locationタブで領域の配置箇所を決めます．これらの設定は自分で決めることもできますし，ツールに任せることもできます．

続いて，作成した配置領域にモジュールを割り付けます．Quartus-ⅡのProject Navigator(モジュールを階層表示するウィンドウ)から希望のモジュールを選択し，作成した配置領域(図2であればRegion_0)へドラッグ&ドロップします．するとContentsタブにモジュール名が表示されます．あとはこの作業を繰り返し，クリティカル・パスをつぶしていきます．

領域を階層状に作成することもできます．つまり特定の階層のモジュールを固めて配置する際に，その中でさらに下位のモジュールの配置領域を指定できるということです．図3は，Region_1の下にRegion_1_1という領域が配置されているようすです．

設定がすんだら，配置配線を行います．結果をフロアプランナで確認したものが図4です．領域内のf_{max}はつねに最適化されます．モジュールを固めて配置することで，デバッグ時の設計変更による影響は，受けにくくなります．

●領域間，I/O部のタイミングが悪化

ここで新たな問題が一つ発生しました．LogicLockを使うことで，領域内については配置を最適化できましたが，領域と領域，領域とI/Oピンの接続部分は逆に速度が遅くなってしまったのです．モジュールを固めて配置すれば，接続部分が犠牲になるのは当然といえば当然です．

この問題については，次のようにHDLソースを変更することで対処しました．

領域間の接続部分は，出力側のモジュールからレジスタ経由で信号を出すようにします．入力側のモジュールでは，入力信号を一度レジスタで受けるようにしました．こうすることで領域間のパスがクリティカル・パスになる可能性は低くなります．

領域とI/Oピンの接続部分は，領域側にレジスタを挿入し，これとは別に，I/Oレジスタ用のレジスタを一つ追加します．高速なt_{su}(セットアップ時間)やt_{co}(クロック-出力時間)を実現しつつ，f_{max}も維持するためには，このような方法をとるしかありません．どちらか一方が欠けると，領域とI/Oの位置関係によっては，t_{su}/t_{co}とf_{max}のどちらかが大幅に低下する可能性があります．

モジュールの入出力信号をすべてレジスタ化することは，全体的なタイミングを考えるうえで非常にめんどうな作業となります．また，レジスタを多く使用するので，そのことによる使用率増加も気になるところです．しかし，