フロアプラン手法ガイド

UG633 (v13.1) 2011 年 3 月 1 日

本資料は英語版 (v13.1) を翻訳したもので、内容に相違が生じる場合には原文を優先します。
資料によっては英語版の更新に対応していないものがあります。
日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	改訂内容
2011 年 3 月 1 日	13.1	全体のマイナー アップデート

目次

第 1 章

フロアプランの概要

フロアプランとは、次のプロセスのことです。

- 最適なロジックのグループおよび接続を選択
- FPGA デバイスの手動でロジック ブロックを配置

フロアプランの目的は、次のとおりです。

- 集積度、配線、パフォーマンスを向上
- より良い配置を指定することにより、選択したロジックの配置遅延を削減

より大型の FPGA デバイス用に複雑なデザインを作成するためには、フロアプランが必要です。デザインの複雑性が増すのに並行して、インプリメンテーション ツールも向上しています。

デザインによっては、フロアプランを実行するとインプリメンテーション ソフトウェアで次が可能になります。

- より高いシステム クロック周波数
- インプリメンテーション ランタイムの短縮
- より一貫したタイミング
- 上記すべての利点

フロアプランの利点

優れたフロアプランの場合、次が可能になります。

- パフォーマンスの向上
- タイミングを満たすために配置配線デザインをイネーブル

ザイリンクスでは、次のような場合にフロアプランを推奨しています。

- タイミングに一貫性がない場合
- タイミングを満たしたことがない場合

フロアプランを実行するタイミング

フロアプランを実行する状況は、デザイン チームによって異なります。フロアプランは、次のような場合に実行できます。

- 配置配線を最初に実行する前
- 問題の特定後
- デザインがセットアップ タイミング制約を満たさない場合

タイミング クロージャの基礎

フロアプランは、タイミング クロージャを達成するため、パス遅延を削減する方法として導入されています。

インプリメンテーション プロセス中、ソフトウェアでは次が実行されます。

- ロジック遅延と配線遅延をタイミング制約で許容される遅延値と比較
- 「クロック ジッタおよびクロック間スキュー」を考慮
- レポートには次が示されます。
  - タイミング制約が満たされたか
  - タイミング制約が満たされていないか

図 1-1「タイミング レポートの例」に、タイミングレポートの例を示します。

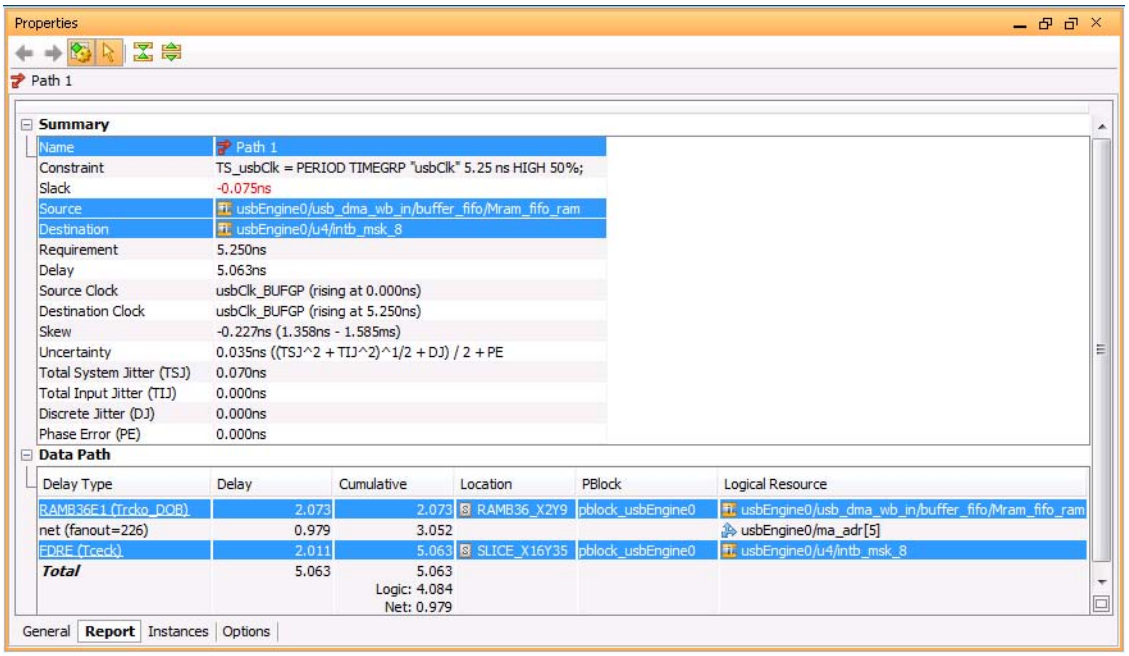

図 1-1：タイミング レポートの例

タイミング制約の確認

フロアプランでは、まず最初に次を実行します。

- タイミング制約が適切であることを確認
- パスが複数サイクル パスであるか、フォルス パスであるかを判断する必要があります。

複数サイクル パスおよびフォルス パス

デザインに、すべてのクロック サイクルでは駆動されない部分や、制御構造によりパスが到達しない部分がある場合があります。インプリメンテーション ツールではこれを判断することはできません。

ロジックを複数サイクル パスまたはフォルス パスとして指定しない限り、これらのパスのタイミングが不必要に検証されます。

デザイン ロジックに適合するよう制約を緩和すると、多くのデザインでタイミングが向上します。

複数サイクル パスおよびフォルス パスの詳細は、付録 A「その他のリソース」に示される『タイミング制約ユーザー ガイド』(UG612) を参照してください。

クロック ジッタおよびクロック間スキュー

許容される時間は、クロック ジッタおよびクロック間スキューにより変更されます。

デスティネーション クロックがソース クロックの前に立ち上がると、許容される時間が短くなり、PERIOD 制約が厳しくなることになります。

ソース クロックにジッタがある場合、ツールで許容される時間を変更する必要があります。

タイミング レポートに、これらの変更が示されます。タイミングが満たされないパスに対しては、ジッタおよびスキューが適度なものであるかを確認してください。

パス遅延の削減

タイミング制約およびクロック構造を検証した後は、パス遅延を削減することによりタイミングを満たします。パス遅延は、ロジック遅延と配線遅延に分けることができます。一方または両方の遅延を削減することが必要になります。

ロジック遅延を PERIOD 制約と比較し、ロジック遅延が許容されるパス遅延の大部分を占める場合は、パスにゲートを追加する必要があります。次のいずれかの操作を実行できます。

- RTL を変更してレジスタに追加
- 制約を変更
- 合成エンジンの設定を変更

パス遅延の大部分が配線が原因である場合、配置に問題がある可能性があります。

ファンアウトの大きいネット、ピン配置、またはその他の構造により、配置が分散されていないかどうかを確認してください。そうでない場合は、フロアプランを使用して次のいずれかを実行します。

- 配線遅延を削減
- RTL をどのように変更したらよいかを判断

フロアプランの基礎

フロアプランは、クリティカル パスの配置遅延を削減するために使用する手法です。次のいずれかの操作を実行できます。

- タイミング問題の原因となっているロジックを特定
- 配置配線ソフトウェアでロジックが近くに配置されるよう指定

目的は、配線遅延を削減してクリティカル パスのタイミングを向上させることにあります。

フロアプランでは、クリティカル パスを構成するロジックは変化しません。合成ソフトウェアでゲートが構成されるよう設定し、フロアプランがサポートされるようにする必要があります。クリティカル パスのほとんどの遅延がロジック遅延である場合は、フロアプランよりも再合成の方が有益です。

フロアプラン中に、再合成した方が良い問題が見つかることもあります。レジスタを複製し、分散されたロードのクラスタの近くに配置することをお勧めします。

適切にフロアプランしても、デザインのタイミングが満たされるとは限りません。フロアプランでは配線が修正されるわけではなく、配置シードが供給されるだけです。

絶対的なパフォーマンスよりもデザインの一貫性が重要である場合は、フロアプランと共にインクリメンタル デザイン手法を使用できます。

詳細は、付録 A「その他のリソース」に示される『階層デザイン手法ガイド』(UG748) の第 2 章にある「パーティションのフロアプラン」を参照してください。

フロアプラン方法

- 「詳細なゲート レベルのフロアプラン」

  クリティカル パスの各ロジック エレメントをチップの特定サイトに配置します。

- 「階層フロアプラン」

  階層レベルをチップの特定領域に制約付けます。

詳細なゲート レベルのフロアプラン

非常に厳しいタイミング クリティカル パスの場合は、図 1-2「手動でフロアプランされたロジック」に示すようにすべてのゲートを手動で配置することも可能です。

ただし、次のような理由から、これは最終手段としてください。

- ゲート レベルのフロアプランには時間がかかる
- 適切な配線が得られるように配置するため、デバイスに関する知識が必要となる
- 詳細なゲート レベルの配置結果は安定しておらず、合成中にゲート名が変更されると、配置が無効になることがある

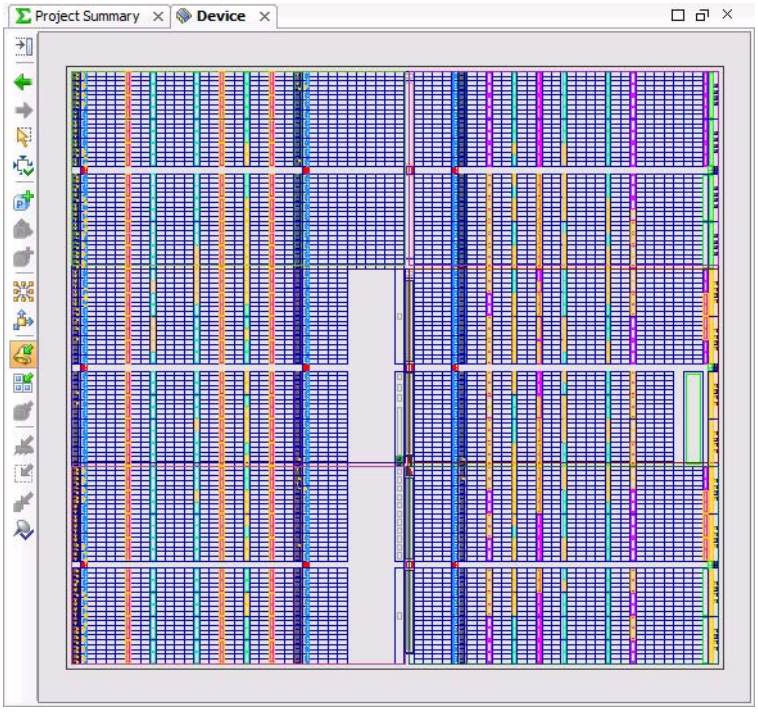

図 1-2 : 手動でフロアプランされたロジック

階層フロアプラン

ザイリンクスでは、ゲート レベル フロアプランではなく、階層フロアプランを推奨しています。階層フロアプランでは、次が可能です。

- チップの小さな領域に 1 つまたは複数の階層レベルを配置

  詳細は、図 1-3「フロアプランされた階層」を参照してください。

- 配置ツールに素早く指示

階層にはすべてのゲートが含まれているので、階層名が変更されなければ、ゲートの変更によりフロアプランが無効になることはありません。配置ツールでは、デバイスに関する詳細な情報とタイミング アークを使用して、詳細な配置を生成します。得られたフロアプランは、通常デザインの変更の影響は受けません。

図 1-3 : フロアプランされた階層

高レベル フロアプランの生成

RTL を構築中またはピン配置をインプリメント中に、高レベルのフロアプランを生成する必要がある場合もあります。高レベル フロアプランを生成すると、次が可能になります。

- デバイス全体のデータ フローを可視化
- より良い RTL およびピン配置の生成方法を理解

メモ：配置配線には使用しないでください。

次の事項に従うことをお勧めします。

- デザインを合成します。
- ピン配置制約のみを使用してインプリメンテーションを実行します。
- デザインのタイミングが満たされない場合は、高レベル フロアプランを配置配線情報と共に使用し、タイミングを向上する新しいフロアプランを作成します。

フロアプランの注意事項

- フロアプランは、通常反復作業です。

  最初のフロアプランでデザインのある部分の問題が解決されても、別の部分でタイミングが満たされなくなることがあります。

- フロアプランでタイミングが悪化することもあります。

  フロアプランでタイミングが悪化することもあります。これは、特にフロアプランすべきものが何か、どこに配置したらよいのかがはっきりしない場合に発生しやすくなります。

- 記録を取り、何回か試すことが、フロアプランでの作業に役立ちます。

タイミング クリティカルなロジックのフロアプラン

デザインのフロアプラン時には、次の推奨事項を考慮してください。

- インプリメンテーション ツールでタイミング クリティカルと判断されたロジックのみをフロアプランします。
- インプリメンテーション ソフトウェアでタイミング クリティカルと判断された下位階層から開始します。

デザイン全体のフロアプラン

ほとんどの FPGA デザインでは、インプリメンテーション ソフトウェアに入力する合成済みネットリストの形でデザイン全体のフロアプランがサポートされますが、

デザイン全体をフロアプランすることはお勧めしません。

データ フロー図に基づいてチップ全体をフロアプランすると、ほとんどの場合タイミングが悪化します。

階層ネットリストの操作

タイミング クロージャのためのフロアプランにおいて、RTL 構造が有益である場合と妨げになる場合があります。合成ソフトウェアに入力する RTL に記述されている階層をフロアプランできます。

合成ソフトウェアで階層ネットリストが生成されるように設定する必要があります。階層のないネットリストよりも、階層ネットリストの方が作業が楽です。

チップ上にデザインがどのように分散されるかを考慮して階層を構築すると、タイミングが満たされやすくなります。

似たようなメモリ インターフェイスを 2 つチップの両側に配置する必要がある場合は、RTL ソースでそれぞれのインターフェイスにファンアウトの大きい制御信号を記述します。

ほとんどの場合、合成ソフトウェアでは信号は最適に複製されません。合成でリセット フリップフロップなどのファンアウトの大きいフリップフロップが複製されると、ロードの小さいコピーが 2 つ作成され、その両方がチップ全体に分配されることがあります。

手動でレジスタを複製し、次のようなファンアウトの低いレジスタを 2 つ作成できます。

- 1 つのレジスタがチップの 1 辺のロードを駆動
- もう 1 つのレジスタが反対側の辺のロードを駆動

ロジック合成での推奨事項

- クリティカル タイミング パスが 1 つのモジュール内に制限されるよう、RTL ロジックを構築します。多数の階層モジュールにまたがるクリティカル パスはフロアプランしにくくなります。
- すべてのモジュールの出力にレジスタを付け、クリティカル パスに関連するモジュールの数を削減します。
- ダイ上で分割されるネットのドライバを複製します。論理的に等価のロジックを保持する合成属性が必要な場合があります。
- 1 つの大型階層ブロックに長いパスがあると、フロアプランが困難になります。RTL で大型階層ブロックを分割してみてください。階層ブロックが小型である方が操作が簡単です。
- 混合されたクリティカル パスはフロアプランが困難です。大型のクリティカル ブロックを小型で操作しやすいブロックに分割します。
- デザインの変更頻度が高い場合は、インクリメンタル合成を考慮します。
  - 個々のブロックを別々に合成、または
  - SYN_HIER=HARD を使用して階層を保持

  階層を保持すると、インクリメンタル フローには役立ちますが、階層をまたがるグローバル最適化を実行できないため、パフォーマンスが低下する場合があります。インクリメンタル RTL 合成を試す前に、このトレードオフを考慮してください。
- 合成エンジンで階層が再構築されるように設定するか、合成済みネットリストの階層を保持します。
  - フラット化されたネットリストは合成の面からは最適ですが、次が困難になります。
    - フロアプラン
    - 配置制約
  - 階層を再構築する合成オプションを使用してみてください。
    - XST で -netlist_hierarchy = rebuilt を使用
    - PlanAhead™ ツールはこのオプションを含むストラテジをデフォルトで使用

一貫性の向上

優れたフロアプランの場合、次が可能になります。

- デザインでより一貫した結果を得る
- QoR (結果の質) を向上
- タイミングが満たされていないデザインでタイミングが満たされる

多くの階層フロアプランは、シミュレーションおよびボード テストからの修正を組み込んだネットリストの複数のリビジョンで機能しますが、1 つのパスでタイミングが満たされているブロックが、別のパスではタイミングが満たされないこともあります。配置は配置配線での単なる指示であり、配線は固定されていません。

より高いパフォーマンスを達成するよりもデザインの一貫性が重要である場合は、インクリメンタル合成とインプリメンテーションを使用することを考慮してみてください。これらのフローを使用すると、ゲート レベル ネットリストの変更範囲が制限され、異なる実行間で配置および配線が保持されます。一貫性は向上しますが、QoR が多少低下することがあります。これらのフローを使用するかどうかは、デザインを開始した後ではなく、デザイン サイクルの開始段階で決定してください。

詳細は、付録 A「その他のリソース」に示される『階層デザイン手法ガイド』(UG748) の第 2 章「設計での考慮事項」を参照してください。

クロック リソースに焦点を置いたフロアプラン

デバイスのクロック リソースの大部分を使用するデザインでは、ロジックの配置制限は FPGA デバイス ファミリによって異なります。ロジックを配置する際は、デバイスのクロック規則を考慮してください。

PlanAhead ツールを使用すると、次が実行できます。

- 特定のクロックをチップ上の特定の領域に制約できます。
- チップ上のさまざまなクロック領域およびクロック区画をグラフィカルに表示できます。

[Clock Region Properties] または [Pblock Properties] ビューの [Statistics] タブに、AREA_GROUP 制約で定義されるすべての Pblock に含まれるクロック ネットとクロック領域が表示されます。

- クロック リソースが [Clock Resources] ビューのどこにあるか表示
- 次のクロック ネットとクロック領域を表示
  - すべての Pblock に含まれる
  - AREA_GROUP 制約で定義される

回路図ビューを表示すると、各クロック ネットに接続されているロジックおよび階層を確認できます。

第 2 章

フロアプラン フロー

次のフロアプラン フローがサポートされています。

- 「再利用フロー」

  タイミングがときどきしか満たされないデザインでタイミング クロージャを達成できます。

- 「階層フロアプラン フロー」

  タイミングがまったく満たされていなかったデザインでタイミング　クロージャを達成できます。

再利用フロー

再利用フローは、タイミングがときどきしか満たされないデザインでタイミング クロージャを達成できます。このフローでは、タイミングが満たされたインプリメンテーション実行からのブロック RAM および DSP48 の配置を、次のインプリメンテーション実行でのシードとして使用します。

再利用フローの利点と欠点

再利用フローには、次のような利点があります。

- 簡単に適用できます。
- インプリメンテーションのランタイムを短縮できます。
- タイミングが満たされる確率が向上します。
- デザインの配置にデバイスに関する知識はそれほど必要ありません。

再利用フローには、次のような欠点があります。

- デザインのタイミングがまったく満たされない場合は使用できません。
- デザインの変更が制限されます。
- タイミングが一貫して満たされるとは限りません。

再利用フローの詳細

タイミングが変動する原因の 1 つは、ブロック RAM や DSP48 などのマクロ配置です。マクロを配置すると、LUT および FF 配置のシードとなります。タイミングが満たされたインプリメンテーション実行のマクロ配置を再利用すると、インプリメンテーション実行間での変動が少なくなります。これにより、インプリメンテーション ツールでタイミングが満たされる配置を見つけて、その一部を再利用することができます。

このフローは、次のような場合に使用できます。

- デザインでときどきタイミングが満たされる。
- マクロの名前と構造が変化しない。

大型マクロの配置から、ほかのゲートの配置を特定できることがあります。タイミングは安定し、場合によってはインプリメンテーションのランタイムが短縮されます。

配線が完了し、タイミングが満たされた PAR の実行から開始してください。[Project Summary] ビューで、[Timing Score] が 0、[Unrouted] が 0 の run を見つけます。タイミングが満たされている run が複数ある場合は、インプリメンテーションのランタイムが最短のものを使用します。

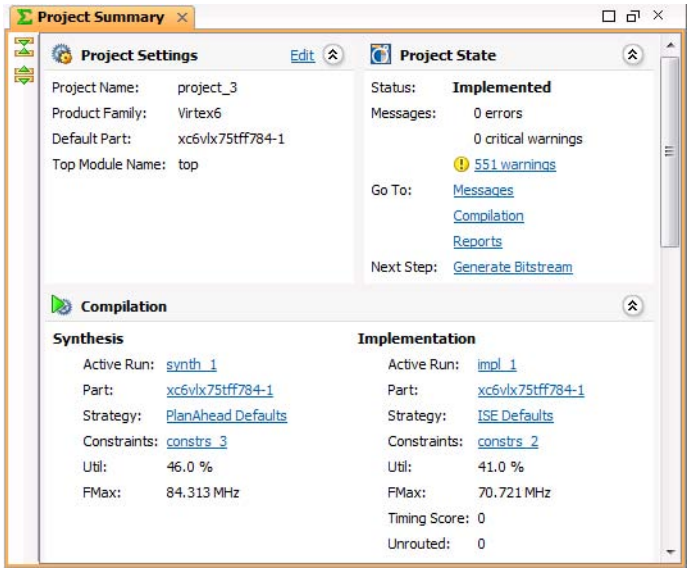

図 2-1 : [Project Summary] ビュー

インプリメンテーション run の使用

インプリメンテーション run は、次を実行すると開くことができます。

- スクリプト
- Project Navigator
- PlanAhead™ ソフトウェア

タイミングを満たしているデザインを PlanAhead に読み込んで、配置に制約を付けます。

インプリメンテーション配置の表示

インプリメンテーション ソフトウェアでどこにゲートが配置されたかを確認するには、次のいずれかの方法を使用します。

1. Project Navigator で [Analyze Timing/Floorplan Design] プロセスを実行
2. PlanAhead の Flow Navigator で [Implement Design] をクリックし、インプリメンテーション済みデザインを開く
3. インプリメンテーションをスタンドアロン スクリプトで実行した場合は、次を実行します。
   a. PlanAhead プロジェクトを新規作成

b. New Project ウィザードで [Import ISE Place & Route results] をオン

図 2-2 : インプリメンテーション配置の表示

配置の再利用

デザインでタイミングが満たされた場合、その配置を再利用することができます。デザインが変更される可能性があるので、配置されているすべてのものを固定しないでください。

多くのデザインでは、ブロック RAM および DSP48 プリミティブは比較的安定したプリミティブです。ブロック RAM と DSP48 の配置のみを再利用すると、ほかのゲートが変更された場合でもタイミングを保持できます。

プリミティブの検索

PlanAhead ソフトウェアでは、次を簡単に検索できます。

- ブロック メモリ (RAMB および FIFO プリミティブ)
- ブロック演算 (アーキテクチャにより MULT および DSP プリミティブ)

これらのプリミティブを検索するには、[Edit] → [Find] をクリックして図 2-3「メモリ ブロックおよび演算ブロックの検索」に示すダイアログ ボックスを表示します。

検索が終了すると、一致するオブジェクトがすべてリストされます。インプリメンテーション実行のすべての配置が読み込まれます。シードに必要なマクロ配置は、その他の配置から分離する必要があります。

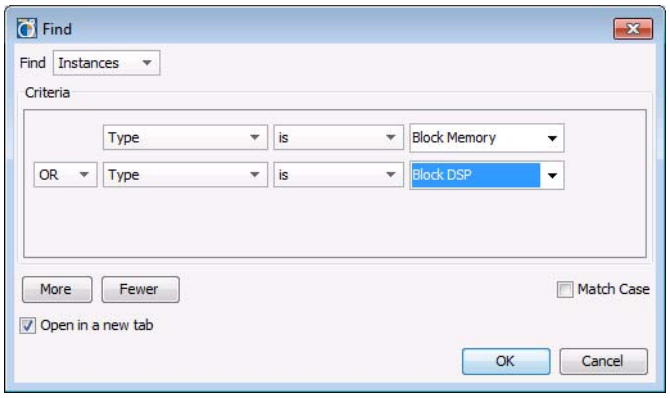

図 2-3 : メモリ ブロックおよび演算ブロックの検索

固定された配置制約および固定されていない配置制約

PlanAhead ソフトウェアには、固定されたのとされていないのの 2 種類の配置があります。

	固定された配置	固定されていない配置
作成箇所	ユーザー制約ファイル (UCF) に記述	インプリメンテーション ソフトウェアからバックアノテート
	PlanAhead ソフトウェアで ユーザーが指定	
再利用	可能	不可能

ロジックを固定するには、次の手順に従います。

1. 固定する配置済みオブジェクトをすべて選択
2. 右クリック
3. [Fix Instances] をクリック

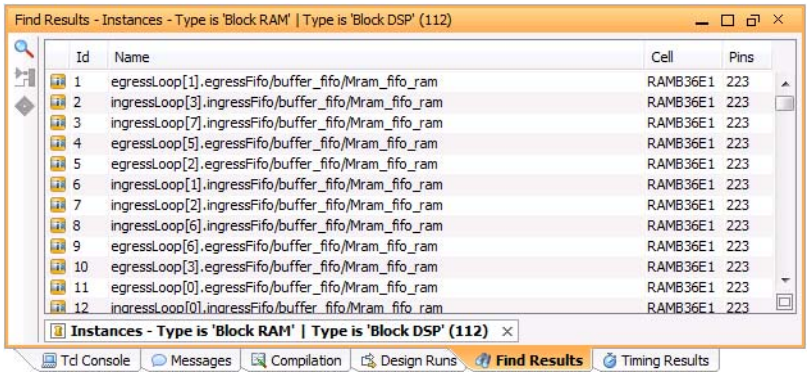

図 2-4 : [Find Results] パネルでロジックを選択

配置の解析と変更

[Device] ビューで配置されたロジックの色が変わり、配置方法の変更が示されます。

UCF はプロジェクトを保存しないと新しい制約でアップデートされません。

UCF ファイルに、次のようなゲート レベルの制約が複数記述されます。

```
INST "usbEngine0/usb_out/buffer_fifo/Mram_fifo_ram" LOC =
RAMB36_X3Y14;
INST "fftEngine/fftInst/arnd2/ct5/Maddsub_n0027" LOC = DSP48_X1Y26;
```

- ゲート レベル ネットリストで名前が変更された場合、配置を再実行して LOC 制約で定義されている参照をアップデートする必要があります。
- マクロまたはマクロ周辺のロジックが変更された場合、配置を削除して再実行する必要があります。
- デザインのタイミングが頻繁に満たされなくなった場合は、次を実行します。
  - マクロの LOC 制約なしで PAR を実行
  - 各ブロック RAM または DSP48 の配置を調整 (アドバンス ユーザー用)

配置の解析方法および変更方法の詳細は、次を参照してください。

- 付録 A「その他のリソース」に示される『PlanAhead ユーザー ガイド』(UG632) の第 10 章「インプリメンテーション結果の解析」
- 付録 A「その他のリソース」に示される『PlanAhead ユーザー ガイド』(UG632) の第 11 章「デザインのフロアプラン」

階層フロアプラン フロー

階層フロアプラン フローには、次のような特徴があります。

- 「再利用フロー」よりも効果的です。
- タイミングがまったく満たされていなかったデザインでタイミング　クロージャを達成できます。
- タイミングをより簡単に一貫して満たすためのデザインおよびロジック変更が示されます。

階層フロアプラン フローも「再利用フロー」もタイミングに大きく影響を与えます。

階層フロアプラン フローには、次のような利点があります。

- デザインの変更の影響はあまり受けません。
- タイミング クロージャを達成できます。
- 一貫性が向上します。

階層フロアプラン フローには、次のような欠点があります。

- かなりのエンジニアリング時間が必要
- 反復作業が必要なこともあり

どちらのフローでも、タイミングが大幅に向上する可能性があります。フロアプランされたロジックが低速の場合は、フロアプランを削除して別のフロアプランを試してください。フロアプランされていないロジックが低速の場合は、フロアプランしてみてください。

タイミングが満たされていないデザイン

タイミングが満たされていないデザインのタイミング クロージャには、階層フロアプラン フローを使用するのが最適です。これにより、次が実行可能です。

- 階層の小さいレベルを取り出す
- 階層レベルをチップのある領域に制約
- インプリメンテーションのガイドとして使用

これは、タイミングを満たすデザインよりも手間がかかります。

インプリメンテーションでは、次が実行されます。

- クリティカル パスおよびチップの構造に関する詳細な情報が適用されます。
- 通常配置の微調整はうまく機能します。
- 大型のフラット デザインの大まかな配置では常によい結果が得られるとは限りません。

インプリメンテーション後にタイミングが満たされていないゲートを含む階層の大まかな配置を指定すると、インプリメンテーションを向上できる場合があります。

最終的なピン配置がわかっていると、フロアプランしやすくなります。I/O に接続されるブロックは、I/O の近くに配置するのが適切なことがほとんどです。ピン配置によりタイミング クリティカル パスが引き離されているかどうかは、フロアプランで明らかになります。このような問題を早期に発見すれば、ピン配置を変更してタイミング クロージャを向上することが可能です。

各行で、チップ上での形と場所が定義されます。

次の操作を実行できます。

- これらの一部のみを制約する領域を設定します。
- 次の制約を使用すると、ブロック RAM のみをチップ上のサイトに制約できます。

```
INST "usbEngine1" AREA_GROUP = "pblock_usbEngine1";
AREA_GROUP "pblock_usbEngine1" RANGE=RAMB18_X0Y24:RAMB18_X2Y47;
AREA_GROUP "pblock_usbEngine1" RANGE=RAMB36_X0Y12:RAMB36_X2Y23;
```

この場合、スライスと DSP は制約されません。

第 3 章

タイミング クロージャでのフロアプランの使用

フロアプランを作成する際、次の事項を考慮する必要があります。

- どのようなタイミング エラーが発生しているのか。
- クリティカルな階層はどれか。
- フロアプランまたはロジックを変更するだけでタイミング クロージャを達成できるか。
- フロアプランすべきものがほかにあるか。
- クリティカルな階層をフロアプランできるか。
- 何をどこに配置すればよいか。

これらは、次の情報から判断できます。

- タイミング パス、パス上のロジックの配置と構造
- 次に示す例で説明するようなピン配置およびデザインに関する情報

配置配線結果

タイミングを満たしていないロジックを特定するには、インプリメンテーション後のタイミング結果を参照する必要があります。デザインをインプリメンテーションまで実行してタイミングが満たされない場合は、結果を PlanAhead ソフトウェアに読み込みます。配置およびタイミングの結果とゲートをすべて 1 つの場所で表示できます。複数のクリティカル パスを選択し、配置を表示できるので、トラブルシューティングに役立ちます。詳細は、図 3-1「タイミングを満たしていないパスの配置」を参照してください。

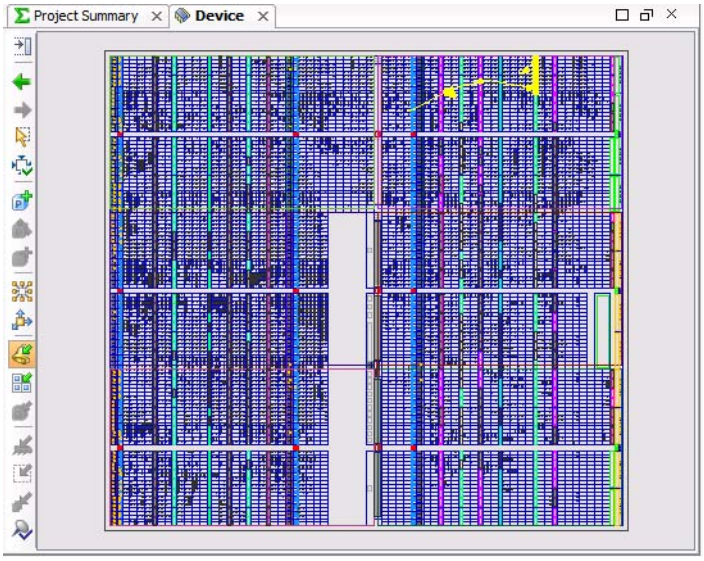

図 3-1 : タイミングを満たしていないパスの配置

図 3-1 は、クリティカル パスを含むブロック RAM が、チップ上で必要以上に分散しているところを示しています。フロアプランを使用すると、これらを近づけて配置できます。タイミングの問題は、ブロック RAM 間のパスで発生しています。これらのパスは、フロアプランの対象となります。

タイミング結果

ブロック RAM 間のパスを解析すると、次が必要であるかどうかを判断するのに役立ちます。

- フロアプラン
- ロジックの変更
- フロアプランとロジックの変更の両方

上記のクリティカル パスのパス遅延では、配線遅延の長いネットが 2 つ示されています。詳細は、図 3-2「詳細なデータ パス」を参照してください。パスは、837ps 以上の差でタイミングを満たしていません。3 番目のネットの配線遅延は 1.177ns です。4 番目のネットの配線遅延は 0.958ns です。配置を向上することによって配線遅延を削減できます。

Data Path

Delay Type	Delay	Cumulative	Location	Logical Resource
RAMB36E1 (Trcko_DOB)	2.073	2.073	RAMB36_X2Y13	usbEngine0/usb_dma_wb_in/buffer_fifo/Mram_fifo_ram
net (fanout=1118)	0.886	2.959		usbEngine0/ma_adr[2]
LUT3 (Tilo)	0.196	3.155	SLICE_X30Y60	usbEngine0/u4/Mmux_adr[6]_dout[31]_wide_mux_97_OUT1817
net (fanout=1)	0.236	3.391		usbEngine0/u4/Mmux_adr[6]_dout[31]_wide_mux_97_OUT1816
LUT6 (Tilo)	0.068	3.459	SLICE_X30Y60	usbEngine0/u4/Mmux_adr[6]_dout[31]_wide_mux_97_OUT1818
net (fanout=1)	1.177	4.636		usbEngine0/u4/Mmux_adr[6]_dout[31]_wide_mux_97_OUT1817
LUT3 (Tilo)	0.205	4.841	SLICE_X30Y39	usbEngine0/u4/Mmux_adr[6]_dout[31]_wide_mux_97_OUT1819
net (fanout=1)	0.958	5.799		usbEngine0/u4/Mmux_adr[6]_dout[31]_wide_mux_97_OUT1818
LUT5 (Tas)	0.070	5.869	SLICE_X29Y57	usbEngine0/u4/Mmux_adr[6]_dout[31]_wide_mux_97_OUT1820
FDE	0.000	5.869	SLICE_X29Y57	usbEngine0/u4/dout_14
Total	5.869	5.869 Logic: 2.612 Net: 3.257		

図 3-2 : 詳細なデータ パス

階層フロアプランにより、クリティカル ロジックの配線遅延を削減できます。ロジック遅延により、パフォーマンスが制限されます。デザインのロジック遅延の割合が高い場合、コードを変更するか合成をアップデートしてゲートを変更します。

ゲートおよび階層

LOC および配置制約を使用することによりゲートを個別にフロアプランできますが、ザイリンクスでは、次の理由から、タイミングを向上するためにゲートを手動で移動することはお勧めしません。

- ゲートを特定し、配置するのは時間がかる困難な作業であるため
- フロアプランしたゲートのロジックが変更された場合、フロアプランも再実行する必要があるため

その代わり、どの階層がタイミング クリティカルであるかを特定します。図 3-2「詳細なデータ パス」には、usbEngine1 にタイミングの問題があることがレポートされています。この階層レベルまたはそのサブ階層が階層フロアプランの対象となります。デザインを解析して、どの階層をフロアプランすべきかを判断する必要があります。

まず、クリティカル パスを回路図に表示してみてください。図 3-3「クリティカル パスのゲートと階層」に示すように、回路図には次が表示されます。

- クリティカル パスが含まれているゲート
- そのゲートが配置されている階層

回路図でクリティカル ゲートの周辺のロジックを確認し、クリティカルでないロジックの構造を調べます。

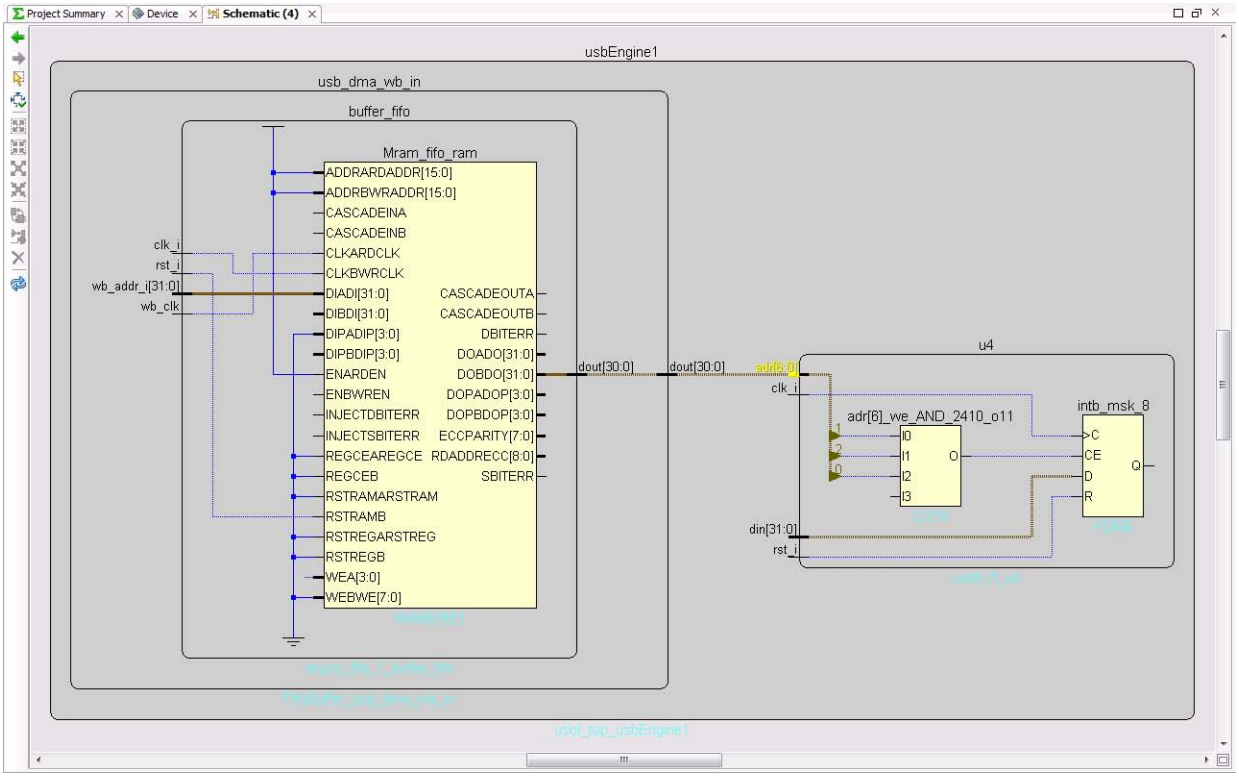

図 3-3 : クリティカル パスのゲートと階層

フロアプランでは、少なくとも usbEngine 内のブロック RAM に関連するタイミング クリティカル パスを制約する必要があります。usbEngine ブロックがフロアプランのよい対象であると判断できます。usbEngine ブロックがチップの大部分を占めている場合は、クリティカル パスを含むサブ階層レベルをフロアプランします。

クリティカルな階層とクリティカルでない階層

どのゲートをフロアプランするかを判断するには、[Device] ビューで配置を表示します。図 3-4「usbEgine のクリティカル パスとクリティカルでないパス」で次を実行してください。

- クリティカルな階層のゲートは赤色で示されています。
- クリティカルでない階層のゲートは黄色で示されています。

図 3-4 からは、次のことがわかります。

- クリティカルな階層ではブロック RAM の使用率が高い
- クリティカルでない階層にはブロック RAM 間に配置可能な LUT/FF ロジックが多数含まれる
- この階層は、デザインの約 20% を占めています。

図 3-4 : usbEgine のクリティカル パスとクリティカルでないパス

usbEngine1 をフロアプランする前に、ピン配置とデザインの接続を解析します。**usbEngine1** がフロアプランのよい対象ではないと示される場合もあります。

フロアプランが有益かどうかの判断

次を実行します。

- usbEngine1 のフロアプランが有益であるかどうかを確認します。
- どこに配置するべきかを判断
- デバイス上に最上位フロアプランを作成 (オプション)

最上位フロアプランは、どのロジックがほかのロジックの配置に影響を与えているかを理解するのに役立ちます。チップ全体に分散されているブロックは、フロアプランには適しません。

図 3-5、「解析用の最上位フロアプラン」で次を実行してください。

- I/O 接続は、緑色の線で表示されます。

  チップの左側中央の I/O バンクから中央の黄色のロジックに接続されている線があります。

- 階層ブロック間の接続は、ネットの束で表されます。黄色でハイライトされるブロックには、次のような特徴があります。
  - ほとんどのほかのブロックと通信します。
  - チップ上に広がるのでフロアプランには向きません。

  図からは、内部接続された階層が多くあることがわかります。ピン配置により線がチップを横切って階層に接続されている場合にもそれがわかります。

図 3-5 は、このデザインの最上位デザインを示します。1 つの階層のみがチップを横切っているのがわかります。右半分を横切る接続を持つ階層もあります。このピン配置では、usbEngine1 のフロアプランが可能です。このピン配置から、usbEngine1 (真ん中の四角) はデバイスの左上に配置する必要があることがわかります。

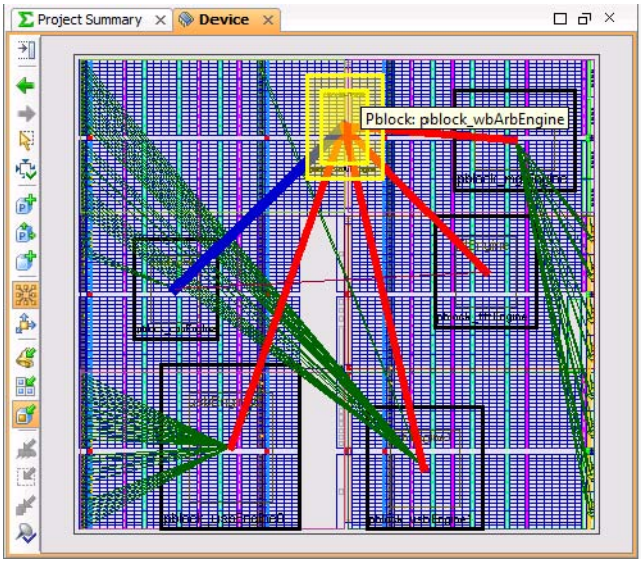

図 3-5 : 解析用の最上位フロアプラン

クリティカルな階層のフロアプランの成形

最上位フロアプランにより、クリティカルな階層を左上に配置する必要があることがわかりました。デザイン解析から、クリティカルな階層では複数のブロック RAM サイトが使用されています。また、ピン配置から、クリティカルな階層はチップの左上の 2 つの I/O バンクに接続されています。このロジックを、これらのバンクの間にあるスライスとブロック RAM を使用するようフロアプラ

ンするのが適切です。ブロック RAM (または DSP) の 100% を使用し、スライスの 80% を使用するようブロックのサイズを調整します。

ほかにフロアプランが必要かどうかの判断

このデザインには、同じゲートが 2 つあります。

- usbEngine1
- usbEngine0

インプリメンテーションで usbEngine0 にタイミング問題があることが示されましたが、usbEngine1 でもタイミング問題が発生すると予測されます。

この問題を回避するには、次を実行します。

- 各ブロックのタイミング問題を別々に解決する必要があります。
- 両方の USB ブロックを 2 つの個別のタイミング クリティカルな階層であると考えます。
- 各階層を個別にフロアプランします。

タイミングを満たす最終的なフロアプランを図 3-6「最初のフロアプラン」に示します。

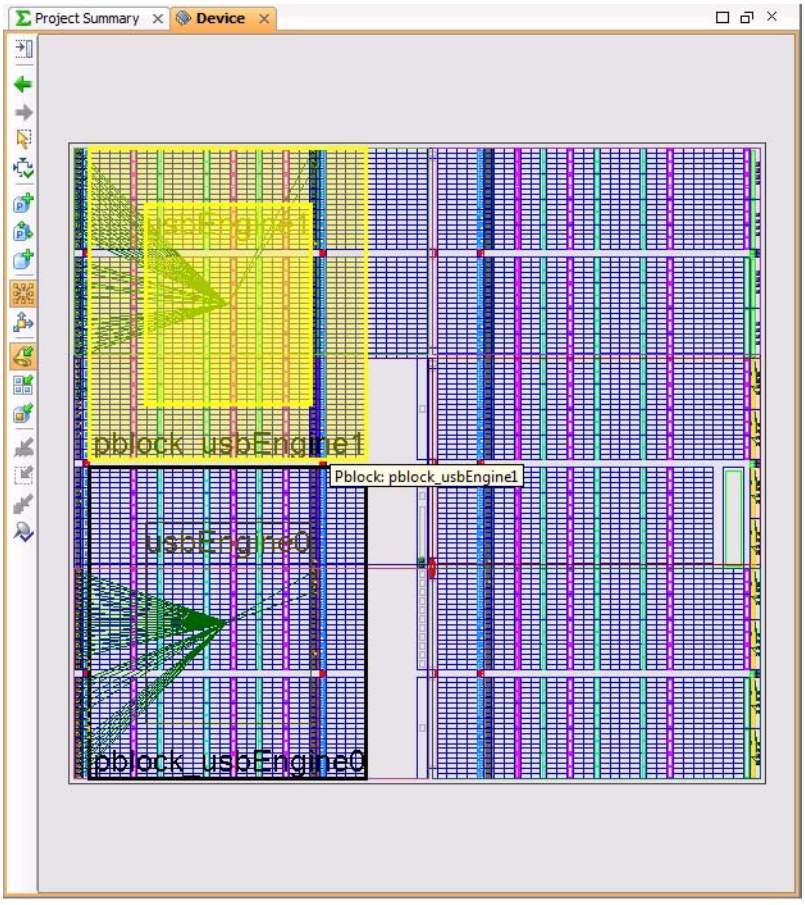

図 3-6 : 最初のフロアプラン

ネットリスト階層のサブセットの制約

PlanAhead には、ネットリスト階層の任意のサブセットをチップ上の特定の領域に制約するためのコンストラクトがあります。このコンストラクトを作成するには、[New Pblock] と [Assign] コマンドを使用します。

Pblock は UCF に AREA_GROUP 制約として記述されてインプリメンテーションで使用され、階層をチップ上のさまざまな領域に配置します。

```
INST "usbEngine1" AREA_GROUP = "pblock_usbEngine1";
AREA_GROUP "pblock_usbEngine1" RANGE=SLICE_X0Y60:SLICE_X43Y119;
AREA_GROUP "pblock_usbEngine1" RANGE=DSP48_X0Y24:DSP48_X2Y47;
AREA_GROUP "pblock_usbEngine1" RANGE=RAMB18_X0Y24:RAMB18_X2Y47;
AREA_GROUP "pblock_usbEngine1" RANGE=RAMB36_X0Y12:RAMB36_X2Y23;
```

各行で、チップ上での形と場所が定義されます。

次の操作を実行できます。

- これらの一部のみを制約する領域を設定します。
- 次の制約を使用すると、ブロック RAM のみをチップ上のサイトに制約できます。

```
INST "usbEngine1" AREA_GROUP = "pblock_usbEngine1";
AREA_GROUP "pblock_usbEngine1" RANGE=RAMB18_X0Y24:RAMB18_X2Y47;
AREA_GROUP "pblock_usbEngine1" RANGE=RAMB36_X0Y12:RAMB36_X2Y23;
```

この場合、スライスと DSP は制約されません。

第 4 章

反復フロアプラン

フロアプランは、反復作業です。

- フロアプランすべき階層がはっきりしない場合は、タイミングが向上するまでいろいろ試してみてください。
- フロアプランしたブロックでタイミングが悪化した場合は、理由を解明します。デザインに、すぐには明確にならなかった接続がある可能性があります。
- 最初のフロアプランの後、フロアプランを修正する必要がある場合があります。
- 作成したフロアプランを後で参照できるように、各フロアプランを保存しておくと便利です。
- フロアプランはなるべくシンプルにします。通常、シンプルなフロアプランの方が短時間でよい結果が得られます。
- クリティカル パスがフロアプランしないロジックに含まれている場合は、次を実行します。
  - クリティカル パスを含む階層レベルを特定
  - そのレベルを新規 Pblock に割り当て
  - その Pblock をチップ上に配置
  - 配置が適当である場合は、この Pblock を配置配線に使用
- クリティカル パスが 1 つの Pblock 内に含まれる場合は、Pblock を修正します。Pblock 内にタイミングを満たさないパスを含む Pblock を作成し、クリティカルな階層をより狭い範囲に制約するようにします。また、下位階層で作業し、一部のロジックを削除して小型の Pblock を使用します。
- クリティカル パスが Pblock と制約されていない階層の間にある場合は、次のいずれかの方法で Pblock に制約されていないロジックを追加します。
  - クリティカル パスを含む Pblock を作成して近くに配置します。
  - 制約されていないロジックが小型である場合は、クリティカル パスと制約されていないロジックの両方を含む Pblock を作成します。
- クリティカル パスが 2 つの Pblock の間にある場合は、次を実行します。
  - 移動したり形を変更したりして、Pblock 同士が近くに配置されるようにします。
  - 一方の Pblock をもう一方の Pblock に組み込みます。
  - ブロックを一方の Pblock からもう一方の Pblock に移動します。
- クリティカルな階層のロジックが大型の場合、接続が多数の場合、またはロードが分散されているために配線がチップ全体に広がっている場合は、この階層をまずは配置せず、タイミング クリティカルな階層で接続が適切なものから開始してください。この階層で継続して問題が発生する場合は、後の反復作業で検討します。パスのタイミング問題が解決しない場合は、RTL を検証して再合成してみてください。

- デザインの一部をフロアプランし、タイミングが頻繁に満たされない場合は、フロアプラン制約を削除して何が発生しているかを調べてみます。これでタイミングが向上しない場合は、別の方法を試します。新しい方法が明らかになる場合もあります。
- ISE® Design Suite のリリースをアップグレードする場合は、制約を設定せずにデザインのインプリメンテーションを実行します。新しいリリースではフロアプランが不要な場合もあります。

付録 A

その他のリソース

ザイリンクス リソース

- 『ISE® Design Suite 13 : インストールおよびライセンス ガイド』(UG798) : http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/iil.pdf
- 『ISE Design Suite 13 : リリース ノート ガイド』(UG631) : http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/irn.pdf
- ザイリンクス資料 : http://japan.xilinx.com/support/documentation.htm
- ザイリンクス用語集 : http://japan.xilinx.com/support/documentation/sw_manuals/glossary.pdf
- ザイリンクス サポート : http://japan.xilinx.com/support.htm
- データシート : http://www.xilinx.com/support/documentation/data_sheets.htm

ISE 資料

- ISE マニュアル : http://japan.xilinx.com/support/documentation/dt_ise13-1.htm
  - 『コマンド ライン ツール ユーザー ガイド』(UG628) : http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/devref.pdf
  - 『制約ガイド』(UG625) : http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/cgd.pdf
  - 『合成/シミュレーション デザイン ガイド』(UG626) : http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/sim.pdf
  - 『XST ユーザー ガイド (Virtex-4、Virtex-5、Spartan-3 および CPLD 用)』(UG627) : http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/xst.pdf
  - 『XST ユーザー ガイド (Virtex-6、Spartan-6、および 7 シリーズ用)』(UG627) : http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/xst_v6s6.pdf

PlanAhead 資料

- PlanAhead 資料 :
  http://japan.xilinx.com/support/documentation/dt_planahead_planahead13-1_userguides.htm
- 手法ガイド :
  http://japan.xilinx.com/support_documentation
  - 『フロアプラン手法ガイド』(UG633) :
    http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/Floorplanning_Methodology_Guide.pdf
  - 『階層デザイン手法ガイド』(UG748) :
    http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/Hierarchical_Design_Methodology_Guide.pdf
  - 『ピン配置手法ガイド』(UG792) :
    http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/ug792_pinplan.pdf
  - 『PlanAhead ユーザー ガイド』(UG632) :
    http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/PlanAhead_UserGuide.pdf
  - 『PlanAhead Tcl コマンド リファレンス ガイド』(UG789) :
    http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/ug789_tcl_commands.pdf
- PlanAhead チュートリアル :
  http://japan.xilinx.com/support/documentation/dt_planahead_planahead13-1_tutorials.htm
  - 『デザイン解析およびフロアプラン』(UG676) :
    http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/PlanAhead_Tutorial_Design_Analysis_Floorplan.pdf
  - 『I/O ピンの配置』(UG674) :
    http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/PlanAhead_Tutorial_IO_Pin_Planning.pdf
  - 『デザイン保持の利用』(UG747) :
    http://japan.xilinx.com/support/documentation/sw_manuals/xilinx13_1/PlanAhead_Tutorial_Design_Preservation.pdf