

---

# フロアプラン設計手法 ガイド

UG633 (v14.1) 2012 年 4 月 24 日



Xilinx is disclosing this user guide, manual, release note, and/or specification (the “Documentation”) to you solely for use in the development of designs to operate with Xilinx hardware devices. You may not reproduce, distribute, republish, download, display, post, or transmit the Documentation in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Xilinx expressly disclaims any liability arising out of your use of the Documentation. Xilinx reserves the right, at its sole discretion, to change the Documentation without notice at any time. Xilinx assumes no obligation to correct any errors contained in the Documentation, or to advise you of any corrections or updates. Xilinx expressly disclaims any liability in connection with technical support or assistance that may be provided to you in connection with the Information.

THE DOCUMENTATION IS DISCLOSED TO YOU “AS-IS” WITH NO WARRANTY OF ANY KIND. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DOCUMENTATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS. IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOSS OF DATA OR LOST PROFITS, ARISING FROM YOUR USE OF THE DOCUMENTATION.

© Copyright 2011 Xilinx Inc. All Rights Reserved. XILINX, the Xilinx logo, the Brand Window and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners. The PowerPC name and logo are registered trademarks of IBM Corp., and used under license. All other trademarks are the property of their respective owners.

本資料は英語版 (v14.1) 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。

この資料に関するフィードバックおよびリンクなどの問題につきましては、[jpn\\_trans\\_feedback@xilinx.com](mailto:jpn_trans_feedback@xilinx.com) までお知らせください。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。

## 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
2012 年 4 月 24 日	14.1	<ul style="list-style-type: none"><li>図をアップデート</li><li>テキストのマイナーな修正</li></ul>

# 目次

---

改訂履歴.....	2
<b>第 1 章：フロアプランの概要</b>	
フロアプランについて.....	5
タイミング クロージャ ..... 6	6
フロアプランの基礎 ..... 8	8
フロアプランでの注意事項..... 11	11
階層ネットリストの操作..... 11	11
ロジック合成での推奨事項..... 12	12
一貫性の向上およびフロアプランのその他の利点..... 13	13
クロック リソースに焦点を置いたフロアプラン ..... 13	13
<b>第 2 章：フロアプラン フロー</b>	
再利用フロー (デザインでときどきタイミングが満たされる場合)..... 15	15
階層フロアプラン フロー (デザインでタイミングが満たされない場合)..... 20	20
<b>第 3 章：タイミング クロージャでのフロアプランの使用</b>	
フロアプランを使用する際の質問..... 23	23
配置配線結果..... 23	23
タイミング結果..... 24	24
ゲートおよび階層 ..... 25	25
クリティカルな階層のフロアプランの成形 ..... 28	28
ほかにフロアプランが必要かどうかの判断 ..... 29	29
<b>第 4 章：フロアプランの反復実行</b>	
一般的な推奨事項 ..... 31	31
クリティカル パスの修正 ..... 32	32
クリティカルな階層でのタイミングの向上 ..... 33	33
<b>付録 A：その他のリソース</b>	
ザイリンクス リソース ..... 35	35
ISE 資料..... 35	35
PlanAhead 資料..... 36	36



# フロアプランの概要

---

この章では、フロアプランの概要を示します。

## フロアプランについて

フロアプランを実行すると、次のことが可能です。

- デザインのロジックの最適なグループおよび接続を選択
- FPGA デバイスにロジック ブロックを手動配置

## フロアプランの目的

フロアプランの目的は、次のとおりです。

- 集積度、配線性、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線遅延を削減

## フロアプランの利点

フロアプランの利点は次のとおりです。

- 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済みデザインを利用
- 次を達成可能
  - 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

## フロアプランの制限

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。

## フロアプランを実行するタイミング

次のような場合にフロアプランを推奨します。

- タイミングに一貫性がない場合

詳細は、第 2 章「フロアプラン フロー」の「再利用フロー (デザインでときどきタイミングが満たされる場合)」を参照してください。

または

- タイミングがまったく満たされない場合

詳細は、第 2 章「フロアプラン フロー」の「階層フロアプラン フロー (デザインでタイミングが満たされない場合)」を参照してください。

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

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ます。

インプリメンテーション中、ツールでは次が実行されます。

1. 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します。
2. クロック ジッターおよびクロック間スキューを考慮します。
3. レポートに次を示します。
  - パスでタイミング制約がどの程度満たされたか
  - パスでタイミング制約がどの程度満たされていないか

## タイミング レポートの例

**Properties**

Path 1

**Summary**

Name	Path 1
Constraint	TS_usbClk = PERIOD TIMEGRP "usbClk" 5.25 ns HIGH 50%;
Slack	-0.075ns
Source	usbEngine0/usb_dma_wb_in/buffer_fifo/Mram_fifo_ram
Destination	usbEngine0/u4/intb_msk_8
Requirement	5.250ns
Delay	5.063ns
Source Clock	usbClk_BUFPG (rising at 0.000ns)
Destination Clock	usbClk_BUFPG (rising at 5.250ns)
Skew	-0.227ns (1.358ns - 1.585ns)
Uncertainty	0.035ns ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
Total System Jitter (TSJ)	0.070ns
Total Input Jitter (TIJ)	0.000ns
Discrete Jitter (DJ)	0.000ns
Phase Error (PE)	0.000ns

**Data Path**

Delay Type	Delay	Cumulative	Location	PBlock	Logical Resource
RAMB36E1 (Trcko_DOB)	2.073	2.073	RAMB36_X2Y9	pblock_usbEngine0	usbEngine0/usb_dma_wb_in/buffer_fifo/Mram_fifo_ram
net (fanout=226)	0.979	3.052			usbEngine0/ma_adr[5]
EDRE (Trcek)	2.011	5.063	SLICE_X16Y35	pblock_usbEngine0	usbEngine0/u4/intb_msk_8
<b>Total</b>	<b>5.063</b>	<b>5.063</b>			
		Logic: 4.084			
		Net: 0.979			

General **Report** Instances Options

図 1-1 : タイミング レポートの例

## タイミング制約の確認

フロアプランの前に、タイミング制約が適切であることを確認します。

## パスが複数サイクルパスでないか、フォルス パスでないか

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、これらを判断できません。

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 クロージャ ユーザー ガイド』(UG612)を参照してください。

## クロック ジッターおよびクロック間スキュー

許容される時間は、次のものにより変化します。

- クロック ジッター
- クロック間スキュー

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 クロックにジッターがある場合、ツールで許容される時間が変更されます。

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、ジッターおよびスキューが許容範囲内であるかどう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、パスのゲートを変更する必要があります。次の操作を実行できます。

- RTL を変更してレジスタを追加
- 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遅延である場合、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

次に、フロアプランの基本的な原理を示します。

### パス遅延の削減

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できます。次が可能です。

- タイミング問題の原因となっているロジックを特定
- 配置配線ツール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です。

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更されません。合成ツール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なことがあります。

フロアプラン中に、再合成が有益である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ーの近くに配置することをお勧めします。

### インクリメンタル デザイン

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『階層デザイン設計手法ガイド』(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。



## 詳細なゲート レベルのフロアプラン

詳細なゲート レベルのフロアプランでは、クリティカル パスの各ロジック エLEMENTをチップ上の特定サイトに配置します。

タイミング クリティカル パスのゲートの一部またはすべてを手動で配置できます。次の図では、配置されたロジックをオレンジ色で示しています。

詳細なゲート レベルのフロアプランは、次の理由から、最終手段としてのみ使用するようにしてください。

- 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、デバイスの詳細な知識が必要となる。
- 配置結果は安定しておらず、合成中にゲートや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

階層フロアプランを使用して、階層レベルをチップの特定領域に制約します。

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。
- 配置ツールに簡単に指示

階層にはすべてのゲートが含まれています。

- ・ 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

次を実行中に、高レベルのフロアプランを生成する必要がある場合があります。

- ・ RTL を構築中
- ・ ピン配置をインプリメント中

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

**注記：**配置配線には使用しないでください。

次のようにする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。
- ・ 配置配線情報と共に高レベル フロアプランを使用し、デザインのタイミングが満たされない場合は、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。

## フロアプランでの注意事項

次に、フロアプランを実行する際の注意事項を示します。

### フロアプランは通常反復作業

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。

### フロアプランによりタイミングが悪化することがある

フロアプランでタイミングが悪化することがあります。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、複数回試す

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をフロアプランする

デザインのフロアプラン時には、次の推奨事項に従います。

- ・ 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 ツールでタイミング クリティカルと判断された下位階層から開始します。

### デザイン全体をフロアプランしない

ほとんどの FPGA デザインでは、インプリメンテーション ツールに合成済みネットリストの形で入力されるので、デザイン全体のフロアプランがサポートされます。

ただし、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デザイン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

階層ネットリストを操作する場合、次の点を考慮してください。

1. 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。合成ツールに入力する RTL に記述されている階層をフロアプランできません。
2. 合成ツールで階層ネットリストが生成されるように設定します。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。
3. 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。
4. 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。

5. ほとんどの場合、合成ツール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。
6. レジスタを手動で複製し、次のようなファンアウトの小さいレジスタを 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

次のロジック合成での推奨事項に従ってください。

1. 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。クリティカル パスが多数の階層モジュールにまたがっていると、フロアプランしにくくなります。
2. 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。
3. ダイ上で分割されるネットのドライバーを複製します。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。
4. 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ることがあります。階層ブロックが小型である方が操作が簡単なので、RTL で大型階層ブロックを分割してみてください。
5. クリティカル パスが混合していると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てください。
6. 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。
  - 個々のブロックを個別に合成
  - **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ことがあります。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。

7. 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
    - 配置の制約
  - 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 では **-netlist\_hierarchy = rebuilt** を使用
    - PlanAhead™ ツールでは、デフォルトでこの合成オプションを使用

## 一貫性の向上およびフロアプランのその他の利点

適切なフロアプランを実行すると、次が可能になります。

- デザイン結果がより一貫したものになる。
- QoR (結果の品質) が向上する。
- 満たされていなかったデザインのタイミングが満たされるようになる。

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のトレードオフを考慮してください。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 run の間で配置および配線が保持されます。これにより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。どのフローを使用するかは、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『階層デザイン設計手法ガイド』(UG748) の第 2 章「設計に関する考慮事項」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、次が可能になります。

- 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、次が表示されます。

- クロック リソースが [Clock Resources] ビューのどこにあるか
- 次のクロック ネットとクロック領域
  - 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。



# 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 (デザインでときどきタイミングが満たされる場合)
- 階層フロアプラン フロー (デザインでタイミングが満たされない場合)

どちらのフローでも、タイミングを大幅に向上できる可能性があります。

## 再利用フロー (デザインでときどきタイミングが満たされる場合)

再利用フローを使用すると、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 `run` のブロック RAM および DSP48 コンポーネントの配置を、次のインプリメンテーション `run` 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 コンポーネント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 `run` のマクロ配置を再利用すると、インプリメンテーション `run` の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、次の両方の条件が満たされる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。タイミングがより安定し、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。[Design Runs] 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。

Name	Part	Constraints	Strategy	Status	Progress	Start	Elapsed	Util (%)	FMax (MHz)	Timing Score	Unrouted
synth_1	xc7k701fpg676-2	constrs_4	PlanAhead Defaults (XST 14)	XST Complete!	100%	4/22/12 9:42 PM	00:07:39	53	112.648	0	0
impl_1 (active)	xc7k701fpg676-2	constrs_2	ISE Defaults (ISE 14)	PAR Complete!	100%	4/23/12 12:03 AM	00:26:07	46	57.270	0	0
impl_2	xc7k701fpg676-2	constrs_2	ParrigIEffort (ISE 14)	PAR Complete!	100%	4/22/12 11:55 PM	00:15:02	48	50.51	0	0

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、次のもので使用できます。

- スクリプト
- Project Navigator
- PlanAhead™ ツール

タイミングを満たしているデザインを PlanAhead ツールに読み込んで、配置を制約します。

## 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 ツール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。

- Project Navigator で [Analyze Timing/Floorplan Design] プロセスを実行
- PlanAhead ツールの Flow Navigator で [Implemented 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
- 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 ツール 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



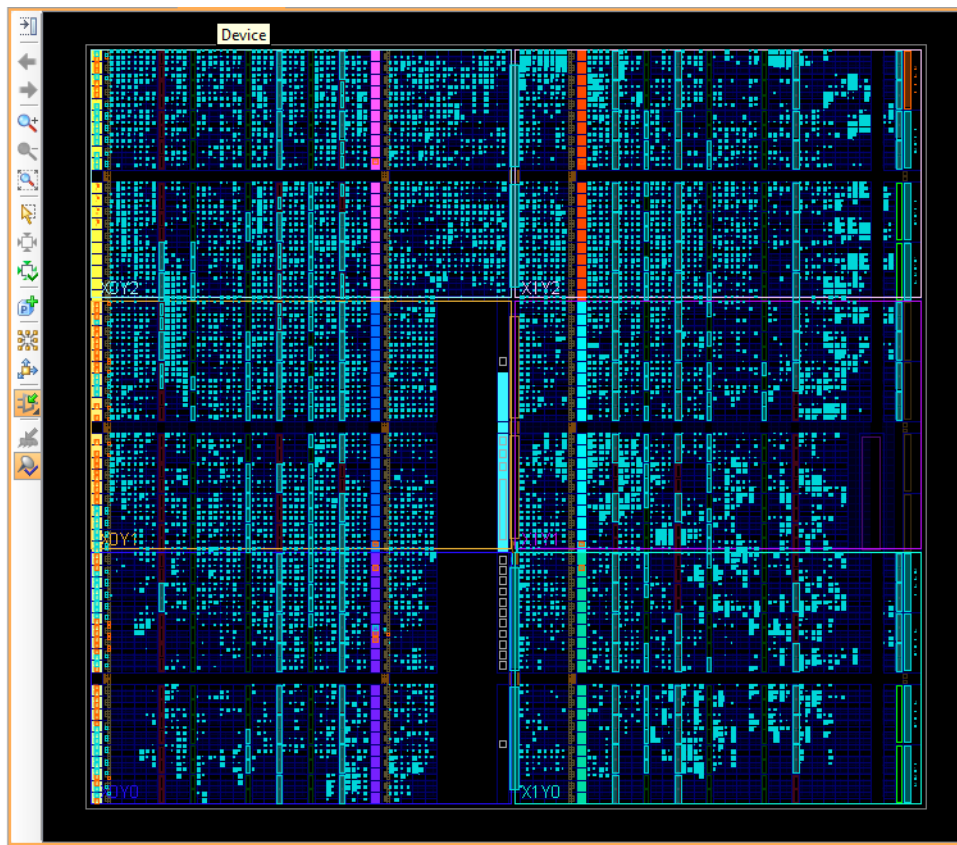


図 2-2 : インプリメンテーション配置の表示

## 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。デザインは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 プリミティブ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 ツールでは、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 (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] をクリックします。次の図を参照してください。

検索すると、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 run 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。

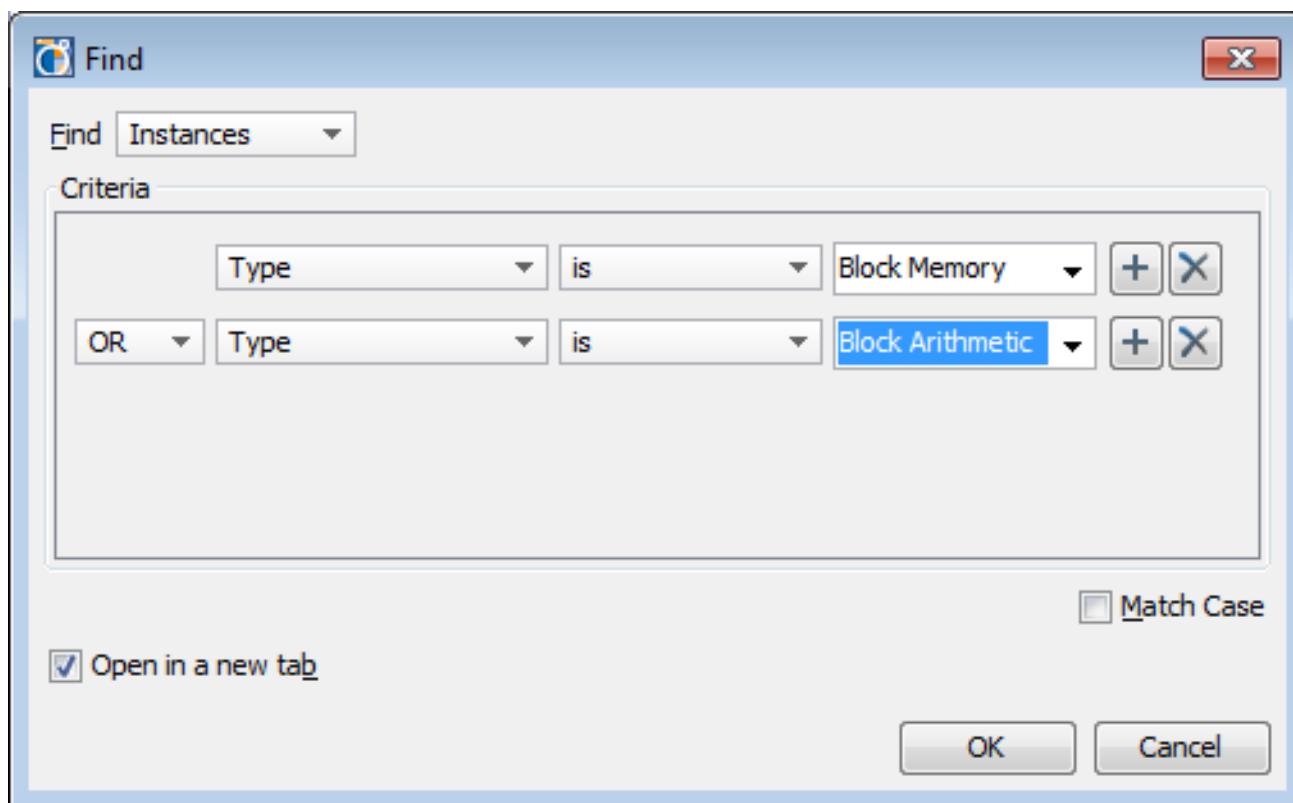


図 2-3：メモリ ブロックおよび演算ブロックの検索

## 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 ツールには、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

表 2-1：配置タイプ

	固定された配置	固定されていない配置
作成場所	<ul style="list-style-type: none"> <li>• ユーザー制約ファイル (UCF) からの配置</li> <li>• PlanAhead ツールでユーザーが指定</li> </ul>	<ul style="list-style-type: none"> <li>• インプリメンテーション ツールからバックアノテート</li> </ul>
再利用	可能	不可能

## ロジックの固定

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選択します。
2. 右クリックします。
3. [Fix Instances] をクリックします。

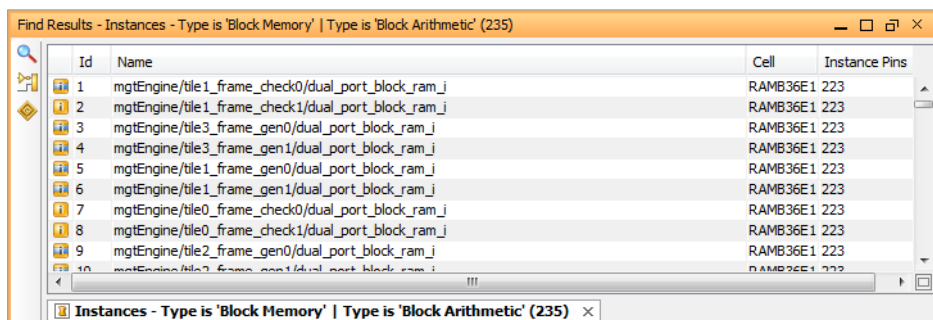


図 2-4 : [Find Results] ビューでロジックを選択

## 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、ツールでの配置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 コンポーネントの配置を調整 (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「デザインのフロアプラン」
- 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「インプリメンテーション結果の解析」

## 階層フロアプラン フロー (デザインでタイミングが満たされない場合)

階層フロアプラン フローは、「**再利用フロー (デザインでときどきタイミングが満たされる場合)**」よりも効果的です。このフローでは、次が可能です。

- ・ 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。
- ・ タイミングが大きく向上します。

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。

### 階層フロアプラン フローの利点と欠点

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る

### タイミングが満たされていないデザインでの階層フロアプラン フローの使用

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。

階層フロアプランでは、次が可能です。

- ・ 階層の小さいレベルを取り出す
- ・ 階層レベルをチップの特定の領域に制約
- ・ 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り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 コンポーネント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。



# タイミング クロージャでのフロアプランの使用

---

この章では、タイミング クロージャを達成するためのフロアプランの使用について説明します。

## フロアプランを使用する際の質問

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。

1. どのようなタイミング エラーが発生しているのか。
2. クリティカルな階層はどれか。
3. 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。
4. フロアプランすべきものがほかにあるか。
5. クリティカルな階層をフロアプランできるか。
6. 何をどこに配置すればよいか。

これらの情報を入手するには、次を確認します。

- タイミング パス、配置、パス上のロジックの構造
- ピン配置とデザイン

次に示す例を参照してください。

## 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。インプリメンテーションを実行してタイミングが満たされない場合は、結果を **PlanAhead™** ツールに読み込みます。

配置およびタイミングの結果とゲートをすべて 1 箇所に表示できます。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。

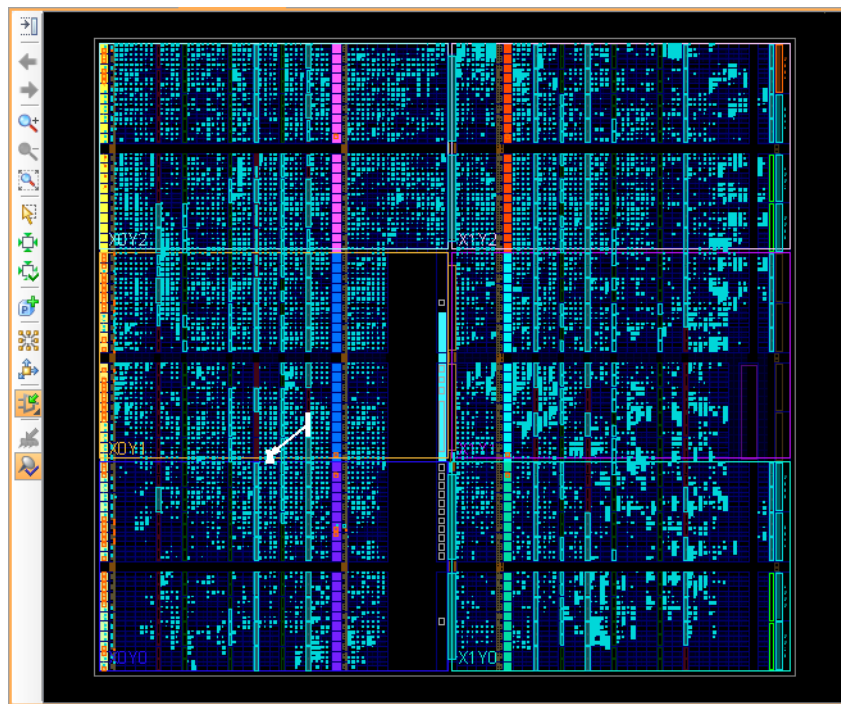


図 3-1：タイミングを満たしていないパスの配置

この図は、クリティカルパスを含むブロック RAM コンポーネントが、チップ上で必要以上に分散していること示しています。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 コンポーネント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

タイミング クロージャを達成するには、ブロック RAM コンポーネント間のパスを解析して、次が必要であるかどうかを判断します。

- 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、28ps の差でタイミングを満たしていません。最初のネットの配線遅延は 937ns 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 (trco_D0B)	(r) 2.073	2.073	RAMB36_X2Y9	usbEngine0/usb_dma_wb_in/buffer_fifo/Mram_fifo_ram
net (fanout=23)	(r) 0.937	3.010		usbEngine0/ma_adr[14]
LUT4 (Tlo)	(r) 0.201	3.211	SLICE_X26Y41	usbEngine0/u5/state_rf_we_d1
net (fanout=17)	(r) 0.341	3.552		usbEngine0/rf_we
LUT5 (Tlo)	(r) 0.198	3.750	SLICE_X26Y41	usbEngine0/u4/adr[6]_we_AND_2411_o11
net (fanout=3)	(r) 0.406	4.156		usbEngine0/u4/N11
LUT4 (Tlo)	(r) 0.191	4.347	SLICE_X26Y41	usbEngine0/u4/adr[6]_we_AND_2415_o1
net (fanout=6)	(r) 0.388	4.735		usbEngine0/u4/adr[6]_we_AND_2415_o
EDRE (trcedk)	(r) 0.318	5.053	SLICE_X25Y40	usbEngine0/u4/intb_msk_6
<b>Total</b>		5.053		

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。パフォーマンスをどれだけ向上できるかは、ロジック遅延により制限されます。

デザインのロジック遅延の割合が高い場合は、次のようにしてゲートを変更します。

- ・ コードを変更する。
- ・ 合成をアップデートする。

## 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。

ただし、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、時間がかかり、困難である。
- ・ 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。

その代わりに、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、usbEngine0 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。

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。

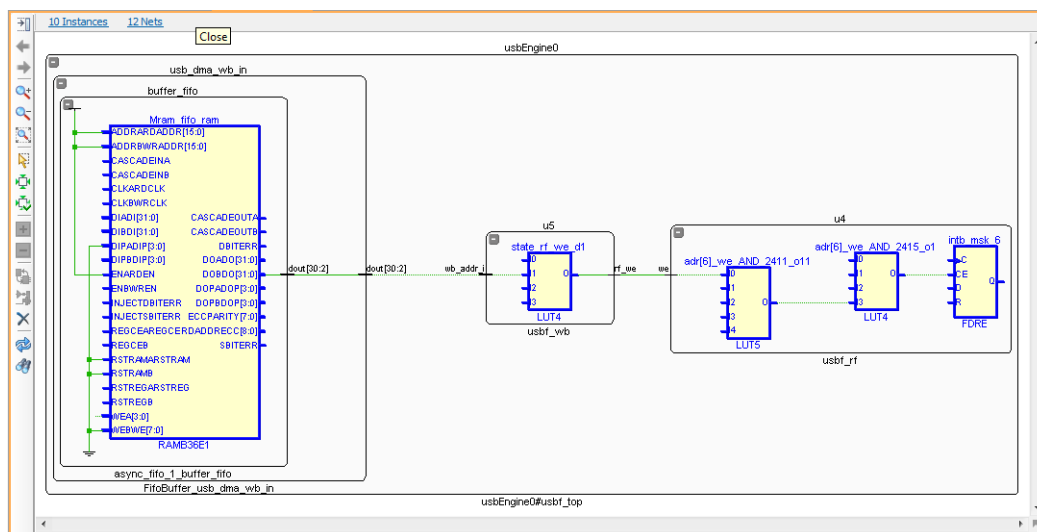


図 3-3：クリティカルパスのゲートと階層

フロアプランでは、少なくとも各 **usbEngine** 内のブロック **RAM** コンポーネント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。両方の **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 「usbEngine のクリティカルパスとクリティカルでないパス」では、次のように示されています。

- 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緑色で示されています。

図 3-4 から、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** コンポーネントの使用率が高い。
- クリティカルでない階層には、ブロック **RAM** コンポーネント間に配置可能な 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。

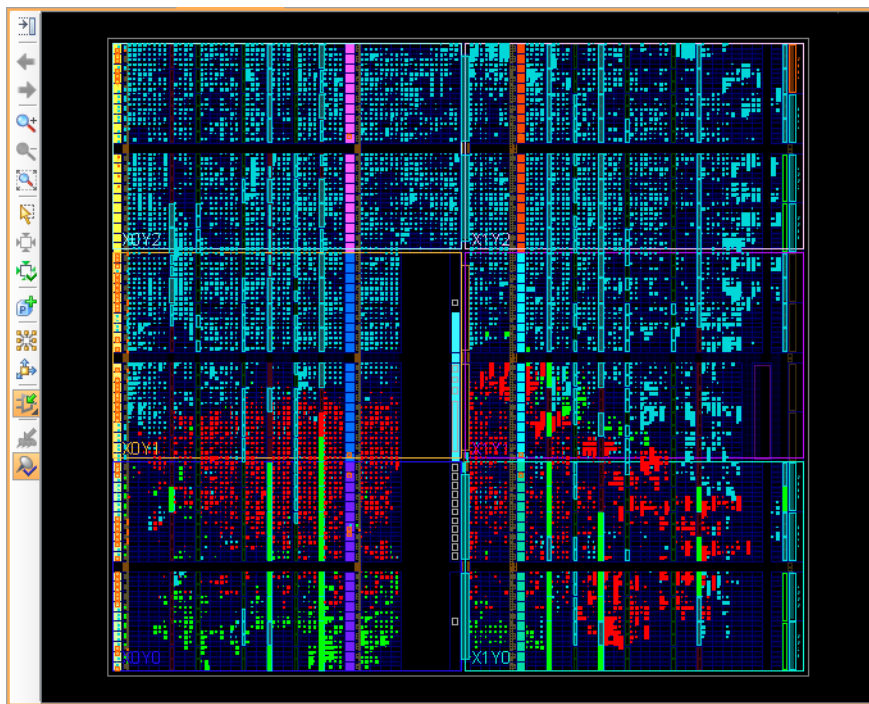


図 3-4 : usbEngine のクリティカル パスとクリティカルでないパス

**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確認します。**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します。
- デバイス上に最上位フロアプランを作成します (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」では、次のように示されています。

- I/O 接続は、緑色の線で示されています。  
チップの左側中央の I/O バンクから、デバイスの右下にあるロジックに描か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。白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。
  - ほかのほとんどのブロックと通信します。
  - チップ上のさまざまな部分に接続されてい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フロアプランを示します。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 (右下の四角) 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。

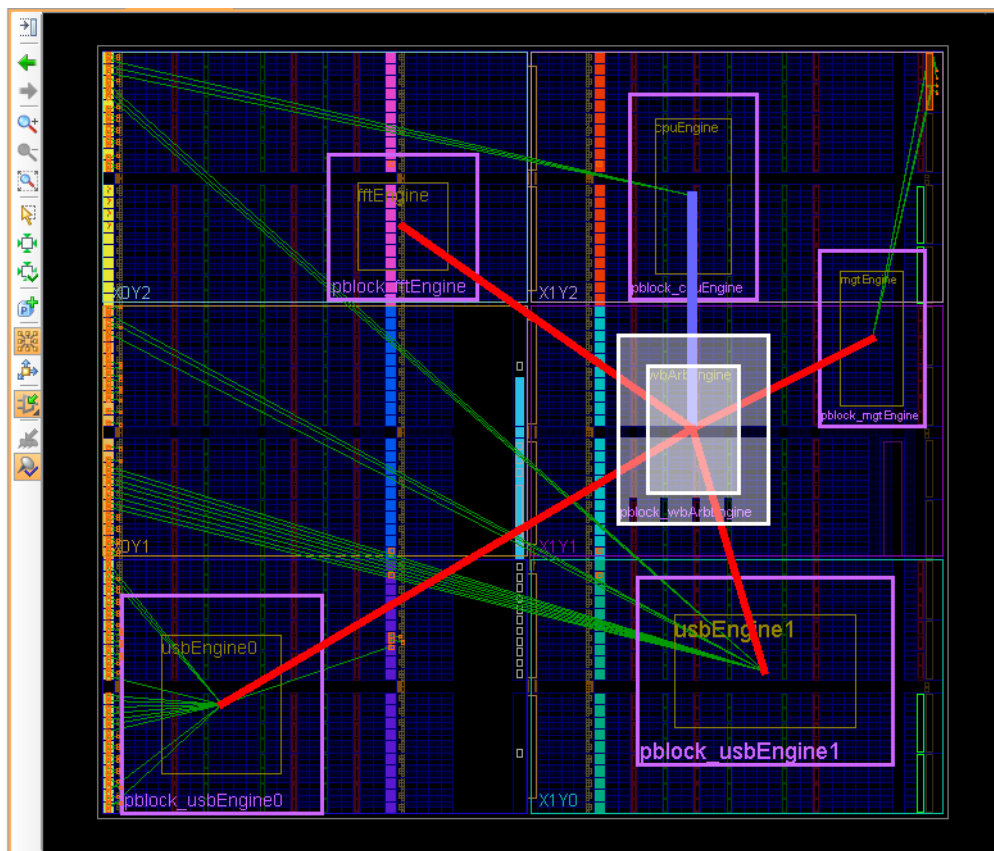


図 3-5：解析用の最上位フロアプラン

## 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 コンポーネントを使用するようフロアプランするのが適切です。

次を使用するようブロックのサイズを調整します。

- ブロック RAM (または DSP) の 100%
- スライスの約 80%

## 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。

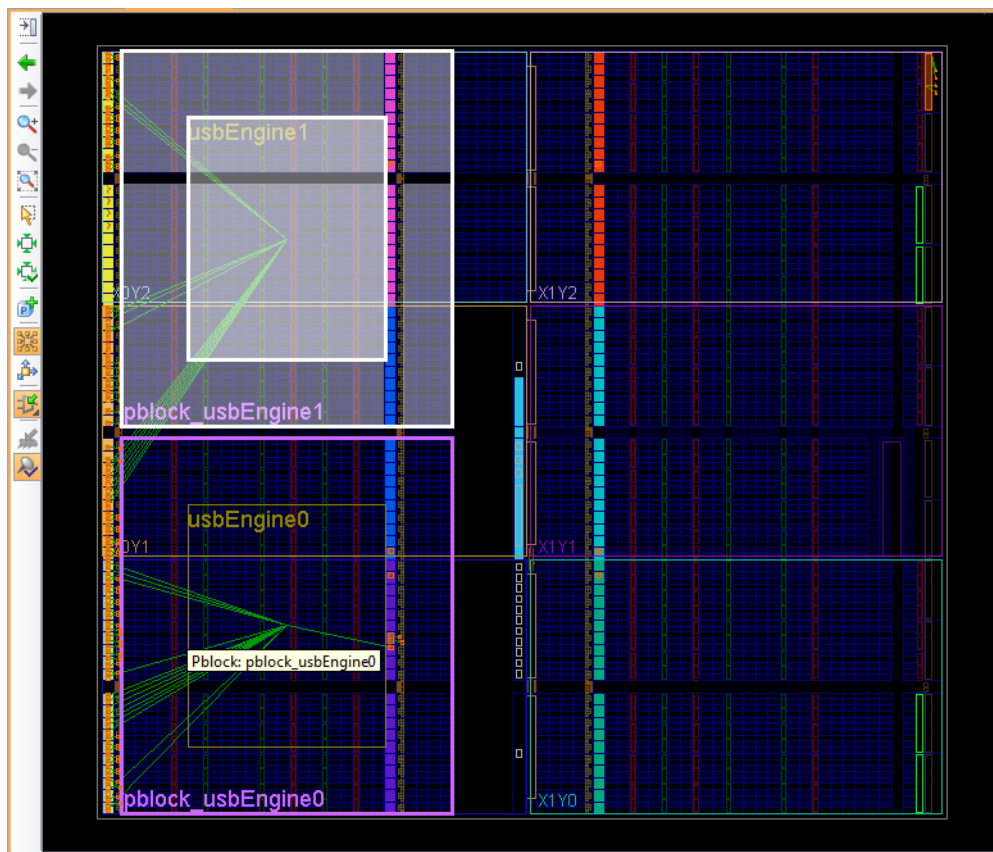


図 3-6 : 最初のフロアプラン

## ネットリスト階層のサブセットの制約

PlanAhead ツールでは、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作成されます。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;
```

これら各行で、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 コンポーネント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。

## フロアプランの反復実行

---

フロアプランは反復的に実行し、次のセクションで示すオプションを考慮することをお勧めします。すべての状況ですべてのオプションが有益であるとは限りません。デザインで最適なソリューションを見つけるため、複数のオプションを試してみてください。

### 一般的な推奨事項

次に、フロアプランの反復実行における一般的な推奨事項を示します。

#### いろいろ試行する

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。

#### 隠れた接続がないかを探す

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、隠れた接続がないか探します。デザインに、簡単には明らかにならない接続がある可能性があります。

#### フロアプランを修正する

必要に応じてフロアプランを修正します。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くと便利です。

#### フロアプランをシンプルにする

フロアプランはなるべくシンプルにします。通常、複雑なフロアプランより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。

#### アップグレード後にデザインを再実行する

ISE® Design Suite のリリースをアップグレードした場合は、制約を適用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。

## クリティカル パスの修正

次のセクションに、クリティカル パスのロケーションに基づいてフロアプランを向上するための推奨事項を示します。

### クリティカル パスがフロアプランされていないロジックに含まれる

クリティカル パスがフロアプランされてい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します。
- その階層レベルを新規 Pblock に割り当てます。
- その Pblock をチップ上に配置します。
- 配置が適当である場合は、この Pblock を配置配線に使用します。

### クリティカル パスが 1 つの Pblock に含まれる

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します。
- 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

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 パスを含む新しい 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

クリティカル パスが 2 つの Pblock の間にある場合は、次のようにします。

- 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。
- ロジックを一方の Pblock からもう一方の Pblock に移動します。



## クリティカルな階層でのタイミングの向上

1. 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、次のようにしてください。
  - 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ます。
  - 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 を修正して再合成してみます。
2. フロアプランした部分のタイミングが満たされない場合は、フロアプラン制約を削除してみます。
3. 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。



## その他のリソース

---

### ザイリンクス リソース

- デバイス ユーザー ガイド :  
[http://japan.xilinx.com/support/documentation/user\\_guides.htm](http://japan.xilinx.com/support/documentation/user_guides.htm)
- ザイリンクス用語集 : <http://japan.xilinx.com/company/terms.htm>
- 『ザイリンクス デザイン ツール : 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/xilinx14\\_1/iil.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/iil.pdf)
- 『ザイリンクス デザイン ツール : リリース ノート ガイド』(UG631) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_1/irn.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/irn.pdf)
- 製品サポートと資料 : <http://japan.xilinx.com/support>

### ISE 資料

- ライブラリ ガイド :  
[http://japan.xilinx.com/support/documentation/dt\\_ise14-1\\_librariesguides.htm](http://japan.xilinx.com/support/documentation/dt_ise14-1_librariesguides.htm)
- ISE Design Suite 資料 :  
[http://japan.xilinx.com/support/documentation/dt\\_ise14-1.htm](http://japan.xilinx.com/support/documentation/dt_ise14-1.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/xilinx14\\_1/devref.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/devref.pdf)
  - 『制約ガイド』(UG625) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_1/cgd.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/cgd.pdf)
  - 『Data2MEM ユーザー ガイド』(UG658) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_1/data2mem.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/data2mem.pdf)
  - 『ISim ユーザー ガイド』(UG660) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_1/plugin\\_ism.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/plugin_ism.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/xilinx14\\_1/sim.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/sim.pdf)
  - 『タイミング クロージャ ユーザー ガイド』(UG612) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_1/ug612.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ug612.pdf)
  - 『ザイリンクス/Cadence PCB ガイド』(UG629) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_1/cadence\\_pcb.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/cadence_pcb.pdf)
  - 『ザイリンクス/Mentor Graphics PCB ガイド』(UG630) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_1/mentor\\_pcb.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/mentor_pcb.pdf)

- 『XPower Estimator ユーザー ガイド』(UG440) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_1/ug440.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ug440.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/xilinx14\\_1/xst.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/xst.pdf)
- 『XST ユーザー ガイド (Virtex-6、Spartan-6、7 シリーズ用)』(UG627) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_1/xst\\_v6s6.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/xst_v6s6.pdf)

## PlanAhead 資料

- PlanAhead ユーザー ガイド :  
[http://japan.xilinx.com/support/documentation/dt\\_planahead\\_planahead/14-1\\_userguides.htm](http://japan.xilinx.com/support/documentation/dt_planahead_planahead/14-1_userguides.htm)
- 『フロアプラン設計手法ガイド』(UG633) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_1/Floorplanning\\_Methodolgy\\_Guide.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/Floorplanning_Methodolgy_Guide.pdf)
- 『階層デザイン設計手法ガイド』(UG748) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_1/Hierarchical\\_Design\\_Methodolgy\\_Guide.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/Hierarchical_Design_Methodolgy_Guide.pdf)
- 『ピン配置設計手法ガイド』(UG792) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_1/ug792\\_pinplan.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ug792_pinplan.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/xilinx14\\_1/ug789\\_pa\\_tcl\\_commands.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ug789_pa_tcl_commands.pdf)
- 『PlanAhead ユーザー ガイド』(UG632) :  
[http://japan.xilinx.com/support/documentation/sw\\_manuals/xilinx14\\_1/PlanAhead\\_UserGuide.pdf](http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/PlanAhead_UserGuide.pdf)
- PlanAhead チュートリアル :  
[http://japan.xilinx.com/support/documentation/dt\\_planahead\\_planahead14-1\\_tutorials.htm](http://japan.xilinx.com/support/documentation/dt_planahead_planahead14-1_tutorials.htm)