

フロアプラン手法ガイド

UG633 (v12.1) 2010 年 5 月 3 日





Xilinx is disclosing this Document and Intellectual Property (hereinafter “the Design”) to you for use in the development of designs to operate on, or interface with Xilinx FPGAs. Except as stated herein, none of the Design may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted 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. Any unauthorized use of the Design may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes.

Xilinx does not assume any liability arising out of the application or use of the Design; nor does Xilinx convey any license under its patents, copyrights, or any rights of others. You are responsible for obtaining any rights you may require for your use or implementation of the Design. Xilinx reserves the right to make changes, at any time, to the Design as deemed desirable in the sole discretion of Xilinx. Xilinx assumes no obligation to correct any errors contained herein or to advise you of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or technical support or assistance provided to you in connection with the Design.

THE DESIGN IS PROVIDED “AS IS” WITH ALL FAULTS, AND THE ENTIRE RISK AS TO ITS FUNCTION AND IMPLEMENTATION IS WITH YOU. YOU ACKNOWLEDGE AND AGREE THAT YOU HAVE NOT RELIED ON ANY ORAL OR WRITTEN INFORMATION OR ADVICE, WHETHER GIVEN BY XILINX, OR ITS AGENTS OR EMPLOYEES. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DESIGN, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOST DATA AND LOST PROFITS, ARISING FROM OR RELATING TO YOUR USE OF THE DESIGN, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE TOTAL CUMULATIVE LIABILITY OF XILINX IN CONNECTION WITH YOUR USE OF THE DESIGN, WHETHER IN CONTRACT OR TORT OR OTHERWISE, WILL IN NO EVENT EXCEED THE AMOUNT OF FEES PAID BY YOU TO XILINX HEREUNDER FOR USE OF THE DESIGN. YOU ACKNOWLEDGE THAT THE FEES, IF ANY, REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT XILINX WOULD NOT MAKE AVAILABLE THE DESIGN TO YOU WITHOUT THESE LIMITATIONS OF LIABILITY.

The Design is not designed or intended for use in the development of on-line control equipment in hazardous environments requiring fail-safe controls, such as in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support, or weapons systems (“High-Risk Applications”) Xilinx specifically disclaims any express or implied warranties of fitness for such High-Risk Applications. You represent that use of the Design in such High-Risk Applications is fully at your risk.

© 2010 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

Demo Design License

© 2010 Xilinx, Inc.

This Design is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Library General Public License along with this design file; if not, see: <http://www.gnu.org/licenses/>

The PlanAhead™ software source code includes the source code for the following programs:

Centerpoint XML

The initial developer of the original code is CenterPoint – Connective Software

Software Engineering GmbH. portions created by CenterPoint – Connective Software

Software Engineering GmbH. are Copyright© 1998-2000 CenterPoint - Connective Software Engineering GmbH. All Rights Reserved. Source code for CenterPoint is available at <http://www.cpointc.com/XML/>

NLView Schematic Engine

Copyright© Concept Engineering.

Static Timing Engine by Parallax Software Inc.

Copyright© Parallax Software Inc.

Java Two Standard Edition

Includes portions of software from RSA Security, Inc. and some portions licensed from IBM are available at <http://oss.software.ibm.com/icu4j/>

Powered By JIDE – <http://www.jidesoft.com>

The BSD License for the JGoodies Looks

Copyright© 2001-2010 JGoodies Karsten Lentzsch. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of JGoodies Karsten Lentzsch nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Free IP Core License

This is the Entire License for all of our Free IP Cores.

Copyright (C) 2000-2003, ASICS World Services, LTD. AUTHORS

All rights reserved.

Redistribution and use in source, netlist, binary and silicon forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of ASICS World Services, the Authors and/or the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

本資料は英語版 (v.12.1) 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。

目次

このガイドについて.....	7
その他のリソース	7
フロアプランの概要	9
はじめに	9
タイミング クロージャの基礎.....	9
フロアプランの基礎.....	11
注意事項	14
クリティカル タイミングを含むロジックのフロアプラン	14
階層ネットリストの操作	14
ロジック合成での推奨事項.....	14
一貫性の向上.....	15
クロック リソースに焦点を置いたフロアプラン	15
フロアプラン フロー.....	16
推奨されるフロー.....	16
再利用フロー	16
階層フロアプラン フロー	17
再利用フロー.....	17
階層フロアプラン フロー	22
タイミング クロージャでのフロアプランの使用例.....	22
配置配線結果	22
タイミング結果	23
ゲートおよび階層	24
クリティカルな階層のフロアプランの成形.....	27
ほかにフロアプランが必要かどうかの判断	27
反復フロアプラン	29
まとめ	29

このガイドについて

FPGA デバイスは著しく成長しており、より大型の FPGA により複雑なデザインをインプリメントできるようになってきています。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 ツールも向上してきました。デザインによっては、インプリメンテーション ツールの設定によって、システム クロック周波数を向上し、インプリメンテーション ランタイムを短縮し、より一貫したタイミングを得ることが可能です。

このガイドでは、フロアプランの基礎を説明し、デザインのタイミングをより確実に満たすために役立つフロアプランの 2 つの方法を示します。このガイドは、ザイリンクス ソフトウェアでのフロアプランに関する考慮事項および手法に焦点を置いています。

次の内容が含まれています。

- 第 1 章「[フロアプランの概要](#)」: フロアプランおよびタイミング クロージャの基礎、フロアプランにおけるデザインでの考慮事項および手法を説明します。
- 第 2 章「[フロアプラン フロー](#)」: フロアプランに推奨される 2 つの方法、配置の再利用および階層フロアプランを示します。

メモ :このガイドを参照する前に、PlanAhead™ ソフトウェアの使用に慣れておくことをお勧めします。PlanAhead ソフトウェアの詳細は、PlanAhead チュートリアルおよび『PlanAhead ユーザー ガイド』(UG632)を参照してください。

その他のリソース

PlanAhead チュートリアル の 1 つを実行し、サンプル デザインを使用して PlanAhead ソフトウェアの機能を学ぶことができます。

http://japan.xilinx.com/support/documentation/dt_planahead_planahead12-1_tutorials.htm

PlanAhead の特定の機能やコマンドの詳細は、『PlanAhead ユーザー ガイド』(UG632)を参照してください。

http://japan.xilinx.com/support/documentation/sw_manuals/xilinx12_1/PlanAhead_UserGuide.pdf

PlanAhead に関する一般的な情報およびビデオ デモ、ホワイト ペーパーについては、次を参照してください。

<http://japan.xilinx.com/planahead>

フロアプランの概要

この章には、次のセクションが含まれています。

- はじめに
- タイミング クロージャの基礎
- フロアプランの基礎
- 注意事項
- ロジック合成での推奨事項
- 一貫性の向上
- クロック リソースに焦点を置いたフロアプラン

はじめに

適切なフロアプラン手法を使用すると、パフォーマンスを向上し、配置配線済みのデザインのタイミングを満たすことができます。フロアプランとは、デザインのロジックの最適なグループ化および接続を見つけるためのプロセスで、集積度、配線、パフォーマンスを向上することを目的として FPGA にロジック ブロックを手動で配置します。より良い配置を指定することにより、配置遅延を削減することが目的です。

デザインでタイミングが満たされなかったり、満たされたり満たされなかったりする場合、フロアプランが有益である可能性があります。フロアプランは、クリティカル パス上の個々のロジック エLEMENTをチップの特定のサイトに配置する詳細な方法から、階層レベルをチップ上の特定の領域に制約するより大まかな方法まであります。

配置配線を実行する前に多少の時間をかけてデザインをフロアプランする場合や、問題が特定されてからフロアプランを実行する場合があります。

このガイドでは、さまざまなフロアプラン ストラテジの利点と欠点を説明します。

タイミング クロージャの基礎

フロアプランは、多くの場合、デザインでセットアップ タイミング制約が満たされない場合に実行します。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、インプリメンテーション ツールにより、ロジック遅延と配線遅延がタイミング制約で許容される遅延値と比較されます (クロック間のスキューおよびクロック ノイズによる多少の変動を考慮)。

レポートにタイミング制約が満たされたか、満たされていないかが示されます。[図 1](#) に、タイミングレポートの例を示します。

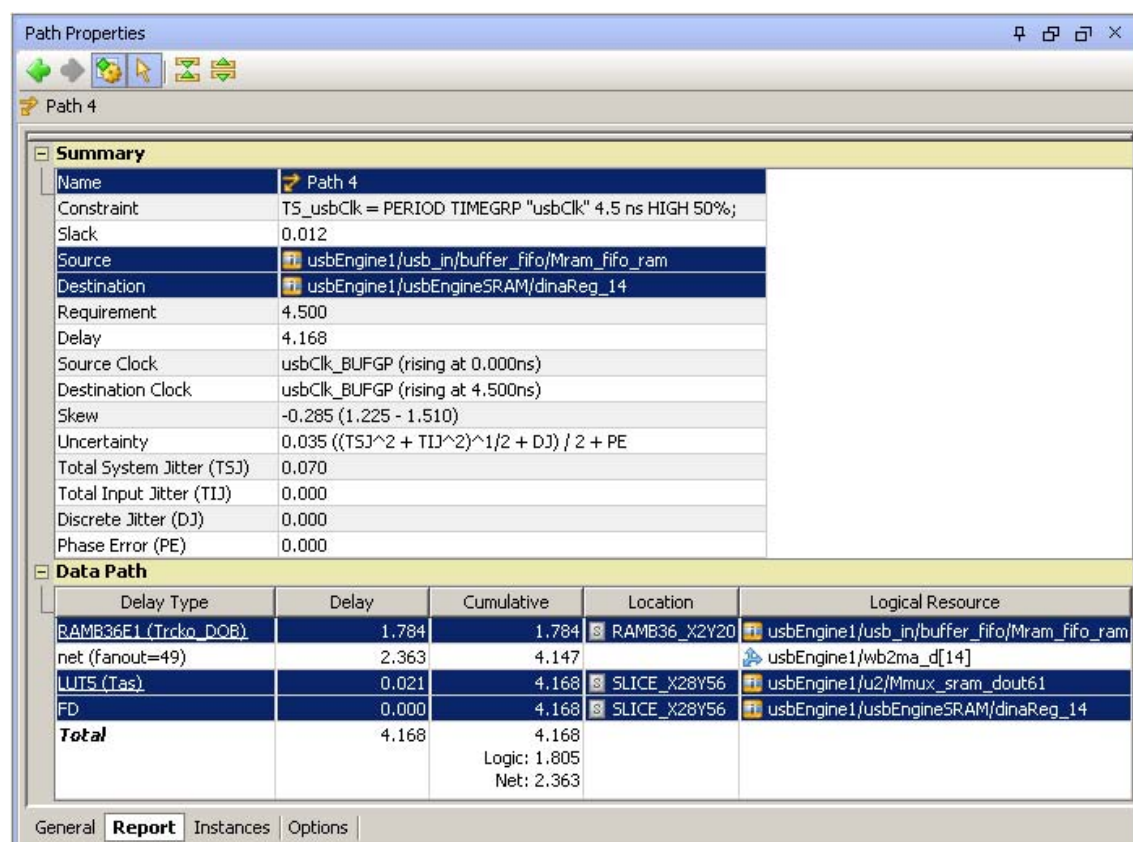


図 1 : タイミング レポートの例

まず、タイミング制約が適切であることを確認するのが重要です。パスが複数サイクル パスであるか、フォールス パスであるかを判断する必要があります。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 パスまたはフォールス 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 パスおよびフォールス パスの詳細は、『Timing Constraints User Guide』(UG612) を参照してください。

http://japan.xilinx.com/support/documentation/sw_manuals/xilinx12_1/ug612.pdf

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。

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。RTL を変更するか、合成エンジンの設定を変更してください。

パス遅延の大部分が配置遅延であり、ロジックがデバイス全体に分散されている場合は、配置に問題がある場合があります。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すると、配線遅延を削減したり、RTL をどのように変更したらよいかを判断できます。

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。タイミング問題の原因となっているロジックを特定し、ロジックが近くに配置されるよう指定します。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が最終的な目標です。

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化しません。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です。フロアプラン中に、再合成することが有益な問題が見つかる場合もあります。共通する推奨事項として、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。

フロアプランは多くのデザインで有益ですが、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、配線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、『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。

http://japan.xilinx.com/support/documentation/sw_manuals/xilinx12_1/Hierarchical_Design_Methodology_Guide.pdf

フロアプランには、大まかなフロアプランから詳細なフロアプランまであります。非常に厳しいタイミング クリティカル パスの場合は、図 2 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が、これは最終手段としてください。すべてを手動配置するには時間がかかり、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も必要です。また、ゲート レベルの配置は安定しておらず、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もあります。

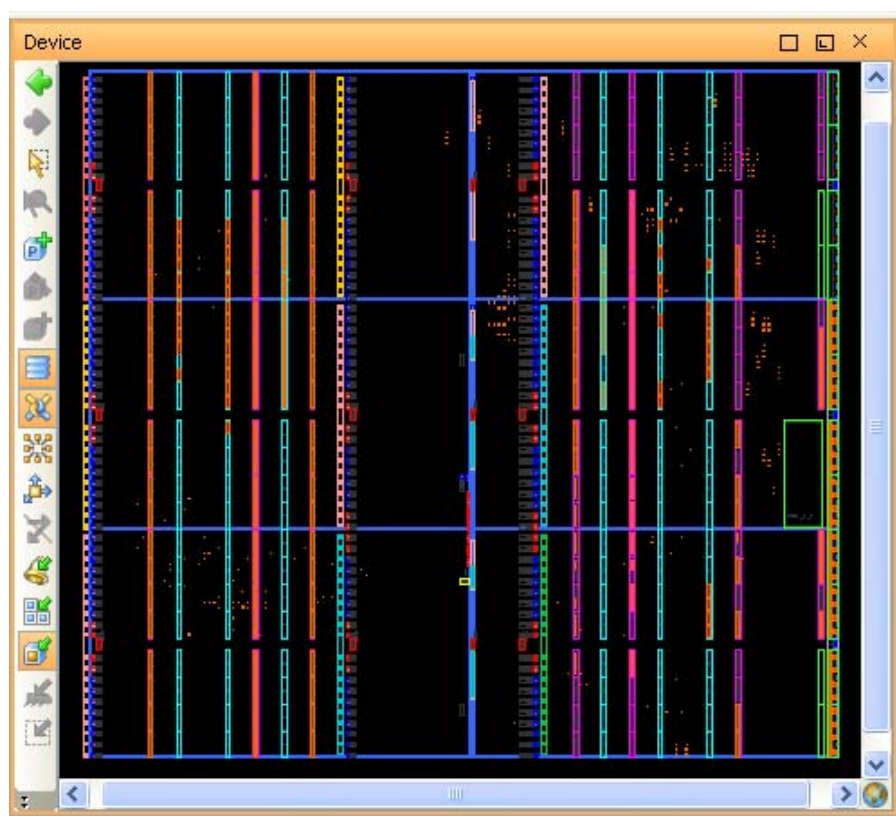


図 2 : 手動でフロアプランされたロジック

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ます。階層フロアプランでは、図 3 に示すように、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し、配置ツールに指示できます。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。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。

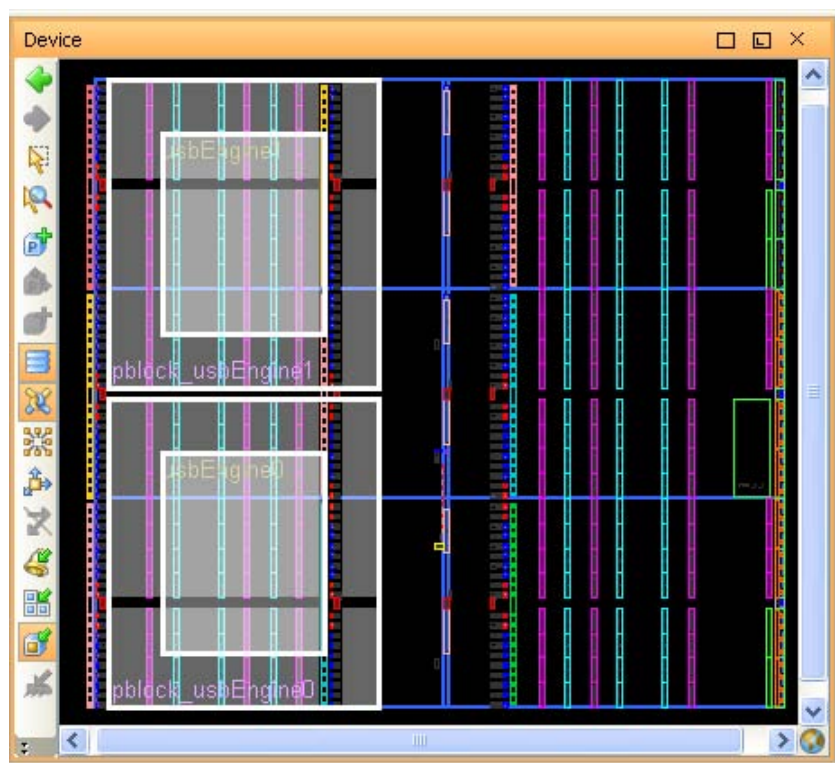


図 3 : フロアプランされた階層

RTL を構築中またはピン配置を実行中に、高レベルのフロアプランを生成する必要がある場合もあります。高レベル フロアプランにより、デバイス全体のデータ フローを可視化することが可能です。高レベル フロアプランは、合成済みネットリストを作成前に階層ブロックに対して実行する必要があります。このフロアプランは、より良い RTL およびピン配置を生成するために使用できます。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。
- 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。

注意事項

フロアプランは、通常反復作業です。最初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。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。

クリティカル タイミングを含むロジックのフロアプラン

フロアプランを初めて実行する場合は、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するのが良い開始点です。一般的には、PAR (Place and Route) 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。ほとんどの FPGA デザインでは、PAR に入力する合成済みネットリストの形であればデザイン全体のフロアプランがサポートされますが、デザイン全体のフロアプランはお勧めしません。

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す。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 2 つを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。手動でレジスタを複製し、ファンアウトの低いレジスタを 2 つ作成すると、その 1 つでチップの 1 辺のロードを駆動し、もう 1 つで反対側の辺のロードを駆動することができます。

ロジック合成での推奨事項

ロジック合成手法での推奨事項は、次のとおりです。

- 可能な限り、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制限します。
- 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てみてください。

- 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。インクリメンタル手法では、個々のブロックを別々に合成したり、合成属性 (SYN_HIER=HARD) を使用して階層を保持したりできます。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、フロアプランおよび配置制約が困難になります。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** を使用します。PlanAhead™ ツールで合成を実行する場合は、このオプションを含むストラテジがデフォルトで使用されます。

一貫性の向上

フロアプランを使用すると、デザインでより一貫した結果を得られるようになり、QoR (結果の質) が向上します。フロアプランにより、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ようにすることが可能です。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。ただし、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、『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。

http://japan.xilinx.com/support/documentation/sw_manuals/xilinx12_1/Hierarchical_Design_Methodology_Guide.pdf

クロック リソースに焦点を置いたフロアプラン

デバイス上のクロック 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 ファミリー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、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。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

この章には、次のセクションが含まれています。

- 推奨されるフロー
- 再利用フロー
- 階層フロアプラン フロー
- タイミング クロージャでのフロアプランの使用例
- 反復フロアプラン
- まとめ

推奨されるフロー

このマニュアルでは、一般的によく使用される次の 2 つのフロアプラン フローについて説明します。

- 再利用フロー
- 階層フロアプラン フロー

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有益です。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ときどきでも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。

階層フロアプラン フロー

階層フロアプラン フローは、再利用フローよりも効果的です。適切な階層フロアプランにより、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る可能性があります。適切なフロアプランを作成するためにデザインを解析することにより、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。

このフローはエンジニアリング時間が必要で、反復作業が必要となる場合があります。

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。

再利用フロー

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/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。インプリメンテーション ツールを使用してタイミングが満たされる配置を見つけ、その一部を再利用するということです。このフローは、次のような場合に使用できます。

- デザインで時々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。

配置が完了し、タイミングが満たされた実行から開始してください。次の図に示す [Project Summary] ビューで、[Timing Score] が 0、[Unrouted] が 0 の実行を見つけます。タイミングが満たされている実行が複数ある場合は、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。スクリプト、Project Navigator、または PlanAhead™ ツールのインプリメンテーション実行を使用できます。PlanAhead にタイミングが満たされているデザインを読み込み、配置を制約します。

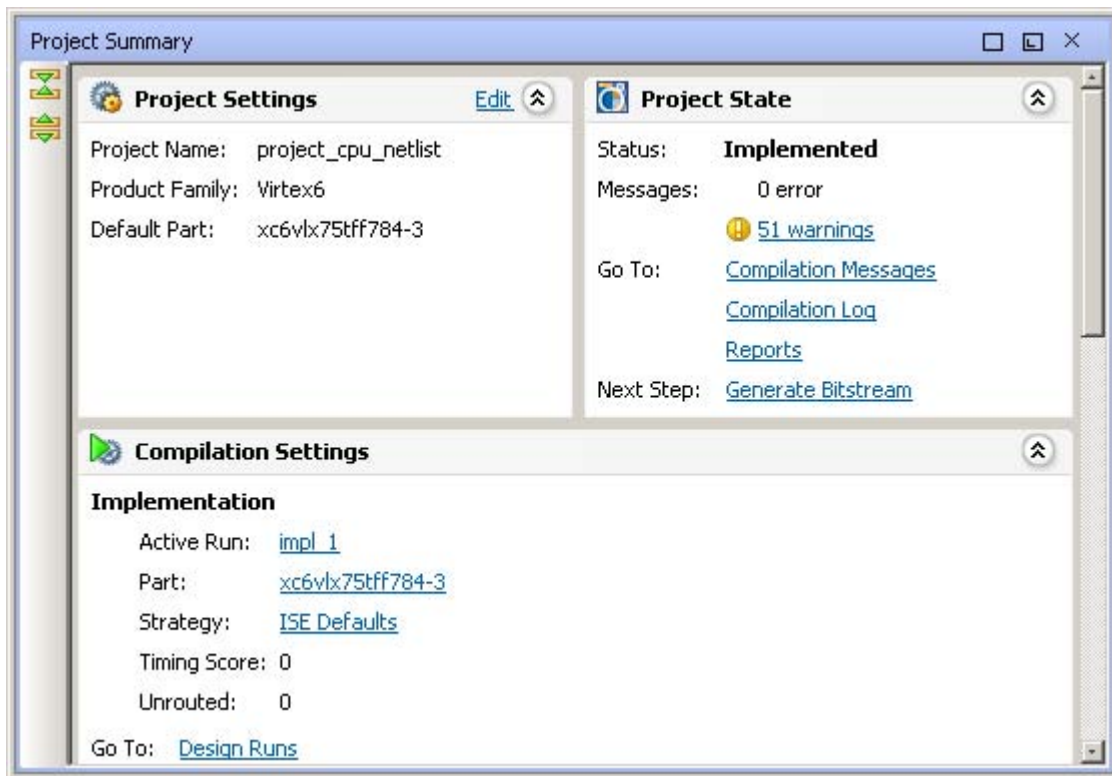


図 4 : [Project Summary] ビュー

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] プロセスを実行します。
 - PlanAhead の Flow Navigator で [Implement] 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きます。
- 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、新規 PlanAhead プロジェクトを作成し、New Project ウィザードで [Import ISE Place & Route results] をオンにします。

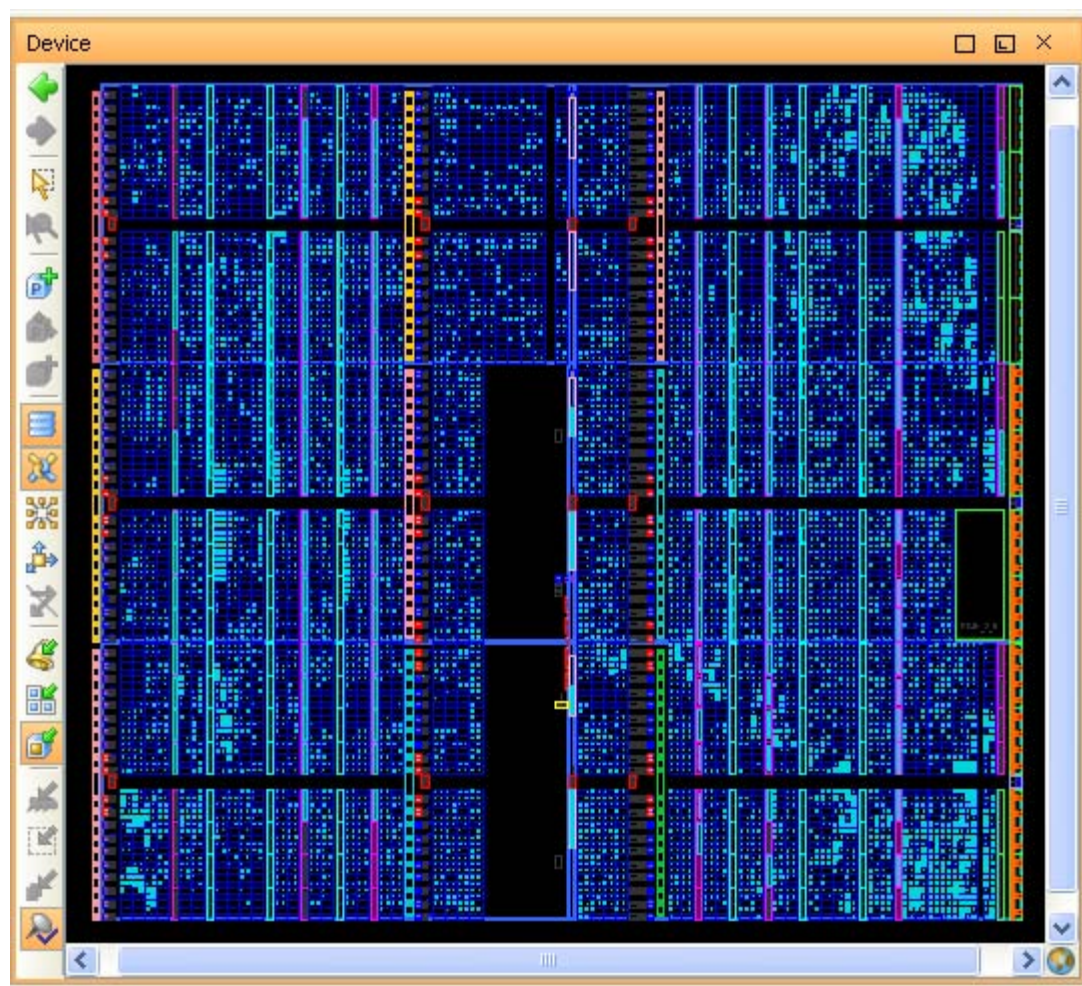


図 5 : インプリメンテーション配置の表示



図 6 : インプリメンテーション配置の表示

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も可能です。デザインが変更される可能性があるため、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、ほかのゲートの変更した場合でもタイミングを保持できます。PlanAhead ツールでは、すべてのブロック メモリ (RAMB および FIFO プリミティブ) およびブロック演算 (アーキテクチャにより MULT および DSP プリミティブ) を簡単に検索できます。[Highlight] または [Mark] コマンドを使用すると、配置が見やすくなります。

[Edit] → [Find] をクリックし、[Find] ダイアログ ボックスを使用してこれらのプリミティブを検索してください。

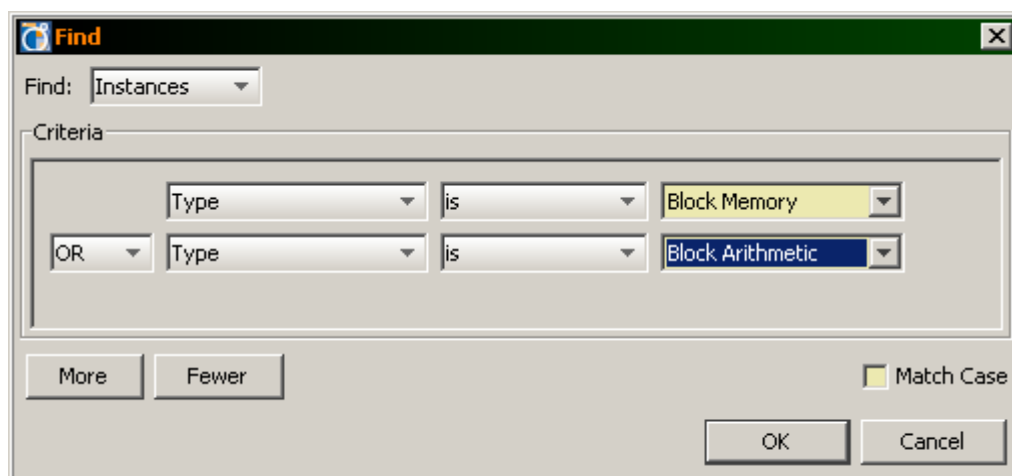


図 7 : メモリ ブロックおよび演算ブロックの検索

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。

PlanAhead ソフトウェアには、2 種類の配置があります。

- 固定：UCF の配置、ユーザーが手動で作成した配置、または PlanAhead でユーザーが指定した配置。このタイプの配置は再利用できます。
- 非固定：インプリメンテーション ツールからバックアノテートされた配置。このタイプの配置は再利用しないでください。

ロジックを固定するには、次の手順に従います。

1. 図 8 に示すように、[Find Results] パネルから固定する配置をすべて選択します。
2. 右クリックして [Fix Instances] をクリックします。

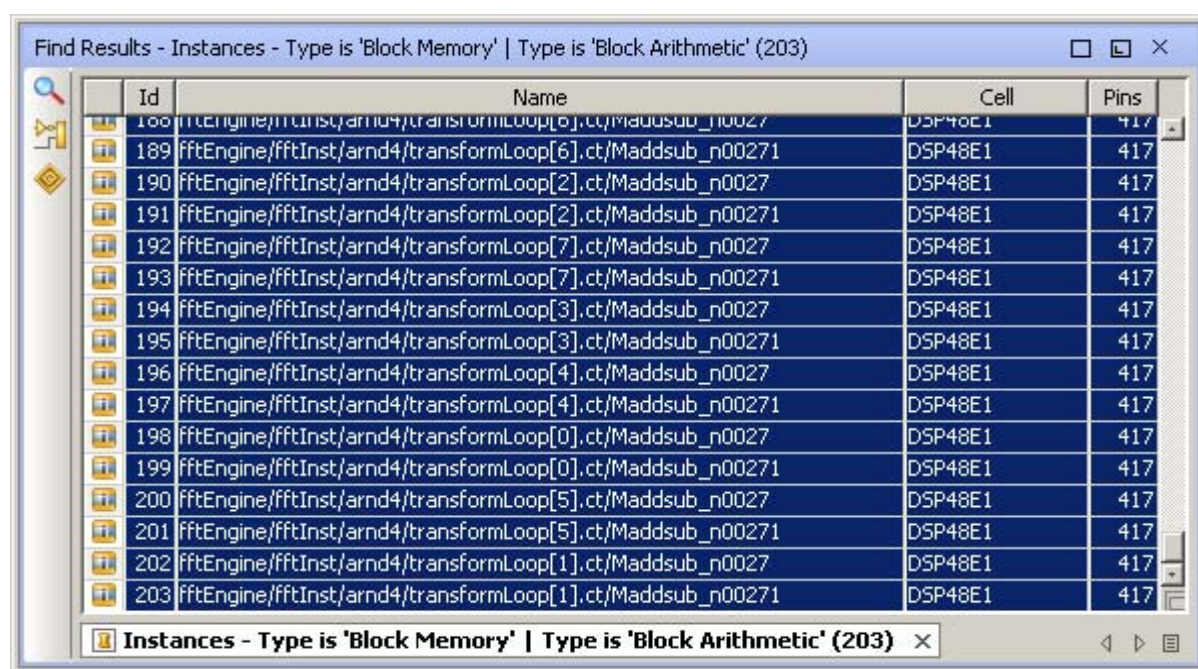


図 8 : [Find Results] パネルでロジックを選択

[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。プロジェクトを保存して閉じます。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。また、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、マクロの LOC 制約なしで PAR を実行できます。各ブロック RAM または DSP48 の配置を調整することも可能です。配置の解析方法および変更方法の詳細は、次のサイトから『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」および第 11 章「デザインのフロアプラン」を参照してください。

http://japan.xilinx.com/support/documentation/sw_manuals/xilinx12_1/PlanAhead_UserGuide.pdf

階層フロアプラン フロー

デザインのタイミングが満たされない場合、より詳細な方法が必要となります。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。階層の小さいレベルを取り出してチップのある領域に制約し、インプリメンテーションのガイドとして使用します。インプリメンテーションでは、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、通常配置の微調整はうまく機能しますが、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場合、フロアプラン プロセスでそれが明らかになります。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。

タイミング クロージャでのフロアプランの使用例

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、次に示す例に説明するように、タイミング パス、パス上のロジックの配置と構造、ピン配置およびデザインに関する情報から判断できます。

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 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 つの場所に表示できます。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 (図 9 を参照)。

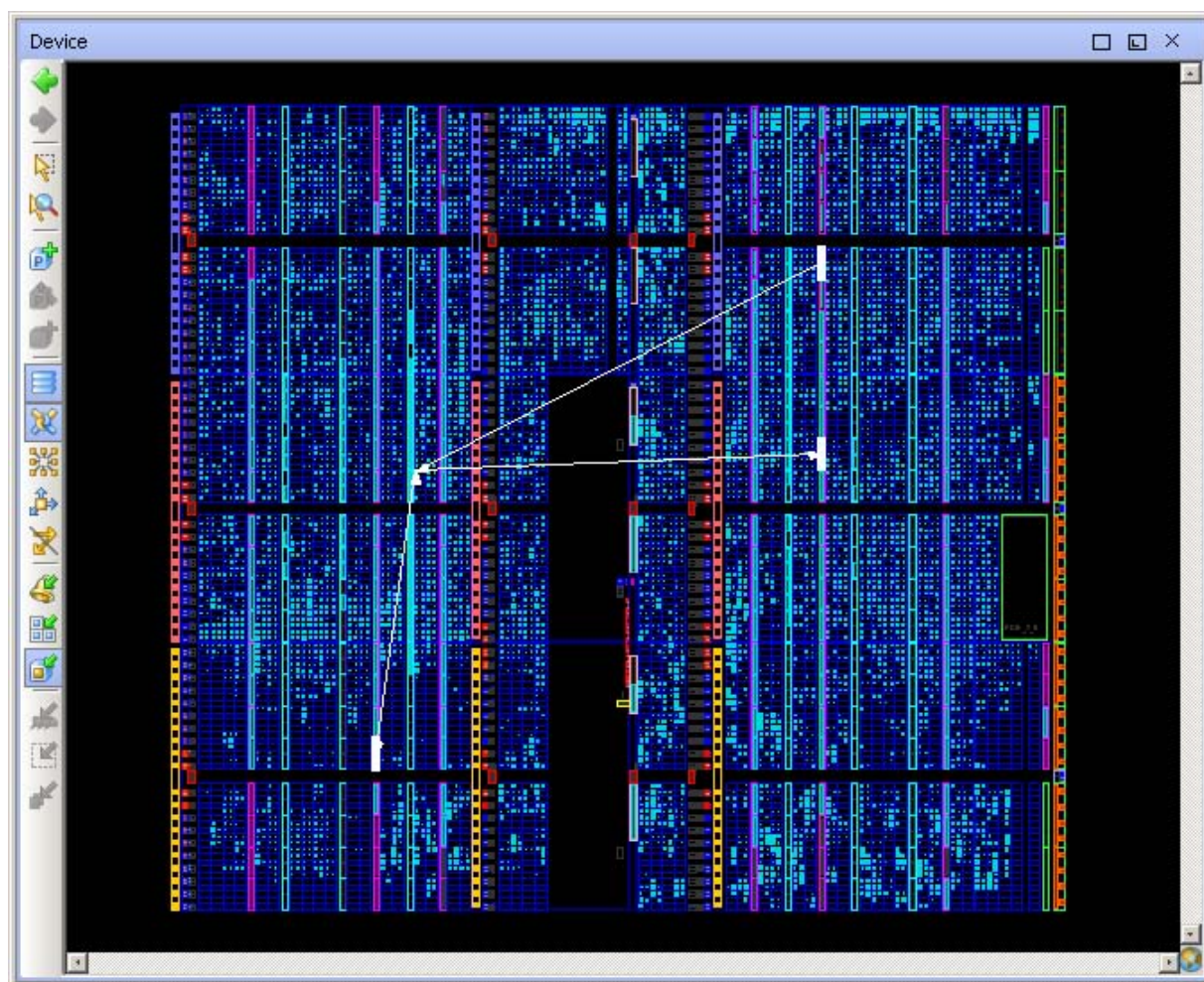


図 9: タイミングを満たしていないパスの配置

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、タイミング クロージャを達成するためにフロアプラン、ロジックの変更、またはその両方が必要であるかどうかを判断するのに役立ちます。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。詳細を [図 10](#) に示します。パスは、1ns 以上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 2.25ns で、3 番目のネットの配線遅延は 1.5ns です。ファンアウトは 40 と 256 ですが、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 (Trcke_DQB)	1.591	1.591	S RAMB36_X4Y19	usbEngine1/usb_dma_wb_in/buffer_fifo/Mram_fifo_ram
net (fanout=40)	2.245	3.836		usbEngine1/ma_adr[14]
LUT6 (Tila)	0.053	3.889	S SLICE_X28Y64	usbEngine1/u5/ma_we1
net (fanout=1)	0.283	4.172		usbEngine1/ma_we
LUT4 (Tila)	0.053	4.225	S SLICE_X28Y64	usbEngine1/u2/Mmux_sram_we11
net (fanout=256)	1.548	5.773		usbEngine1/sram_we_o
RAMB36E1 (Trcke_WEA)	0.437	6.210	S RAMB36_X4Y13	usbEngine1/usbEngineSRAM/Mram_snoopyRam1
Total	6.210	6.210		
		Logic: 2.134		
		Net: 4.076		

図 10 : 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、達成できる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できます。

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、ゲートを特定し、配置するのは時間がかかる困難な作業であるため、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。また、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ります。

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 10 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 11 に示すように、回路図にクリティカル パスが含まれているゲートと、そのゲートが配置されている階層が示されます。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。

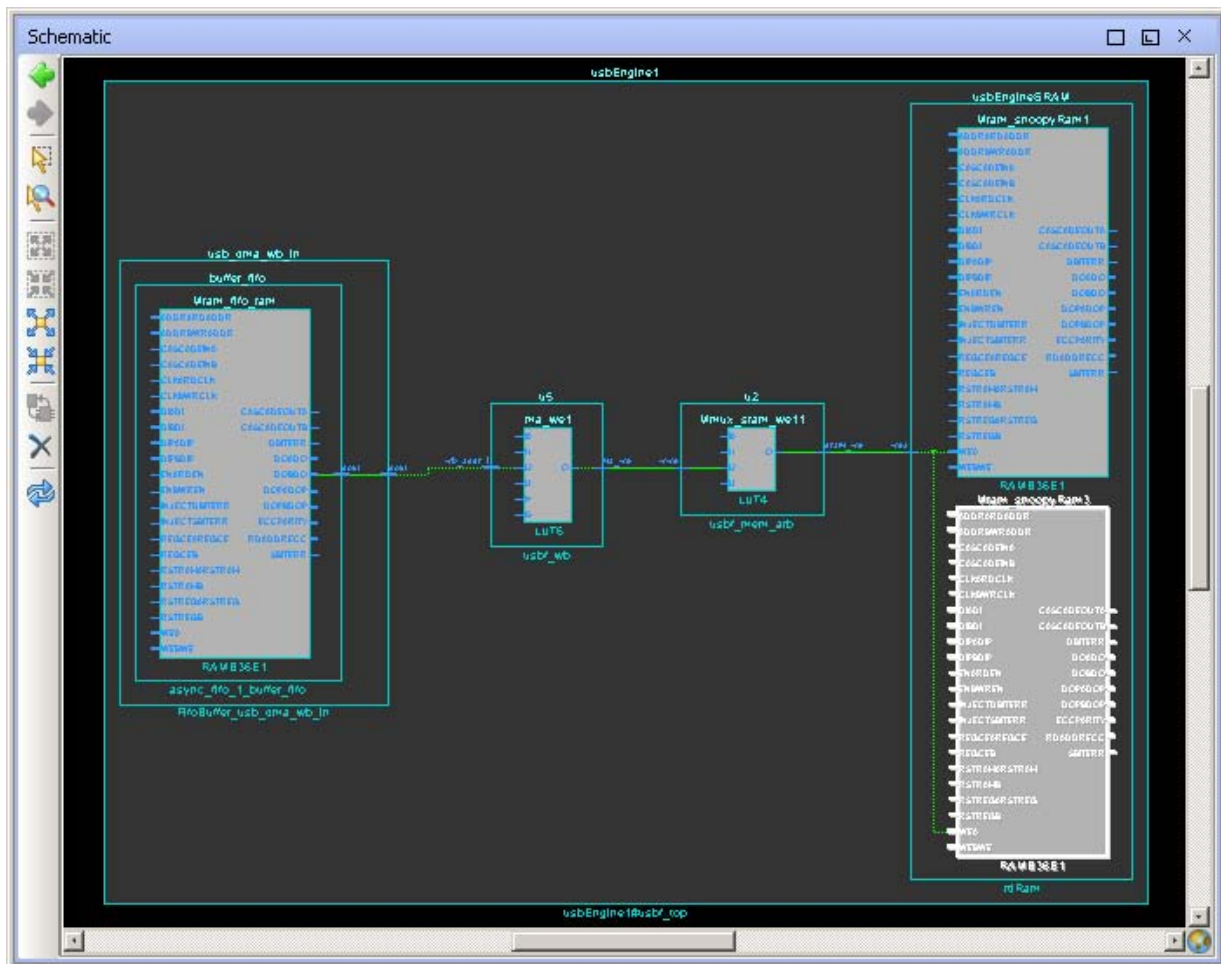


図 11 : クリティカル パスのゲートと階層

フロアプランから、少なくとも `usbEngine1` 内のブロック RAM 間のタイミング クリティカル パスを制約する必要があります。`usbEngine1` がフロアプランのよい対象であると判断できます。`usbEngine1` がチップの大部分を占めている場合は、その代わりにクリティカル パスを含む 4 つのサブ階層をフロアプランします。

どのゲートをフロアプランするかをすばやく判断するには、[Device] ビューで配置を表示します。図 12 では、クリティカルな階層のゲートは緑色、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。クリティカルな階層ではブロック RAM の使用率が高く、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ています。この階層は、デザインの約 20% を占めています。`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。

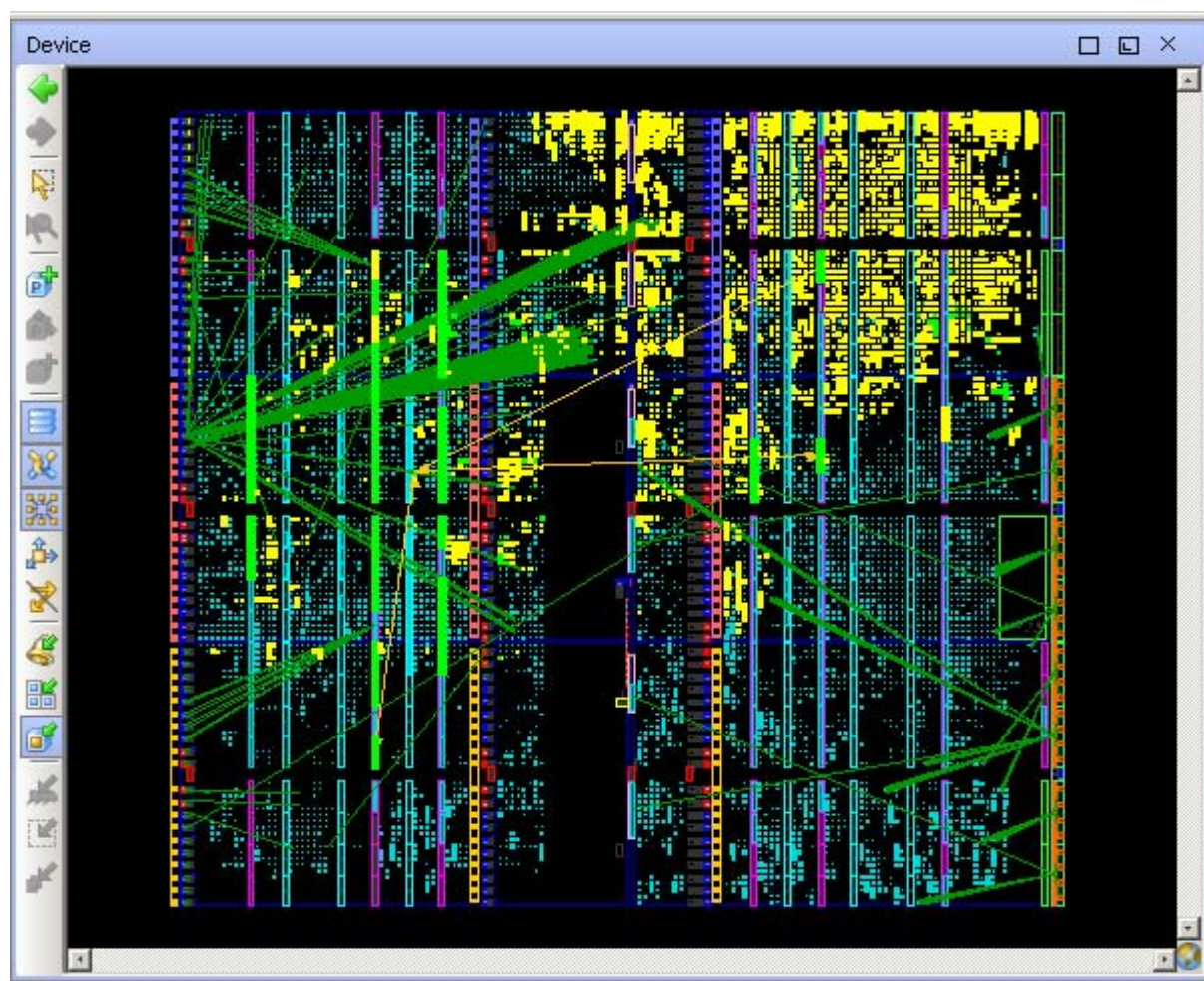


図 12 : usbEngine1 のクリティカル パスとクリティカルでないパス

次に、usbEngine1 がフロアプランのよい候補であるかどうかを確認し、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すると有益です。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。

I/O 接続は、緑色の線で表示されます。図 13 に例を示します。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。図 13 を見ると、接続されている階層が多数あります。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 13 は、このデザインの最上位デザインを示します。1 つの階層の接続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 (白色の四角) 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。

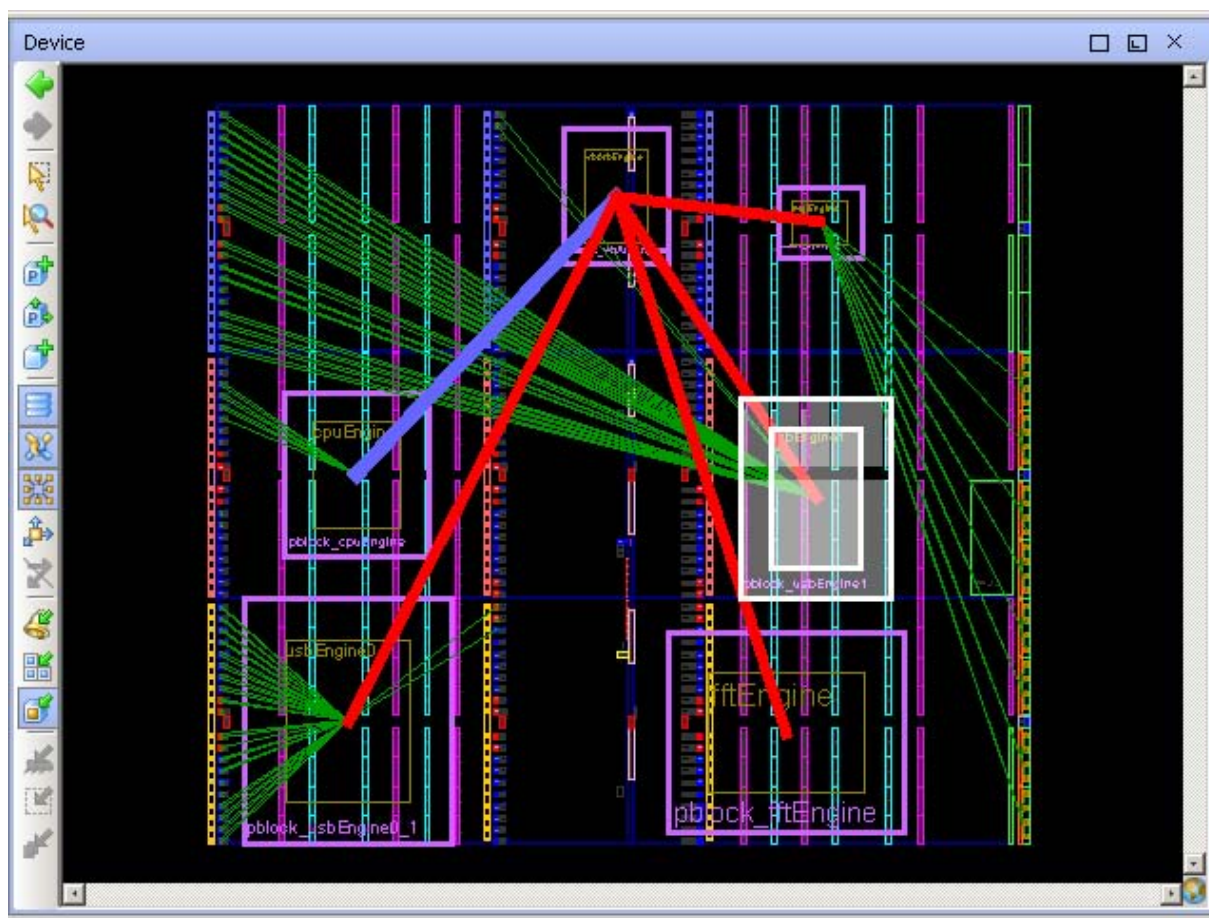


図 13 : 解析用の最上位フロアプラン

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 を使用するように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% を使用するように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 および usbEngine0)。インプリメンテーションで usbEngine1 にタイミング問題があることが示されましたが、usbEngine0 でもタイミング問題が発生すると予測されます。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、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 14](#) に示します。

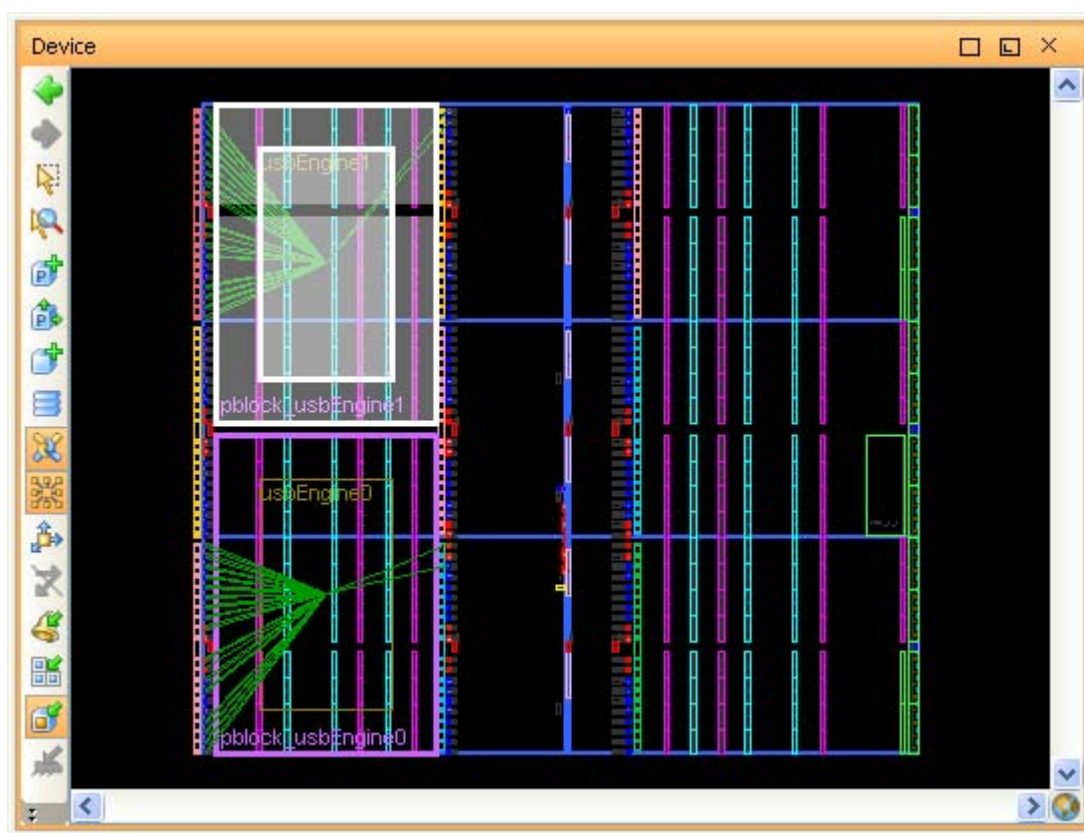


図 14 : 最初のフロアプラン

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。これらの一部のみを制約する領域を設定することも可能です。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。

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、理由を解析します。デザインに、最初の解析でははっきり示され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う便利です。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。

反復フロアプランでのヒントは、次のとおりです。

- 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、新規 Pblock を作成します。クリティカルパスを含む階層レベルを特定し、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 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、Pblock に制約されていないロジックを追加します。1 つの方法は、クリティカルパスを含む Pblock を作成して近くに配置することです。もう 1 つ、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 を作成する方法もありますが、これは制約されていないロジックが小型である場合にのみ使用可能です。
- クリティカルパスが 2 つの Pblock の間にある場合は、Pblock を修正します。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てください。また、一方の Pblock をもう一方の Pblock に組み込んだり、ブロックを一方の Pblock からもう一方の Pblock に移動したりしてみます。
- すべてケースで、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配置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た場合は、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。

まとめ

フロアプランを使用すると、タイミング パフォーマンスが向上し、一貫した結果が得られるようになります。再利用フローでは、タイミングが満たされたデザインで継続して同じ結果が得られるようにします。階層フロアプラン フローは、タイミングが満たされていないデザインのタイミング クロージャに有益で、一貫性も向上します。デザインを変更した場合、これらの方法の再検討が必要な場合があります。