消費電力手法ガイド

UG786 (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	改定履歴
2011 年 3 月 1 日	13.1	初期リリース

このマニュアルについて

この消費電力手法ガイドには、FPGA ロジックを設計してシステムに統合する際に受ける可能性がある消費電力の影響がすべて記載されています。このガイドでは、FPGA 内部また外部の消費電力に影響する電力的要因や物理的要因が示されています。また、デザイン サイクルの任意の段階でデバイスの消費電力を監視して最小限に抑えるフローおよび手法も紹介しています。

このガイドは、次の章で構成されています。

第 1 章「FPGA の消費電力概要」では、FPGA における消費電力を理解するのに必要なさまざまなコンポーネントと FPGA に大規模なシステムを含める際の影響について説明します。また、各種用語の定義、物理的過程、要因について説明します。

第 2 章「ソフトウェアによる消費電力解析」では、FPGA の消費電力の算出方法または概算方法に対するさまざまなアプローチを紹介します。結果がどのように算出されたかを理解できるよう、各アプローチでの利点、必要な入力、想定、および精度と複雑度のトレードオフについて説明します。

第 3 章「消費電力予測手法」では、FPGA の消費電力を概算して大規模システム設計への影響を確認する際のツールおよび手法をデザイン サイクルの各ステップごとに説明します。

第 4 章「消費電力削減のためのヒントおよび手法」では、システムのすべての側面に対して消費電力を最小限に抑える際に使用できる実践的なガイドラインを紹介します。FPGA がインプリメントされるシステムでの考慮事項、HDL コード手法、ソフトウェア設定、および消費電力の最適化アルゴリズムについても説明します。

第 5 章「まとめ」では、ガイドで紹介した情報のサマリを紹介します。

付録 A「その他のリソース」では、このガイドで紹介する情報に関連する資料のリストが含まれています。

目次

第 1 章

FPGA の消費電力概要

この章では、FPGA をボードにインプリメントする際に考慮するさまざまな側面やステップについて紹介します。システム設計という大きな視点から FPGA 開発を捉え、デザイン フローの各段階での消費電力に関連した考慮事項を大まかに説明します。フローの詳細、ヒント、および手法は、第 3 章「消費電力予測手法」および第 4 章「消費電力削減のためのヒントおよび手法」で説明します。

FPGA 消費電力の側面およびシステムの依存性

現代の FPGA は、カスタマイズ可能な大型ロジック アレイに加えてメモリ、DSP、プロセッサー、およびデータを処理して別のチップと通信する多数のブロックが統合された優れたチップです。プリント回路基盤 (PCB) には複雑なチップが多数あるため、FPGA の消費電力は多くの要因に依存しています。FPGA およびユーザー デザインによりシステムの電源要件および放熱要件が作成される一方で、システムの物理的、電気的な要因が FPGA の電源および冷却に影響を与える可能性があります。次のセクションでは、これらの側面を詳細に説明する前に、このガイドで使用される用語および概念について説明します。

電源パス

FPGA に電源を供給するには、複数の電源が必要です。各電源では、さまざまな FPGA リソースで必要になる電力が提供されます。これにより、異なる電圧レベルでさまざまなリソースが動作できるので、ノイズや寄生効果に対して高い耐性を保ちながら、パフォーマンスおよび信号強度を向上できます。

表 1-1 に、各種電源およびその電源が供給されるザイリンクス FPGA 内のロジック リソースを示します。これらの詳細は、ザイリンクス デバイス ファミリによって異なる場合があるので、この表はガイドラインとしてのみ示しています。

表 1-1 : FPGA リソースおよびその電源

電源	電源が供給されるリソース
V_{CCINT} および V_{CCBRAM}(3)	• すべての CLB リソース • すべての配線リソース • すべてのクロック バッファーを含むクロック ツリー全体 • ブロック RAM/FIFO(1) • DSP スライス(1) • すべての入力バッファー • IOB に含まれるロジック エレメント (ILOGIC/OLOGIC)(1) • ISERDES/OSERDES(1) • PowerPC™ プロセッサー(1) • トライモード イーサネット MAC(1) • クロック マネージャー (DCM、PLL など) (大部分は Vccaux で供給されるため少量のみを供給) • MGT の PCIE および PCS 部分
V_{CCAUX} および V_{CCAUX_IO}(3)	• クロック マネージャー (MMCM、PLL、DCM など)(1) • IODELAY/IDELAYCTRL(1) • すべての出力バッファー • 差動入力バッファー • V_{REF} ベースのシングルエンド I/O 規格 (HSTL18_I など) • 位相器
V_{CCO}	• すべての出力バッファー • 一部の入力バッファー • デジタル制御インピーダンス (DCI) 回路 (オンチップ終端 (OCT) とも呼ばれる)(2)
MGT*	• トランシーバーの PMA 回路

メモ :

1. これらのリソースは、一部のデバイス ファミリでのみで使用できます。詳細は、該当するデータシートおよびユーザー ガイドを参照してください。
2. バンク 0 (V_{CCO_0} または V_{0CCO_CONFIG}) の V_{CCO} では、バンク 0 に含まれる I/O すべてとコンフィギュレーション回路に電源が供給されます。詳細は、該当するコンフィギュレーション ユーザー ガイドを参照してください。
3. ザイリンクス 7 シリーズ FPGA のみ

電力の種類

各電源で必要な電力は、次の 3 つです。

- デバイス スタティック (リーク) 電力 : デバイスが動作し、プログラムで使用できるために必要な電力。大部分はデバイスのコンフィギュレーションを保持するために使用されるトランジスタのリークが原因です。
- デザインのスタティック消費電力 : デバイスがコンフィギュレーションされていてアクティビティがないときに継続して消費される電力。この電力には、デザインのアクティビティに関わ

らず使用時に電力が必要になる I/O 終端、クロック マネージャー、およびその他の回路のスタティック電流が含まれています。

- デザイン ダイナミック消費電力 : デザイン アクティビティで発生する電力。この電力は、デザイン アクティビティによって時間と共に変化します。また、この電力は使用する電圧レベル、ロジック、および配線リソースによって異なります。

電力消費パス

FPGA デバイスに供給される総電力は、複数のパスを介して出入力されます。

- 熱電力 : FPGA 内部で消費される電力。熱電力は、デバイスの異なるエレメントをイネーブルにして、デバイス ロジックを切り替えます。このような動作では熱が発生し、デバイスのジャンクション温度が上昇する原因になります。この熱は、環境に伝導します。プリント回路基板にはほかのチップも配置されているため、FPGA 設計者は放熱パスを提供し、ジャンクション温度をデバイス動作範囲に維持するようにする必要があります。
- オフチップ電力 : 電源から FPGA 電力ピン、I/O ピンの順に通り、外部のボード コンポーネントで散逸される電流。FPGA で供給される電流は通常、オフチップの I/O 終端、LED、またはほかのチップの I/O バッファーで消費されるので、デバイスのジャンクション温度の上昇の原因にはなりません。

パワー モード

FPGA は、電源を投入してから切るまでの間にいくつかの電力フェーズを経ます。各フェーズでの電力要件は、異なります。

- 電力投入 : 最初に FPGA に電力が投入されるときに発生する過渡スパイク電流。この電流は各電圧電源で異なり、FPGA 構造および電源ソースの公称電圧への上昇能力に依存しています。また、温度、異なる電源間での優先順位など、デバイスの動作条件にも依存しています。現代の FPGA アーキテクチャで電源投入順序のガイドラインに従っている場合は、このようなスパイク電流を考慮する必要はありません。
- コンフィギュレーション : デバイスのコンフィギュレーション中に必要な電力。アプリケーションの消費電力が極端に低くない限り、コンフィギュレーション電力はアクティブ電力より常に低くなるため、この一時的なフェーズが電源要件に影響することはありません。
- スタンバイ : デバイスがコンフィギュレーションされたときに供給される電力で、外部から適用されるアクティビティや内部で生成されるアクティビティはありません。スタンバイは、デザインの動作中に供給する必要がある最小継続電力を示します。
- アクティブ : デバイスでアプリケーションが実行されている間に必要な電力。この電力には、スタンドバイ (すべてのスタティック消費電力) とデザイン アクティビティにより生成された電力すべて (デザインのダイナミック消費電力) が含まれます。この電力は瞬間的で、入力データ パターンおよびデザインの内部アクティビティによりクロック サイクルごとに変化します。
- サスペンド : すべてのデバイスの電力が切れているときに必要な電力。この電力は、デバイス ファミリによって異なりますが、デバイスのコンフィギュレーションは常に保持されます。ウェークアップ ロジックによりアクティブ ステートに戻ることができます。このモードでは、電力を節約しながら完全機能にすみやかに戻ることができます。
- ハイバネート : 1 つまたは複数の電源ソースが切れているときに必要な電力。この機能は FPGA アーキテクチャによって異なりますが、通常は常時動作する必要がないアプリケーションで消費電力を大幅に節約できます。通常の動作に戻る前にデバイスのプログラムを再実行する必要があるため、アクティブ電力ステートに戻るまでのウェークアップ時間は長くなります。また、

このモードではブロック RAM を含むチップ内すべてのデータがクリアされるので、電源投入時と同じ状態にデバイスが戻ります。

環境的な消費電力の要因

- 供給ストラテジ
  - レギュレーター技術 : 入力と出力の電圧差異、応答時間、最大電流、出力電圧の精度制約を調整するさまざまなレギュレーター技術があります。
  - デカップリング ネットワークのパフォーマンス : デカップリング回路を効率よく設計すると、短時間で高電力を要求する期間に FPGA に電力を供給するだけではなく、レギュレーターからの電流サージ要求を減らして、レギュレーターによる総消費電力を抑えることができます。
  - FPGA の選択 : 電源数および電圧レベル要件は、FPGA ファミリによって異なります。低電圧コアや低電圧 I/O インターフェイスをサポートするデバイスを選択すると、消費電力を抑えることができます。
- 冷却ストラテジ
  - システム環境 : システム エンクロージャの形状および寸法とその周囲温度は、生成された熱を環境に移す際に考慮する主要な側面です。
  - ヒート シンク : ヒート シンクの寸法、形状、およびマウントと関連する強制エアフローシステムにより FPGA から抽出可能な熱量が決定します。
  - パッケージ選択 : パッケージの寸法、材質、およびボードへの接続は、コストとシグナル インテグリティに加え、生成された熱が上面と下面の両方からどのように環境に移されるかにも影響します。ヒート シンクとボード間で接触面が大きいほど、熱抵抗が低くなります。
  - コンポーネントの配置 : システム エンクロージャに加え材質、アセンブリ、およびその近隣のコンポーネントなどの構成パラメーターに応じてコンポーネントを配置すると、環境への熱の移り方が変わります。たとえば、障害物を 1 つ配置すると FPGA 付近のエアフローが減少する可能性があります。FPGA の近辺に配置されているその他の熱を発するコンポーネントは、デバイス上部の外気を加熱したり、ヒート シンクの効率を下げたり、またはボード材を介して FPGA に熱を移したりする可能性があります。

デバイスの消費電力の要因

- 製造パラメーター
  - シリコン技術 : ザイリンクスのデザイン チームでは、シミュレーションと解析を精密に行い、製造プロセスにおけるパラメーターを慎重に選択しています。望ましい機能、コスト、パフォーマンス、および信頼性の間で適切なバランスを取るには、さまざまなトランジスタのサイズ、強度、電圧、および配列が要求されます。これにより、トランジスタの極性化または切り替えに必要なエネルギーが最適化されます。ザイリンクスでは、デバイス I/O の電気特性をシステムでシミュレーションできるようビヘイビアーおよびトランジスタ レベルのモデルが提供されます。これらのモデルを入手するには、http://japan.xilinx.com/support/download/index.htm で [デバイス モデル] タブをクリックし、[Model Type] の下から任意モデルをクリックしてください。
  - パッケージ技術 : ザイリンクスでは、デバイス コアおよび I/O への電流の流れに影響するパッケージ技術および物理的/電気的レイアウトのパラメーターを慎重に選択しています。また、形状や材質などのパラメーターは、デバイスで生成される熱がどのようにパッケージの上面および下面の両面から環境に移されるかを定義します。特定のシステムでパッ

ケージの電子的プロパティを評価できるよう、ザイリンクスでは IBIS フォーマットでピンごとの RLCG モデルを提供しています。選択したパッケージの熱動作をシミュレーションするための小型の熱モデルも提供されています。これらの熱モデルは、http://japan.xilinx.com/support/download/index.htm の [デバイス モデル] タブをクリックすると、[パッケージ温度モデル] から入手できます。

- アーキテクチャのパラメーター

  ザイリンクス チームでは、多数のパラメーターが存在する中で、タイプ、機能、量、レイアウト、およびさまざまな FPGA リソース間の接続性を定義しています。相反する要因間でバランスを取るのは、非常に困難です。最も消費電力を節約できるようなアーキテクチャのパラメーターを選択することで、デバイス コストやパフォーマンスに影響したり、ソフトウェアのインプリメンテーション アルゴリズムがさらに複雑になる可能性があります。これらの選択は、デバイスのスタティック消費電力およびダイナミック消費電力に大きく影響します。

  最近のデバイスの改善点 :

  - より多くの機能をフィットできるようにスライスや基本ロジック構造を調整。小さいエリアにより多くのロジックをパックすることで、通常はロジックおよび配線のダイナミック消費電力が節約されます (4 入力 LUT の代わりに 6 入力 LUT)。
  - デザインでよく使用される機能のハード化。専用ブロック (クロック ジェネレーター、PCIe、メモリ コントローラー) を追加することで必要になる FPGA プログラマブル ロジックまたは外部コンポーネントが減り、システムの消費電力を削減。
  - ソースからデスティネーションへのホップ数を最小化するよう配線構造を最適化。
  - ファンアウトの大きいネット (クロック、リセット、クロック イネーブル) を使用しないときにディスエーブルにできるようゲート付きバッファーまたはポートを追加。
  - デバイス全体または個々のロジック ブロックに適用されるパワー ダウン モードを追加。これにより、デザインの未使用部分の電源を切って必要なときのみイネーブルにできます。
  - 低電圧 I/O インターフェイスのサポートの追加。電圧は、I/O のスタティック消費電力およびダイナミック消費電力の両方を駆動する重要な側面です。

デザイン消費電力の要因

- デバイスの選択
  - 適切なデバイス ファミリ : ベンダーおよびデバイス ファミリによって提供されるロジックおよび I/O 機能は異なります。アプリケーションに最適なブロック サイズ、コンフィギュレーション、およびリソース数のデバイスを選択することで、アーキテクチャでのエレメント使用量が最適化され、スタティック消費電力およびダイナミック消費電力が削減されます。
  - 適切なファミリ メンバー : デバイスのサイズは、ほとんどの場合でデバイスのスタティック消費電力に影響します。大きすぎたり小さすぎるデバイスを選択すると、RTL 記述のマップが最適にならず、ダイナミック消費電力という点から効率性が下がります。
- RTL 記述

  デザイン記述は、使用可能なリソースへの論理式のマップ方法に影響します。アーキテクチャ エレメント、ポート、コンフィギュレーション オプション、およびモードをよく理解することで、エンベデッド リソースを最大限に利用できます。
- ツールの制約

  デフォルトでは、インプリメンテーション ツールはパフォーマンス目標を達成し、デバイスの使用率を最小限に抑えるものです。コアおよび I/O ロジックの両方にパフォーマンスと使用率

に対して現実的で完全な一連の制約を提供することで、ツールが最適に実行されます。これにより、ダイナミック消費電力が最小限に抑えられます。

- インプリメンテーション ツールのオプション

  インプリメンテーション ツールには、消費電力を抑える複数のアルゴリズムがあります。ほとんどはオプションで、複数のストラテジを使用してスイープできます。制約を理解すると、どの制約を使用する必要があるのか判断する際に役立ちます。

FPGA 消費電力と全般的なシステム デザイン プロセス

プロジェクト考案から完成までは、消費電力に影響するさまざまな側面を考慮する必要があります。ほかのすべての制約 (機能、パフォーマンス、コスト、およびタイム トゥ マーケット) を一時的に除外すると、消費電力に関連するタスクを次の 2 つに分類できます。

- 物理的領域 : エンクロージャ、ボードの形状、電力分配システム、熱電力の散逸システム
- 論理的領域 : エリア パフォーマンス、I/O インターフェイスの統合

次の章では、物理的領域と論理的領域が相互依存していることについて説明します。ただし、前者はハードウェア、後者は FPGA の論理デザインで判断に関与するという点で異なっています。通常、ハードウェアの選択およびサイズ設定は、プロトタイプ ボードを構築できるよう、デザイン フローの初期段階で行います。FPGA の機能による消費電力への影響は早期に予測可能なので、デザイン ロジックの完成度に伴って改善できます。図 1-1 に通常のシステム デザイン プロセスを示し、電力に関連する判断箇所をハイライトします。この図では、デバイスおよび関連冷却パーツを選択する時点では、FPGA ロジックがまだ完成していないことがわかります。このため、FPGA ロジックの電力要件を予測する手法が必要です。これらの手法については、第 3 章「消費電力予測手法」で説明します。

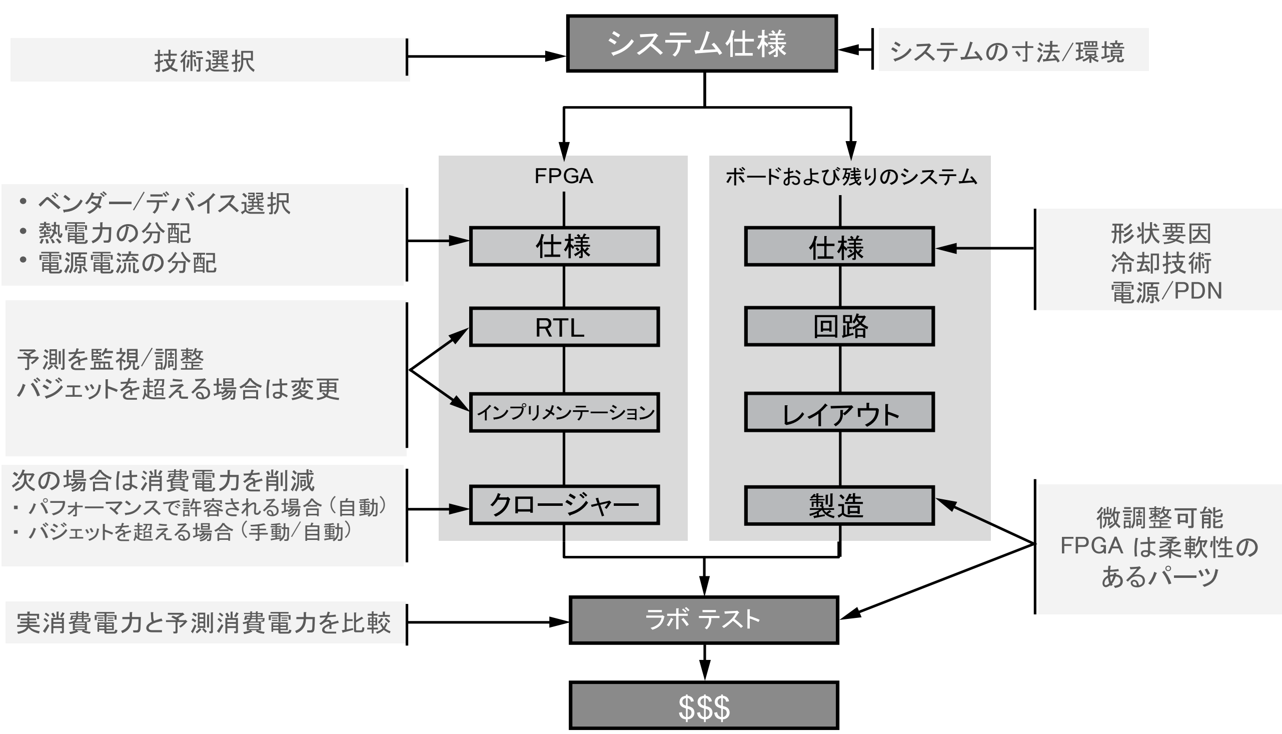

図 1-1 : システム デザイン プロセスでの消費電力面の管理

次の 3 つの章では、電力に影響する各側面での主な要因について説明します。

ザイリンクス消費電力予測/解析ツール

ザイリンクスでは、FPGA の熱および電源要件をデザイン サイクルを通して評価できるようツールおよび資料が提供されています。図 1-2 に、FPGA の各デザイン サイクルで使用できるツールを示します。ツールには、スタンドアロンのものとインプリメンテーション ソフトウェアに統合されているものがあり、後者はデザイン プロセスの各段階で利用できる環境および情報と協調しています。すべてのツールには通信チャネルがあり、解析が効率よく行えるように情報を交換できます。

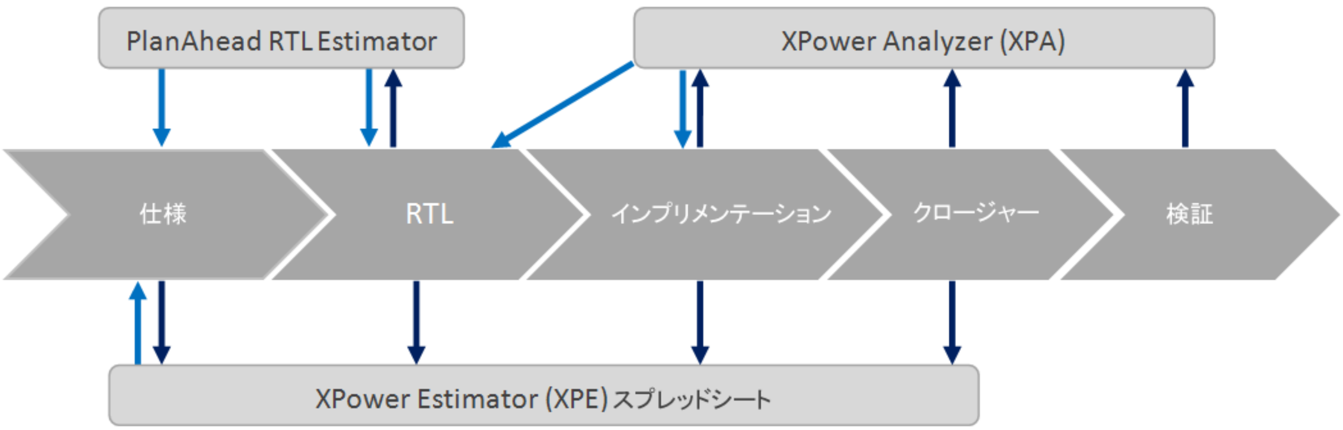

図 1-2 : FPGA デザイン プロセスに含まれるザイリンクス消費電力予測/解析ツール

XPower Estimator (XPE)

XPower Estimator (XPE) スプレッドシートは、通常プロジェクトの設計前とインプリメンテーション前の段階で使用される消費電力予測ツールです。XPE では、アーキテクチャの評価とデバイスの選択が支援され、またアプリケーションで必要になる可能性がある適切な電源や熱管理コンポーネントを選択できます。XPE インターフェイス (図 1-3) では、デザインのリソース使用量、アクティビティ レート、I/O 負荷、および予測される電力分配を計算するためにデバイス モデルと組み合わせるその他多くの要因を選択できます。

また XPE は、デザイン サイクル後半のインプリメンテーションおよびパワー クロージャ中にもよく使用され、たとえばエンジニアリング チェンジ オーダー (ECO) の消費電力への影響を評価するときなどに使用されます。複数のチームによりインプリメントされる大型デザインでは、プロジェクト リーダーが XPE を使用して各チームのモジュールの使用量およびアクティビティをインポートして総消費電力を監視し、制約が満たされるように電力バジェットを割り当て直すことができます。

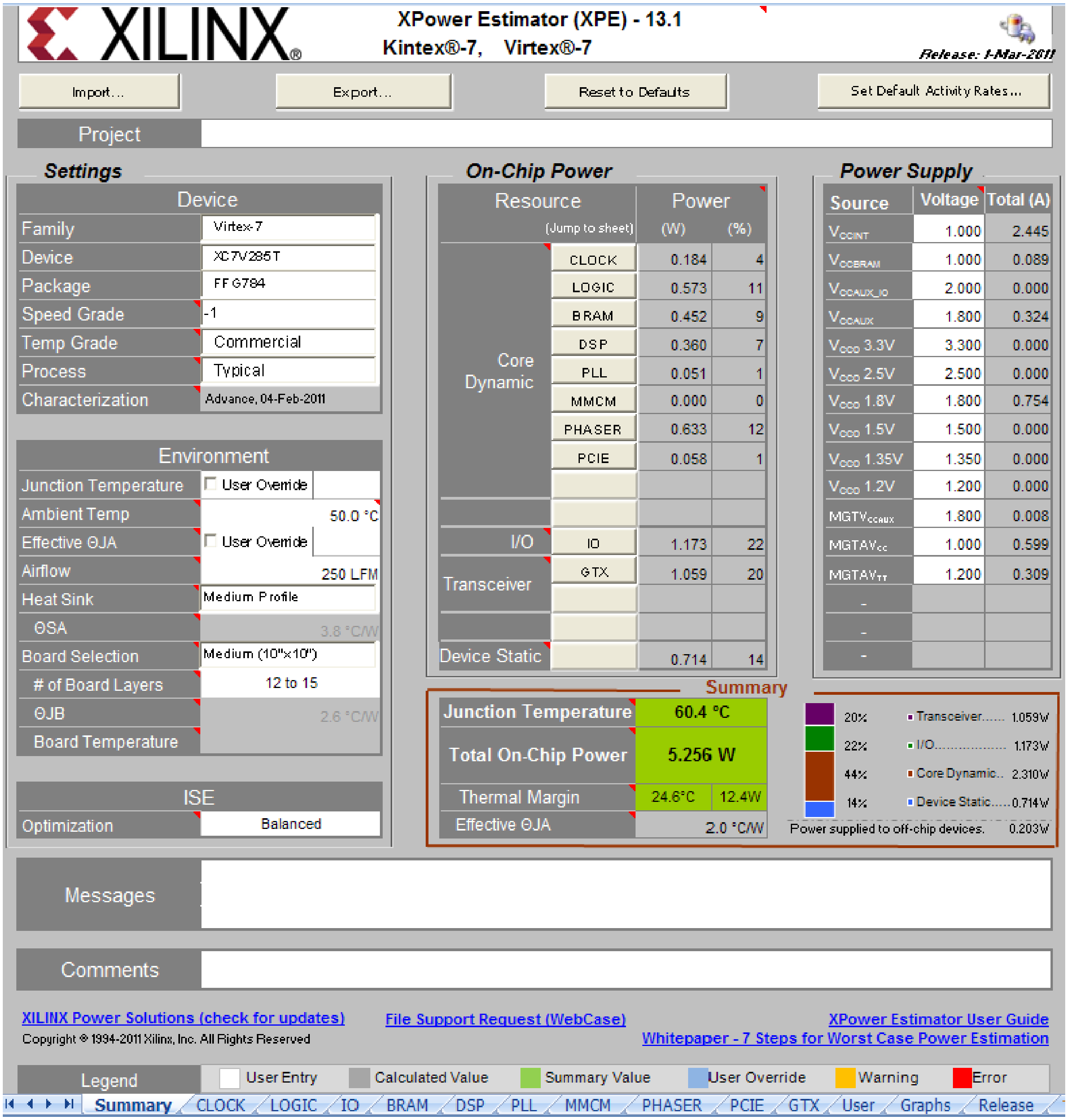

図 1-3 : XPower Estimator スプレッドシート

XPower Analyzer (XPA)

XPower Analyzer (XPA) ツールでは、インプリメンテーション後に消費電力予測を実行できます。使用されたロジックおよび配線リソース情報が含まれるインプリメント済みのデザインのデータベースから読み出すことができるので、精度が最も高くなります。図 1-4 に、消費電力レポートのサマリを表示します。クロック ドメイン、リソースの種類、またはデザイン階層など、さまざまなビューを使用してデザインの消費電力を確認できます。また、XPA では環境設定やデザイン アクティビティを変更でき、デザインの電源および熱電力消費をどのように減らすかを検討できます。

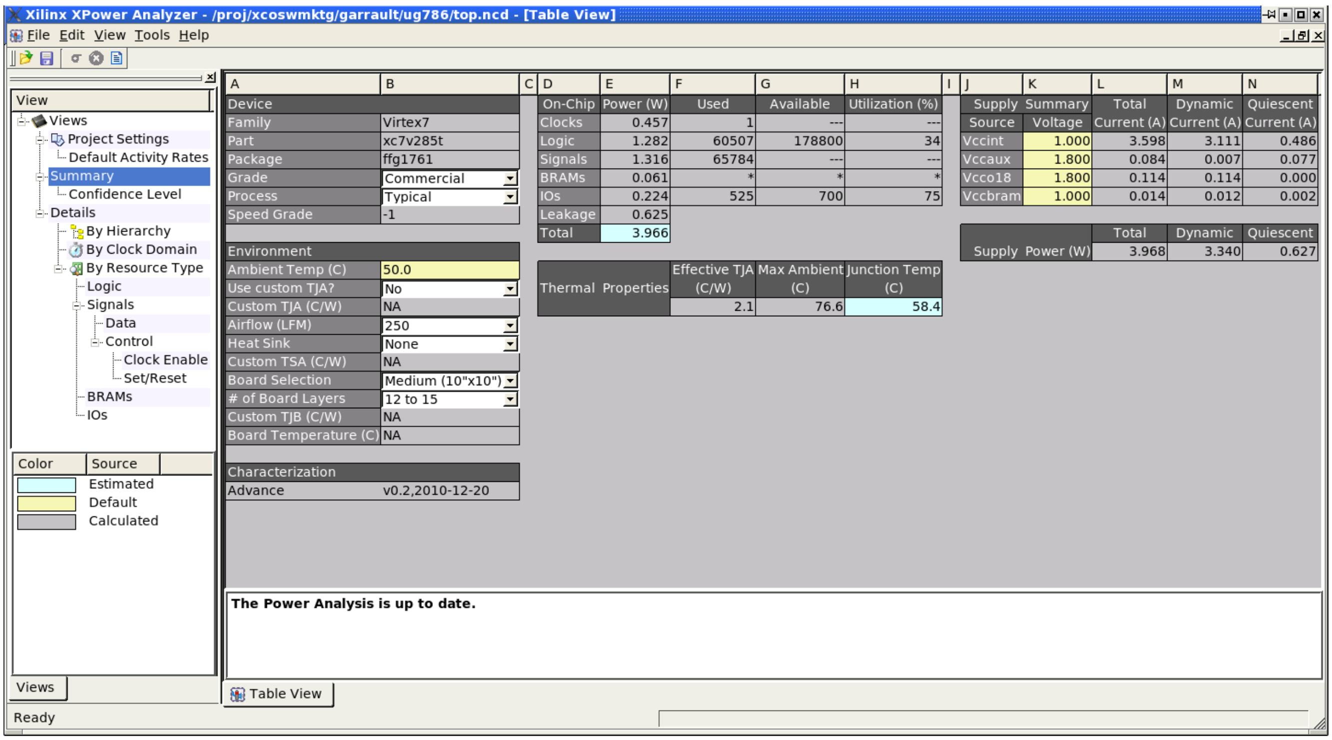

図 1-4 : XPower Analyzer

PlanAhead RTL Power Estimator

PlanAhead ソフトウェアでは、消費電力予測を実行して RTL レベルでのデザインの電力分配を確認できます。デバイスの動作環境、I/O プロパティ、およびデザインのデフォルトのアクティビティ レートは制約を付けるか、または GUI から指定できます。PlanAhead では、HDL コードが読み出されて必要なデザイン リソース数が予測され、各リソースのアクティビティの統計解析に基づいて予測消費電力がレポートされます。図 1-5 に、レポートおよびリソース/階層ビューが表示されています。これらのビューでナビゲートして、電力分配を解析できます。RTL Power Estimator でアクセスできるデザイン意図に関する情報は比較的多いので、XPower Estimator スプレッドシートと比べると精度は高くなり、XPower Analyzer を使用して実行された配置配線後の解析と比べると低くなります。

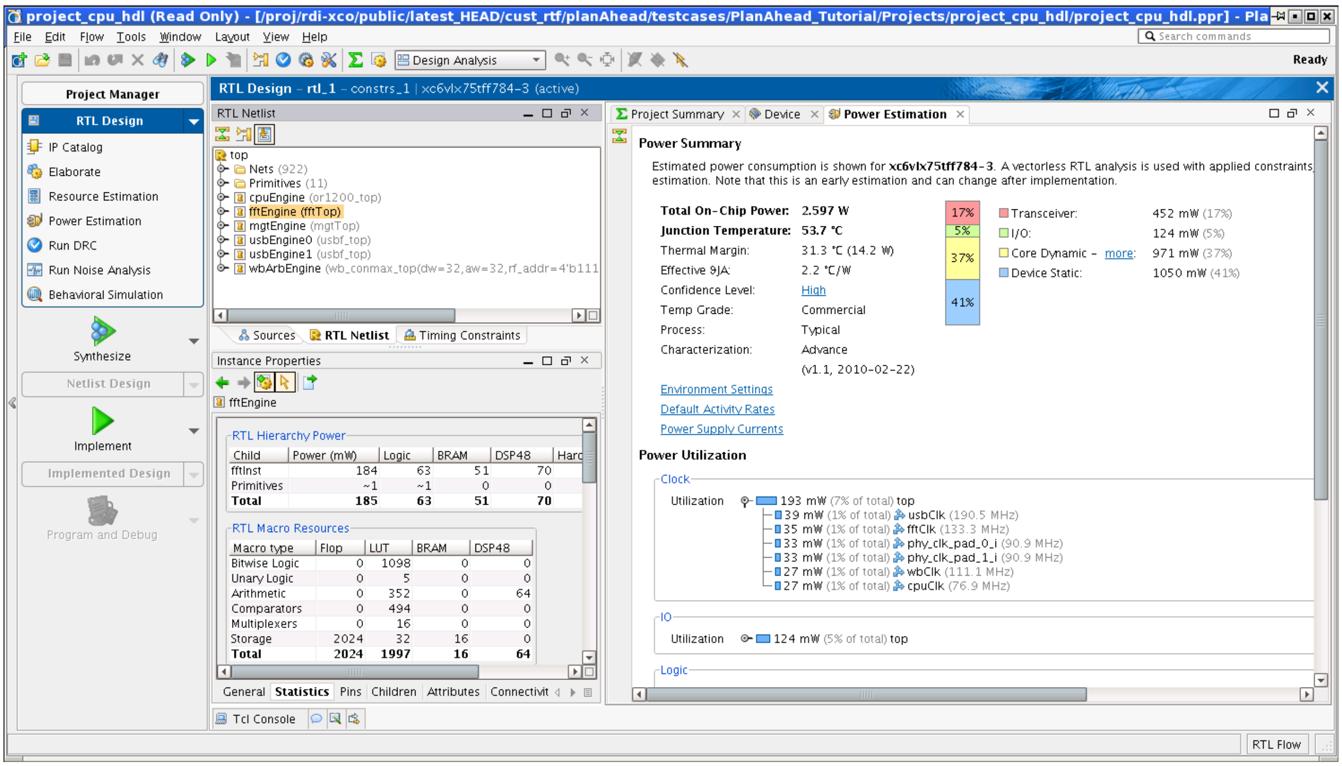

図 1-5 : PlanAhead RTL Power Estimator

第 2 章

ソフトウェアによる消費電力解析

第 1 章では、FPGA で必要になる総消費電力に影響するさまざまな要因について説明しました。この章では、これらの要因がどのように測定されてソフトウェア ツールでの計算に使用されるかについて説明します。デバイスのリソース使用量、コンフィギュレーション、およびアクティビティに関する情報は、デザイン サイクルの初期段階では明らかではないので、ソフトウェアではデフォルトの値が使用されます。デザインのインプリメンテーションの進行に伴い、ソフトウェアに供給できるこのような情報が増えるので、消費電力予測の精度が高くなります。ザイリンクスでは、一連のユーザー デザインに加えて WebTalk プログラムで収集された匿名データを使用して、統計的に現実的な値をツールのデフォルト値に決定しています。これらの値は、パラメーターが自動的に算出されないときやデータがユーザーにより供給されなかったときに使用されます。

消費電力計算

FPGA の各電圧ソースの総消費電力は、次のように計算されます。

> アクティブ電力 = デバイスのスタティック消費電力 + デザインのスタティック消費電力 + デザインのダイナミック消費電力

次に、総消費電力の内訳となる要素について説明します。

- デバイスのスタティック消費電力

  デバイスのスタティック消費電力の大部分は、製造、プロセス プロパティ、適用される電圧、およびデバイスのジャンクション温度に依存しています。ジャンクション温度自体は、周囲温度、電圧レベル、および供給される総電流に依存しています。ただし、供給される総電流には、デバイスのスタティック消費電力が含まれているので、はっきりとした循環依存があります。ツールでは、計算を連続的に繰り返して、指定されている動作条件での実スタティック消費電力の概算値が算出されます。

- デザインのスタティック消費電力

  I/O 終端、トランシーバー、ブロック RAM、およびクロック ジェネレーターなど、FPGA の一部のブロックはデフォルトでディスエーブルにされていますが、デザイン要件によってはイネーブルにされます。これらのブロックがイネーブルのときは、ユーザー デザインのアクティビティに関わらず一定の電力が消費されます。この消費電力は、回路のコンフィギュレーションによって異なります。ソフトウェアでは、デザイン全体のスタティック消費電力に含められる各回路の電力をモデルできます。このモデルでは、リソース レベルのコンフィギュレーション設定の多数が考慮され、電圧およびデバイスの外部環境設定によって変化します。外部環境設定では、環境へどのように熱が放散されるか決まります。

- デザインのダイナミック消費電力

  デザインのダイナミック消費電力は、キャパシタンスおよび使用されるリソースのアクティビティに大部分が依存しており、また適用される電圧レベルと共に計測されます。ソフトウェアのデバイス データベースでは、各リソースのキャパシタンスがそのコンフィギュレーションと接続性に従ってモデル化されます。ユーザーによりアクティビティが供給されない場合は、すべてのコンポーネントの消費電力が算出、集計される前にソフトウェア　アルゴリズムによりネットリストに含まれる各ノードのアクティビティが予測されます。

メモ : デバイスのスタティック消費電力とデザインのスタティック消費電力を加算すると、第 1 章の「パワー モード」で定義されているスタンドバイ電力になります。

熱計算

デバイスのジャンクション温度またはシリコンの温度は、次のように計算されます。

ジャンクション温度 = 周囲温度 ＋ 熱電力 * 外気への効果的な熱抵抗

次に、上記の式に含まれる変数について説明します。

- ジャンクション温度 (°C)

  シリコンの温度。デバイスの選択時に温度グレードを選択しますが、このグレードによりデバイスが指定通りに動作することが保障される温度範囲が決まります。動作条件がグレードの最大値を超えているのに、絶対最大温度以下のままにする場合は、デバイス動作が保障されなくなります。絶対最大動作条件を超えると、デバイスが破損する可能性があります。

  この式を逆側から行うときは、ジャンクション温度をデバイス最大に設定すると、ユーザー環境でデバイスが生成できる最大電力を判断できます。さらに正確に言うと、ツールでスタティック消費電力およびダイナミック消費電力の両方が供給されるので、ワーストケースのデバイスリークとユーザー アプリケーションで生成できる最大ダイナミック消費電力を決定できます。

- 周囲温度 (℃)

  予期されるシステム動作条件下でデバイスを直接取り巻く外気の温度

- 熱電力 (W)

  FPGA 内部で消費される電力。熱が発生し、デバイスのジャンクション温度を上昇する原因になります。

- 外気に対する効果的な熱抵抗 : Θ_{JA} (°C/W)

  この係数は、FPGA シリコンから環境 (デバイス ジャンクションから周囲外気) へ電力が放逸される度合いを示します。シリコン チップの寸法から周囲外気まで、またその間のパッケージ、PCB、ヒートシンク、エアフローなど、すべての要素の寄与が含まれています。

  - チップから上方向に向かい外気へ (ジャンクションから外気または Θ_{JA})
  - チップから下方向に向かいボードを通って外気へ (ジャンクションからボードまたは Θ_{JB})

消費電力モデルの精度

ツールに組み込まれている特性データの精度は、使用可能なデバイスまたは製造プロセスの成熟度を反映させるため、時間と共に進化します。この精度表示は、[Characterization] フィールドに表示されます。デバイス ファミリの特性データは、次の順序で進化します。
Advance → Preliminary → Production

Advance

この表示のデバイスには、主にシミュレーション結果または初期製品デバイス ロットからの測定値に基づいたデータ モデルがあります。このデータは通常、製品リリースから 1 年以内に入手できます。データは比較的安定しており、余裕を持たせた設定ですが、一部の値が高すぎたり低すぎたりする可能性があります。この仕様のデータは、Preliminary や Production 仕様のデータほど正確ではありません。

Preliminary

完成している初期製品シリコンに基づいています。この仕様では、デバイス ファブリック内にあるほぼすべてのブロックが特性評価されています。Advance 仕様と比較すると、消費電力値の精度は高くなります。

Production

この仕様は、特定のデバイス ファミリの十分な量産を経た上で特性評価が行われ、相当数の生産ロットを対象とした完全な電力相互関係が確立された後にリリースされます。この特性データを持つデバイス モデルは、これ以上進化しません。

アクティビティ予測

ネットリストに含まれる各ノードのアクティビティは、次の 2 つのパラメーターで表現されます。

- 信号遷移レート：解析中に考慮するエレメントがステートを変更した回数を定義します。この値が発生したポジティブ エッジとネガティブ エッジの回数の合計になります。ザイリンクス ツールでは、この数値が Mtr/s (Millions of Transitions per second : 億遷移/秒) で表現されます。
- 信号スタティック確立レート：解析で考慮するエレメントが論理レベル High で駆動される期間を割合で定義します。このレートは、信号割合 High レートとも呼ばれます。

デザインのノート アクティビティは、正確な消費電力予測を得るための重要な側面の 1 つです。デザインの完成度によっては、次のセクションに示すように値が判明している場合やツールにより計算したり予測できる場合があります。通常は、消費電力計算でのアクティビティ レートの影響を考慮し、判明している値はすべて指定する必要があります。アクティビティが不明のときは、ツールで予測するようにするのが最善策です。

ユーザー入力

どのデザインでも、通常は特定のノードのアクティビティが判明しています。これらは、システムの仕様または FPGA が通信するインターフェイスにより強制されたりするためです。特に、FPGA の複数セルを駆動するノード (セット、リセット、クロック イネーブル、またはクロック信号) に対する情報をツールに供給することで、電力予測アルゴリズムに役立てることができます。

これらのノードには、次が含まれます。

- クロック アクティビティ : すべての FPGA クロック ドメインの正確な周波数、外部供給なのか (入力ポート)、内部で生成されるのか、または外部からプリント回路基盤に供給されるのか (出力ポート) は、通常判明しています。
- I/O データ ポート : FPGA からのデータの入出力の的確なプロトコルおよびフォーマットが判明しているので、通常はツールで少なくとも一部の I/O に対して信号の遷移レートや信号割合 High レートを指定できます。たとえば、一部のプロトコルに DC バランス要件があったり (信号割合 High レートが 50%)、メモリ インターフェイスへのデータの書き込みおよびメモリ インターフェイスからのデータの読み出しの頻度が判明している場合は、ストロボ信号およびデータ信号のデータ レートを設定できます。
- I/O および内部制御信号 : システムおよび予期する機能が判明しているので、セット、リセット、およびクロック イネーブルなどの制御信号のアクティビティを予期できる場合があります。これらの信号は通常、デザイン ロジックの広範囲部分をオン、オフにできるので、これらの情報を供給することで消費電力予測の精度が高くなります。

シミュレーション

通常は、デザイン開発のすべての段階と並行してシミュレーションを実行し、デザインが予測どおりに動作するかを検証します。デザインの開発段階、複雑性、または会社の方針に応じてさまざまな検証手法があります。次に、取り込むことができる有益なデータについて説明します。また、このデータを使用して消費電力を解析する際に陥りやすい過ちについても説明します。正確な消費電力予測を実行するには、アクティビティ レートが現実的である必要があります。アクティビティレートでは、シミュレーションされるブロックに入力されるデータに対して通常のシナリオまたはワースト ケースのシナリオが示されるべきです。このような情報は、検証やファンクションの認証中には必ずしも渡されません。場合によっては、無効なデータが入力されることがあり、無効なデータやコマンドが入力されたときでも、システムが対処して安定したままの状態でいられるかが検証されます。このようなテスト ケースを使用して消費電力解析を実行すると、デザイン ロジックが通常のシステム動作状況でシミュレーションされないため、消費電力予測が不正確になります。

- システムのトランザクション レベル : デザイン サイクル初期に、プリント回路基板上のデバイス間または FPGA アプリケーションの異なるファンクション間で発生するトランザクションを記述している場合があります。この記述から、特定の I/O ポートおよびほとんどのクロック ドメインのアクティビティをファンクション ブロックごとに抽出できます。この情報は、XPower Estimator (XPE) スプレッドシートの入力の際に役立ちます。
- FPGA 記述レベル : アプリケーションの RTL を定義するときは、ビヘイビアー シミュレーションを実行して機能を検証する必要がある場合があります。この情報は、データ フローおよびクロック サイクルに対する計算の有効性を検証する際に役立ちます。この段階では、使用する FPGA リソースの的確な数およびコンフィギュレーションがまだ不明です。リソース使用量を推定して、I/O ポートまたは内部制御信号 (セット、リセット、クロック イネーブル) のアクティビティを抽出できます。この情報を XPower Estimator スプレッドシートに適用すると、情報を改善できます。また、HDL を PlanAhead RTL Power Estimator に読み込むこともできます。このツールはデバイス使用量とアクティビティを予測して即座に消費電力予測を実行するツールですが、詳細は、『PlanAhead ユーザー ガイド』(UG632) の「消費電力予測」のセクショ

ンを参照してください。シミュレーターでは、ノード アクティビティを抽出して SAIF ファイル フォーマットでエクスポートできます。このファイルは、デザイン フローの後半で使用できるように保存できます。たとえば、インプリメンテーション後のシミュレーションを実行するつもりがない場合などは、配置配線後にこのファイルを使用できます。

- FPGA インプリメンテーション レベル :インプリメンテーション プロセスの任意の段階でシミュレーションを実行すると、さまざまな消費電力に関連した情報を取得できます。この情報を使用して XPower Estimator スプレッドシートの情報を改善できます。また、I/O ポートおよび特定のモジュールのアクティビティも保存できるので、デザインを完全に配置配線した後に XPower Analyzer でこの情報を再利用できます。
  - 合成後 : ネットリストがターゲット デバイスで使用可能な実際のリソースにマップされます。
  - 配置後 : ネットリスト コンポーネントが実際のデバイス リソースに配置されます。このパック情報を基に最終的なロジック リソース数およびコンフィギュレーションが判明するので、XPower Estimator スプレッドシートで情報を更新できます。
  - 配線後 : 配線が完了すると、使用される配線リソースに関するすべての詳細およびデザインに含まれる各パスの的確なタイミング情報が定義されます。シミュレーターでは、インプリメントされた回路の機能をベスト ケースおよびワースト ケースのゲートおよび配線遅延で検証することに加え、グリッチを含む内部ノードの的確なアクティビティがレポートされます。このレベルの消費電力解析の精度では、プロトタイプのボードで消費電力を実際に計測する前で最も高くなります。

統計予測

デザイン ノードのアクティビティがユーザーまたはシミュレーション結果から提供されないとき、ベクターレス消費電力予測アルゴリズムでこのアクティビティを推測できます。ベクターレス エンジンでは、デフォルトの信号レートおよびスタティック確立がすべての未定義ノードに割り当てられます。次に、アクティビティがデザインの主要入力から内部ノードの出力まで伝搬され、主要出力に到達するまでこの操作が繰り返されます。このアルゴリズムでは、ネットリスト コンポーネント間のタイミング関係または論理関係は識別されませんが、デザインの接続性、リソースの機能、およびコンフィギュレーションが識別されます。ヒューリスティクスにより、ネットリストに含まれるどのノードのグリッチ レートでも予測できます。グリッチは、デザイン エレメントがアクティブなクロック エッジ間で最終的な値に落ち着くまでに数回ステートが変わるときに発生します。ベクターレス伝搬エンジンは、時間が比較的かかる配置配線後のシミュレーションほどは正確ではありませんが、精度と計算速度という相反する 2 点間でバランスが取れた優れた方法です。

使用量予測

リソースの使用量は、ターゲットしている FPGA で使用される総消費電力を決定する重要な要素です。

- システム仕様レベル : コードを開発する前や、コードを取得したり以前のデザインから再利用したとき、またはウィザードやコア生成ツールを使用して生成したときを除き、インプリメントするさまざまなブロックのリソース使用量を手動で予測する必要があります。この予測は、経験および予期する機能の知識に基づいて行います。この予測の精度は、ユーザーが使用できる情報量やこの情報の XPower Estimator スプレッドシートへの入力に割くことできる時間によって異なります。
- RTL 記述レベル : ザイリンクスでは、RTL コードを読み込むことが可能なエリア エスティメーターが提供されています。このツールに HDL、IP コア、およびブラック ボックスをすべて供給することで、すべてのパラメーターがエラボレートされて 1 つのネットリストに含められます。次にパターン認識が行われて、推論されるリソース数およびこれらのリソースがどのようにターゲット アーキテクチャにマップされるかが予測されます。合成済みまたは配線済みのネットリストに比べると予測精度は落ちますが、アプリケーション コードの開発中に行うデバイス リソース使用量の推測作業の多くが省かれます。これにより、開発プロセスの初期段階で予測される消費電力を監視できます。初期段階では、変更には時間がかからず、効果も高く、リスクも少なくて済みます。
- 合成後レベル : 合成ツールでは記述からロジック構造が推論されて、この RTL が特定のデバイス リソースにマップされます。ネットリストは、パフォーマンス要件とエリア要件を満たすように最適化されます。デバイス リソース使用量の情報がさらに細かく提供されるので、消費電力予測を監視、調整する際に役立ちます。
- インプリメンテーション レベル : マップ後および配置配線後の消費電力予測では、ロジックの自動削除 (trimming)、レジスタの複製、リタイミングなど、最終的なロジック リソース使用量に影響するネットリストの最適化が考慮されるため、精度がさらに高くなります。実際の配線リソース使用量およびレイアウトも、消費電力予測ツールで確認できます。

第 3 章

消費電力予測手法

この章では、第 1 章「FPGA の消費電力概要」で示したフローについて詳しく説明します。典型的なデザイン サイクルの各ステップに対してアプリケーションの電力消費を評価する方法を示し、消費電力予測を自動化または単純化するツールの機能について説明します。消費電力予測を実行した後は、次の章に進み、システムを調べて変更し、デバイスの消費電力を最小限に抑える手法を学びます。この章で説明するツールの詳細は、次の資料を参照してください。

- 『XPower Estimator ユーザー ガイド』(UG440)
- 『PlanAhead ユーザー ガイド』(UG632) (第 5 章「RTL デザイン」の「消費電力予測」)
- XPower Analyzer に関する資料
    - グラフィカル インターフェイス : XPower Analyzer (XPA) ヘルプ
    - コマンド ライン ツール (**xpwr**) : 『コマンド ライン ツール ユーザー ガイド』(UG628)

インプリメンテーション前の消費電力予測

状況説明

この段階では、アプリケーションにとって FPGA が最も効率が優れた技術であるということが判断されています。ここでは、要求される機能、パフォーマンス、コスト、および消費電力バジェットに合うベンダー、ファミリ、およびパッケージを定める必要があります。消費電力という点から言うと、ロジックがまだ 1 つも開発されていない段階でもデバイスの総消費電力を予測する必要があります。総消費電力要件を理解すると、電力分配および冷却仕様を定義する際に役立ちます。電源はいくつ必要か、各電源で使用される消費電力はどれくらいか、または吸収されたエネルギーでどれくらいの熱が生成されるか、などについて考えるはずです。このような問いには XPower Esitimator が対応します。XPower Estimator は、FPGA ロジックとデバイスがはんだ付けされるプリント回路基板を同時に開発する際に役立ちます。この演習を行うことで予期できるマージンについて理解し、インプリメント後にシステムがバジェット内で動作することを確証できます。

図 3-1 に XPower Estimator のインターフェイスを示します。

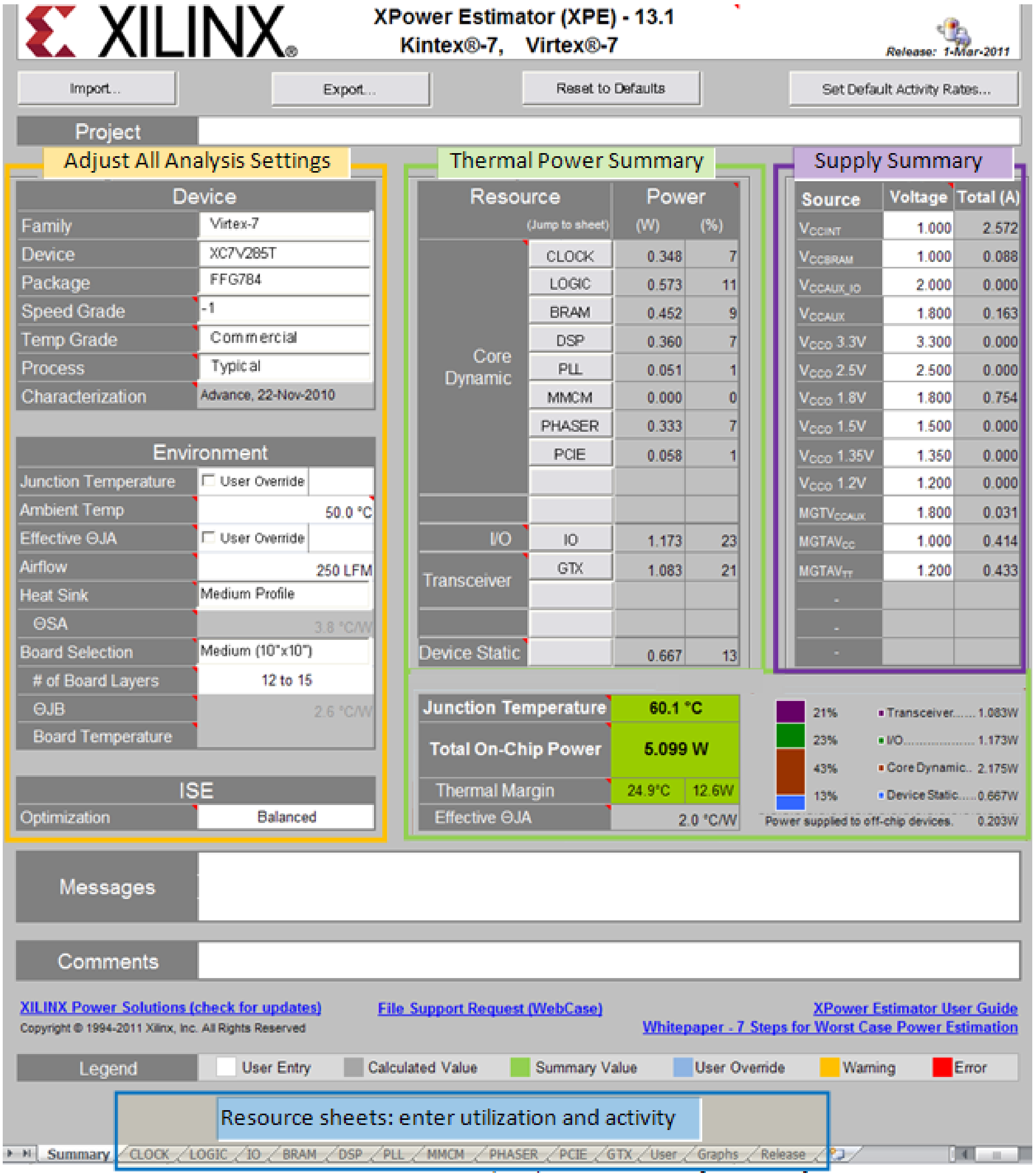

図 3-1 : XPower Estimator (XPE) で示される消費電力情報

手法

この段階では、XPower Estimator スプレッドシートを使用すると、既知の環境およびデザイン情報を確認、整理できます。http://japan.xilinx.com/power にある消費電力ソリューション ページで該当するデバイスのスプレッドシートをダウンロードしてください。

1. XPower Estimator スプレッドシートで、デバイス設定を入力します。

   ドロップダウン リストからデバイスを選択します。

   メモ：[Summary] シートの [Process] フィールドは、製造ばらつきによる典型的またはワースト ケースの消費電力特性を示しています。

2. 環境設定を入力します。

   スプレッドシートを開くと、[Summary] シートに [Environment] パネルがあり、デバイスの消費電力に影響する環境パラメーターをすべて設定できます。これらを変更するとデバイスのスタティック消費電力に影響するだけではなく、FPGA で散逸される電力 (Effective ThetaJA) が増えるごとにジャンクション温度がどれだけ上昇するかが決まります。

   デザイン プロセスでシステムに対して熱シミュレーションを実行可能である場合は、ダウンロード センターから該当するデバイスのパッケージ熱モデルをダウンロードできます (japan.xilinx.com/download で [デバイス モデル] → [パッケージ熱モデル] を表示)。この情報を使用すると、ジャンクションから外気 (Θ_{JA}) の熱抵抗、およびジャンクションからボード (Θ_{JB}) への熱抵抗を算出できます。熱シミュレーションを実行できない場合は、ドロップダウンからシステムに最も近似する値を選択してください。

3. ソフトウェア設定を入力します。

   デザインで最も困難になると思われる側面を選択すると、XPE では使用されるインプリメンテーション アルゴリズムおよび想定される配置配線結果に基づいてダイナミック消費電力計算を調整できます。

4. 電圧設定を入力します。

   この情報が判明している場合は、[Power Supply] の表で電源ごとに正確な電圧値を入力します。電圧は、スタティック消費電力およびダイナミック消費電力の両方に大きく影響します。

5. 使用するデバイス リソース数および予期するアクティビティを入力します。

   デザイン モジュールまたはファンクション ブロックごとに、予期する FPGA リソース使用量および必要なコンフィギュレーションの情報を入手して、これらの平均アクティビティを予測します。最後にこの情報を XPE のさまざまなデバイス リソース シートに入力します。

   デバイスのリソース使用量およびデザイン アクティビティを予測する際のツールの機能およびヒント：

   - デザインにインプリメンテーション済みで以前のデザインから取得または再利用された IP ブロックが含まれている場合は、XPower Analyzer の相互運用ファイルを使用して消費電力情報をインポートします。これにより推測作業を大幅に省くことができるだけでなく、インポートしたデータはデザイン仕様または FPGA アーキテクチャ間の差異に対応できるよういつでも調整できます。
   - 大型モジュールで推測するのではなく、ブロックを細かく分解してからすべてのリソース数を加算します。
   - 現実的なデータを入力し、保守的になりすぎないようにします。ワースト ケースの想定を算出する方法はほかにもあるので、パディング ロジックやアクティブ マージンをモジュールに追加する必要はありません。
   - 以前のデザインでの経験を活かします。

- XPE の User シートは保護されていないため、中間計算や想定を記録するのに非常に役立ちます。

最小限のデータ セット :

- 環境および電圧データは総消費電力に大きく影響するので、予期する実際の環境に合わせたデータを入力します。
- I/O およびトランシーバーの情報も総消費電力に大きく影響するので、それぞれのシートでできる限り情報を入力します。
- それ以外のシートでは、リソース使用量および平均アクティビティを知識に基づいて推測し、不明の列はデフォルト値のままにします。

6. What If? 解析シナリオを実行します。

   コードを記述せずに複数のインプリメンテーションを想定して、リソース使用量および消費電力における影響を再検討できます。たとえば、別の I/O インターフェイス形式を評価し、電力バジェット要件を最も満たす I/O 規格、I/O コンフィギュレーション、および出力終端を判別できます。オンチップ終端を使用すると、外部コンポーネントおよびボード空間を節約できます。また、オンチップ電力が増加するので、デバイスのジャンクション温度を通常の動作範囲内に保持できるように、デバイスからこの電力が環境に放出されるようにする必要があります。

   プロジェクト リーダーはこのような解析を使用して、チーム メンバーにデバイス リソースおよび電力バジェットを配分できます。これらの配分が概算で、後で変更されるとしても、この解析により各開発チームが自立でき、並行して開発しやすくなります。

7. スプレッドシートを定期的に更新します。

   デザインのインプリメンテーションの進行と共により明確な情報が取得できるので、スプレッドシートを更新することで消費電力予測を向上できます。たとえば、デザインの一部分が作成された後に、プロジェクト リーダーがすぐにこのデータを XPE にインポートすることで、デザイン フローの初期に実行された予測と置き換えることができます。通常プロジェクト スコープは新しい機能が依頼されるなどして時間の経過と共に変化するので、デバイス使用量および予期するアクティビティ パターンも変化する可能性があります。スプレッドシートを定期的に更新することで、総熱電力バジェットを上回ってないことを確認し、潜在的な問題を早期に発見できます。

インプリメンテーション中の監視

状況説明

デザイン インプリメンテーションの進行に伴い、消費電力を定期的に監視、確認したり、放熱量がバジェット内に収まっていることを確認することで、制約に近づきすぎているエリアがある場合に早期に発見して対処できるようにしておく必要があります。使用できるツールおよび機能は、次に示すようにロジックの完成度によって異なります。

RTL 記述段階

HDL コードを記述しているときは、合成を実行する前に PlanAhead の RTL 消費電力予測アルゴリズムを使用してリソース仕様量およびダイナミック消費電力を予測できます。プロジェクトは完了している必要はありません。デザインで使用できる部分のみのプロジェクトを作成できます。

図 3-2 に PlanAhead で表示される消費電力情報を示します。

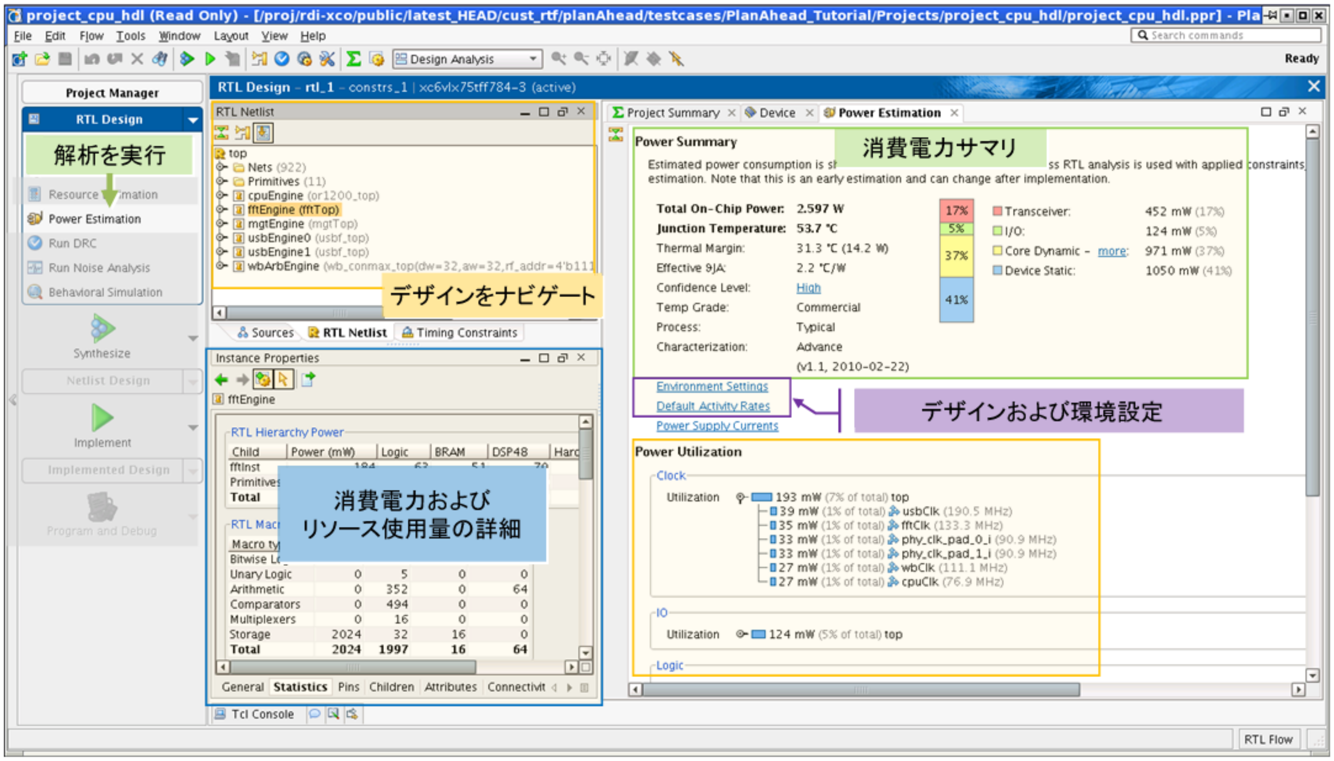

図 3-2 : PlanAhead で示される消費電力情報

手法

1. PlanAhead で RTL ビューを開きます。

   PlanAhead で指定されているすべての HDL ファイルが読み出されます。定数およびパラメーターが分解され、記述がエラボレートされて RTL ネットリストが 1 つ生成され、リソース使用量が予測されます。このプロセスは高速で実行されます。

2. XPower Estimator スプレッドシートに入力したデータと一致するようにデバイスおよび環境情報を設定します。

   これらの情報は、ダイナミック消費電力にはあまり影響しませんが、XPE による現時点での推測と比較する際に役立ちます。特に、RTL 記述が最終段階に近づくほどデザインの HDL コードがほとんどが完了しているので、公正な比較を実行できます。

3. アクティビティ レートを設定します。

   クロックなどの既知のエレメントでは、制約を使用して定義できます。

   不明なエレメントでは、後で変更可能なデフォルト値を使用します。

4. 消費電力予測を実行して、予測されたエリアおよび関連消費電力を確認します。

   エラボレートされたデザインをナビゲートして、デザイン階層内の各エレメントの予測リソース使用量および関連消費電力を確認できます。

5. エリアおよび消費電力の解析後に XPower Estimator またはインプリメンテーション制約を調整します。

   XPower Estimator スプレッドシートで RTL 消費電力予測アルゴリズムで予測されたリソース数および消費電力を確認します。計画した仕様と異なる場合は、必要に応じて変更します。一

方 RTL 消費電力予測ツールでは、特定ブロックの電力分配を確認した後に記述を変更するか、またはダウンストリーム ツールを対応付ける制約を追加する必要がある可能性があります。

これらの予測は、ロジックの合成マップおよび変換や配置配線による最適化が実行されているわけではないので、最終値ではありません。ただし、RTL 消費電力予測とデザイン初期に手動で実行した予測を比較することで、予測を確認したり、値を調整できるので、デザインが電力バジェット内に収まるという確信度が高くなります。

合成段階

デザイン全体またはモジュールごとのいずれの場合でも一度合成されると、配置配線中の最適化により最終的なリソース使用量やアクティビティがわずかに変化することはありますが、リソース使用量はかなり明確になります。合成ツールでは、ターゲット デバイスで使用できるリソース数、消費電力バジェット、およびランタイムなどのその他の制約を満たしながら、パフォーマンス要件を達成しようとします。つまり、ブロックと分散メモリのどちらを使用するか、またはステート マシンで別のエンコーディング スタイルを使用するか、などの決定が下されます。これらの決定は、今まで推測しかできなかったコンフィギュレーションおよびリソース使用量に影響します。XPower Estimator スプレッドシートによる推測とこれらの合成結果を比較し、XPower Estimator の値を必要に応じて変更します。

配置配線段階

状況説明

配置配線が完了した後は、デザインのデータベースに含まれるすべてのロジックのコンフィギュレーション、パッキング、および配線構造が完全に定義されています。XPower Analyzer では最も包括的なレポートが生成されますが、結果を最も効果的にするためには、さらにユーザーが情報を入力する必要があります。次の手法セクションでは、消費電力レポートの生成方法を示し、プロジェクト ファイルには含まれていないが消費電力予測の精度を向上できる情報を指摘します。

手法

このセクションでは、XPower Analyzer (XPA) GUI を使用した消費電力解析について説明します。ここでは、配置配線後に初めて消費電力解析を設定することを想定しています。このため、ツールにアクティビティ情報を入力し、既存のデータ ファイルを参照します。後続の run では、XPower Analyzer GUI でデザインをナビゲートして消費電力を解析するか、または同等のコマンド ライン (xpwr) を使用して GUI をバイパスし、テキスト形式の消費電力レポートを表示するか選択できます。

図 3-3 に XPower Analyzer のインターフェイスを示します。

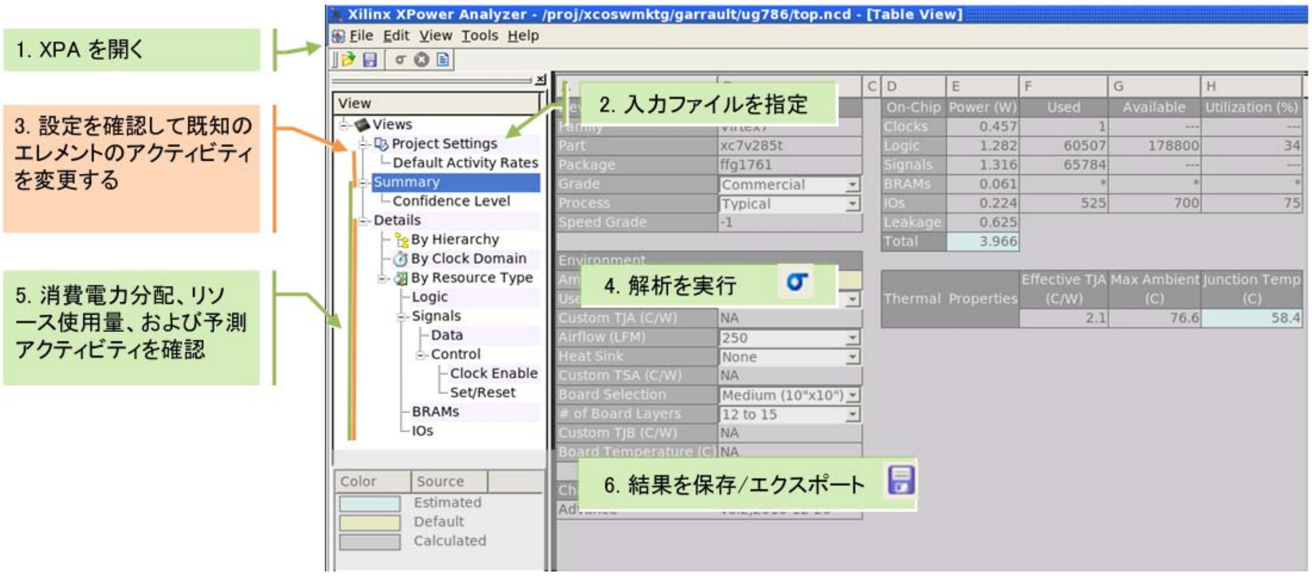

図 3-3 : XPower Analyzer (XPA) で示される消費電力情報

1. XPower Analyzer グラフィカル インターフェイスを開きます。

   XPower Analyzer は Project Navigator またはPlanAhead インターフェイスから起動するか、またはコマンド ラインに「xpa」と入力すると起動できます。

2. 解析に使用する入力ファイルを指定します。

   - 配置配線済みのデザイン データベース (NCD ファイル) : すべてのロジック コンフィギュレーションおよび配線情報が含まれています。
   - 物理的制約 (PCF ファイル) : デザインのすべてのロジックおよび I/O の設定とクロック ネットワークなどの特定ネットのアクティビティが含まれています。
   - シミュレーション結果 (SAIF または VCD ファイル) : XPA ではデザイン データベース内のネットをシミュレーション結果のネットリスト内の名前に一致させます。一致したネットすべてにスイッチング アクティビティとスタティック確立が適用され、デザインの消費電力が算出されます。シミュレーション結果は、合成前や配置配線の前のデザイン フロー早期に生成されている場合があります。この場合、シミュレーション結果からモジュールの I/O ポートのアクティビティのみをキャプチャして、ベクトレス エンジンで内部ノードのアクティビティを予測するようにします。論理シミュレーションでは、グリッチ アクティビティはキャプチャされません。また、一部のロジック はインプリメンテーション中に変換されるので (複製、ゲート化、リタイミングなど)、XPA ではデザインとシミュレーション ネットリスト間で一部のノードを一致できない可能性があります。しかし、ほとんどの主要ポートおよび制御信号が一致するので、この情報がツールに供給されることで一致したノードのアクティビティが現実的になります。アクティビティは、ベクターレス エンジンにより一致しなかったデザイン部分に伝搬されるので、消費電力予測の精度が高くなります。シミュレーション結果を供給するときは、このようなシミュレーション結果を使用するようにしてください。
     - シミュレーションへのテスト　ベクターおよび入力でデザインの典型的な動作または予期される動作が示されることを確認してください。エラー処理およびコーナー ケース (稀にしか発生しないケース) のシミュレーションでは、通常の動作条件でロジックがシミュレーションされません。

- インプリメンテーション後のシミュレーション結果は、ビヘイビアー シミュレーション結果よりも優先されます。

- 設定 (XPA ファイル) : 環境およびアクティビティ データが含まれます。この情報は、XPower Analyzer で早期に生成された可能性があります。この場合は、XPower Analyzer を以前実行したときと同じ条件で結果を開き直すと、簡単にその時点から解析を継続できます。また、このメカニズムを使用して保存、復元、複数の想定の比較もできます。最後に、フロー早期に XPower Estimator スプレッドシートから作成した設定ファイルをインポートでき、すべての環境パラメーターが XPA セッションにインポートされ、手動で再入力する作業を省くことができます。

3. 設定を確認し、既知のエレメントのアクティビティを調整します。

   さまざまな入力フィールドを確認して、予期するシステムが正しく表現されているか確認します。

   - プロジェクト設定 : [Project Settings] ビューで XPA にすべての入力ファイルが読み込まれていることを確認します。シミュレーション結果をインポートする場合は、一致しているレートが予期する数値であるか確認します。階層セパレーターが一致していなかったりフォーマットの差異があると、シミュレーション出力結果からアクティビティが読み出されるエレメント数が減少する可能性があります。
   - 環境設定 : [Summary] ビューで編集可能なセルを確認します。プロセス、電圧、環境データが予期する環境に近似していることを確認します。これらの設定は、予測する総消費電力に大きく影響します。
   - ツールのデフォルト : [Default Activity Rates] でツールの現時点でのデフォルト値を確認し、アプリケーションがこれらの値から大幅にずれないか予測し、調整が必要か判断します。通常、これらの設定は一連の代表的ユーザー デザインに基づいているため、変更しないことを推奨します。これらの値は、GUI または入力ファイルからアクティビティが入力されなかったノードに対して使用されます。次に、伝搬エンジンにより各ノードのアクティビティがロジックの駆動コーンから伝搬されるアクティビティに基づいて変更されます。
   - 既知のエレメント : アプリケーションの動作の情報は、入力ファイルで定義されていないアクティビティを定義する際に役立つので、このステップはデザインのダイナミック消費電力を算出するのに重要です。
     - [By Clock Domain] ビュー : すべてのクロックが指定されていることを確認します。推奨はしませんが、場合によってはデザインの制約を厳しく設定して、インプリメンテーション ツールでの作業を困難にしたり、タイミング マージンを多めに持たせることがあります。消費電力予測では、デザインがボードで実行されるときのクロック周波数を使用する必要があります。使用しないと、デザインのダイナミック消費電力の精度が下がります。
     - [IOs] ビュー : I/O インターフェイスのデータ パターンが判明している場合は、該当する列 ([Signal Rate] および [% High]) に入力します。スプレッドシートなどの個別ツールで電源ごとの総消費電力を算出しない限り、出力の終端方法を指定して、XPA で FPGA の電源からこれらの外部コンポーネントに供給する電力量が含められるようにします。
     - クロック信号ビュー : XPA ではさまざまな制御信号が [Clock Enable] ビューおよび [Set/Reset] ビューに表示されます。アプリケーションで予期される動作と照合したとき、一部のセット/リセット信号が通常のデザイン動作でアクティブではないことがわかる場合があり、この場合はこれらの信号のアクティビティを調整する必要があります。同様に、アプリケーションに含まれる一部の信号では、ブロックが使用されないときにブロック全体がディスエーブルにされる場合があります。予期する機能に合わ

せてアクティビティを調整します。合成ツールおよび配置配線アルゴリズムでは RTL 記述を最適化するように制御信号を推論またはマップし直すことができるので、これらのビューに馴染みのない信号が多くリストされる可能性があります。これらの信号が不明な場合は、ツールによりアクティビティを決定させせます。

4. 解析を実行します。

   ロジックのコンフィギュレーションおよびアクティビティを XPA に入力した後に、解析を実行します。ツールでは、ファイルおよびユーザー入力から供給されたアクティビティを含むネットリストがアノテートされ、残りの未定義のノードにツールのデフォルト値が適用されます。次に、未定義のノードのアクティビティ予測の精度が高くなるよう、デザインの主要入力から主要出力まで初期アクティビティが反復処理により伝搬されます。最後に、デバイスで予期するジャンクション温度よび総消費電力要件を算出するために、使用されるリソースごとにダイナミック消費電力が算出され、これらのリソースでのスイッチング アクティビティで生成されるスタティック消費電力も推論されます。

5. デザインでの電力配分を確認します。

   消費電力解析が完了したら、[Summary] ビューを開いて [Supply Power] 表および [Thermal Properties] 表を確認します。これらの表では、オンチップ消費電力の合計およびデバイス ジャンクション温度が表示されます。これらのセルは、予期するジャンクション温度が通常のデバイスの動作範囲のとき水色になります。ジャンクション温度の予測値がデバイスのグレードの最大値を超えているが絶対定格温度以下のときはオレンジ色になります。ジャンクション温度が絶対定格温度を超える場合、このような動作条件によりデバイスが破損する可能性があるため、赤色になります。

   [Supply Summary] 表では、各電源の電流およびそのスタティック消費電力およびダイナミック消費電力の内訳が表示されます。[On-Chip] 表では、デバイス リソース タイプごとの消費電力が表示されます。この高位ビューを利用して、消費電力が最も高いデザイン箇所を特定できます。

   次に各種 [Details] ビューを開いて、リソースごとの消費電力の詳細を確認します。[Details] ビューは、それぞれ表で表示され、表示できるアイテムが管理しやすいように最大 2000 個までに制限されています。列ヘッダーをドラッグすると、列の順序を変更できます。また、列ヘッダーをクリックすると並び替えの順序を変更できます。レポートされている消費電力が、熱バジェットまたは電源バジェットを超えている場合は、第 4 章「消費電力削減のためのヒントおよび手法」に含まれているデバイスの消費電力を削減する手法の一覧を参照してください。使用できる手法は、デザインの完成度や開発プロセスの変更許容度によって異なります。

6. 結果を保存するかエクスポートします。

   結果に納得し、さまざまなビューでアプリケーションに関係する情報を確認したら、次を実行できます。

   - テキスト ファイルとして結果を保存 : プロジェクトの記録として消費電力予測結果を保存します。または、別のマップ、配置、および配線オプションを試してパフォーマンスまたはエリア制約を決定します。それぞれの条件で消費電力結果を保存しておくと、複数の条件で要件が満たされたときに最も消費電力が低いソリューションを選択するときに役立ちます。
   - 変更した入力を設定ファイルとして保存 : 現在の設定で解析を保存すると、後で結果をそのまま読み込み直せるので便利です。XPA では、ツールに手動で入力された情報すべてを含むファイル (.xpa) が保存されます。このファイルは、アプリケーションの動作条件下でさまざまな環境またはモード/機能を使用して消費電力を予測するようなときにも役立ちます。

- デザインを XPower Estimator で解析できるようエクスポート：環境情報、デバイス使用量、デザイン アクティビティのすべてを 1 つのファイル (.xpe) に保存すると、XPower Estimator スプレッドシートにインポートできます。消費電力バジェットが超えてしまい、ソフトウェアの最適化機能だけではバジェットを満たせないと判断するときにとても便利です。この場合、RTL コードを変更したり、インプリメンテーションを実行し直す前に、現在のインプリメンテーション結果を XPower Estimator にインポートし、さまざまなマップ、ゲート化、畳み込み、およびその他の手法を試して、それらの消費電力への影響を予測します。

消費電力クロージャ段階

デザイン サイクルでは、二つの主要な状況で消費電力クロージャが含まれます。

制約は満たされているがデザインを最適化する必要がある場合

現在のシステムの複雑化やタイム トゥ マーケットの圧力において、この状況は稀です。通常、開発プロセスのこの段階では、RTL、ボード電源、冷却パラメーターの変更は検証に時間がかかったり、または PCB リスピン コストがかかるため、最小限に抑えたいはずです。ただし、この段階でも異なるソフトウェア オプションおよび制約を試すと、ロジックおよび配線リソース数、コンフィギュレーション、およびアクティビティを最適化できます。最適化により、ダイナミック消費電力が最小限に抑えられ、同時にスタティック消費電力も削減されます。デザイン マージンにもよりますが、重要なダイナミック消費電力で 15% ～ 20% は節約でき、一部のデザインではそれよ以上削減できます。

ISE の SmartXplorer 機能または PlanAhead の [Design Runs] ビューを使用すると、この操作を簡単に実行できます。これらには、合成および配置配線ツールの設定を調整する定義済みのストラテジ セットがあります。また、既存のストラテジを変更したり、独自のストラテジを作成できます。満足の行くストラテジが用意できたら、これらを使用してインプリメンテーション ソリューションを試します。この際、1 つのマシンで実行するか、ランタイムを抑えるために複数のマシンを使用して実行します。正しく完了した run の中から最も消費電力を抑えるオプションを選択し、最終アプリケーションのネットリストに使用します。

消費電力バジェットが超えている場合

通常、この段階では、システムを市場にリリースする圧力は高くなり、ボード環境および冷却オプションなどのシステムに含まれる多くのパラメーターは詳細に定義されています。これにより、エンジニアリング作業のやり直しが制限されているとしても、次の手法を使用すると、消費電力を削減できる可能性が高いエリアを特定できます。

手順 1：消費電力バジェットが超えている箇所の確認

GUI を使用している場合は、XPower Analyzer の [Summary] ビューで、コマンド ラインを使用している場合は xpwr コマンド レポート ファイル (.pwr) の Summary セクションで確認できます。[On-Chip] および [Supply Power] の表では、高位の消費電力配分が確認できます。[Summary] ビューで消費電力バジェットを超えている消費電力の種類および電力量を確認します。

手順 2：フォーカスするエリアの特定

XPower Analyzer または XPower Estimator に含まれるさまざまなビューを確認します。環境パラメーター、各リソースで消費される電力、デザイン階層、およびクロック ドメインを解析します。消費電力が高いエリアを見つけた場合は、次の情報を元にその原因を特定できます。

熱バジェットが超えている場合

予測されたジャンクション温度がデバイス指定の動作条件を超えている場合は、デバイスへの電力量を減らして熱として放散するか、熱を除去しやすくするようにシステムの熱機能を向上させる必要があります。ダイナミック消費電力とスタティック消費電力間の配分率を確認します。

スタティック消費電力の削減

- V_{CCINT} または V_{CCAUX} などの電源の消費電力は、プロセス、電圧、および温度に大きく影響を受けます。デバイスのスタティック消費電力の大部分に寄与するデザイン リソースは、トランシーバー、I/O、およびクロック生成モジュールです。
  - トランシーバー : パワー ダウン モードおよび低消費電力モードの仕様を検討してください。トランスミッターの電圧幅を減らすよう検討してください。PLL など複数のチャネル間で共有する回路のサポートを最大限にしてください。
  - I/O : I/O は比較的遠い距離にある信号を駆動、受信する必要があるため、トランジスタはコアのトランジスタと比べて大きくなるため、使用されるユニット リソースごとの消費電力が大きくなります。
    - V_{CCAUX} : 通常入力バッファーに供給されます。使用する I/O 規格を確認します。一部のデバイス ファミリではこの電源に異なる電圧レベルを使用できるので、低めの電圧が使用できるか評価します。
    - V_{CCO} : 主に出力バッファーに供給されます。パフォーマンス要件に照らして I/O 規格、駆動強度、およびオンチップ終端設定を確認し、トライステートが可能な DCI I/O 規格 (T_DCI) を使用して駆動強度を下げたり、終端の使用を省いたり、外部終端を使用したりできないか、を評価します。
    - V_{CCINT} : デバイス コア ロジックがある I/O インターフェイスに供給されます。アプリケーションで必要な最小限の I/O 機能のみをイネーブルにします。データ レートで許容されるときは、一部の機能で低電力モードを使用することも考慮します。
    - クロック生成モジュール : 通常これらのモジュールのコンフィギュレーションの電力は、V_{ccaux} 電源から供給され、デバイスのコア ロジックがあるインターフェイスに供給される電力よりも多くの電力が消費されます。使用するクロック生成モジュールの数を最小限に抑えるようにします。ほとんどのブロックにはプログラム可能な出力、周波数、および位相シフトがあるため、同じモジュールを使用して関連しない複数の IP ブロックのクロック信号を生成できることがよくあります。任意のクロック レートを生成するために乗算および除算係数を選択するときに、VCO 周波数を最小限に抑えるようにします。
    - パーシャル リコンフィギュレーション : 環境に依存して異なる機能を持つ複数のアプリケーションが デザインに含まれる場合、パーシャル リコンフィギュレーションを使用して各環境に該当する機能のみを使用してデバイスをプログラムすることを検討してください。パーシャル リコンフィギュレーションを実行すると、ロジックおよび配線リソースを節約できるので、小さいサイズのパーツを使用できる可能性があります。
- デバイス環境
  - 電圧 : 標準値の電源範囲では、デバイスが予期どおりに動作します。これらのレールに接続されている電源レギュレーターおよびその他のコンポーネントで電圧レベルを下げることができる場合は、1 ～ 2 % 下げるだけでもトランジスタのリーク電力およびスイッチング電力に大幅に影響します。電圧レベルは、スタティック消費電力とダイナミック消費電力の両方に影響します。
  - 放熱パス : 生成された熱がデバイスから放散するパスは、主に 2 つあります。熱は、パッケージを通って空中に放散されるか (上方向)、パッケージ ボールおよびボードを通って空

中に放散されます (下方向)。デバイス ダイと周囲環境間で各移動係数を確認します。これらの熱抵抗を下げると、デバイスで生成された熱が多くかつ迅速に環境に放散され、デバイスのジャンクション温度が下がりますが、その代わりにトランジスタのスタティック消費電力が下がります。周囲温度を減らすか、システムのエアフローを増やすか、または FPGA のファンが使用できないか、などを評価します。また、ヒート シンクを追加したり、既存のヒート シンクの特性を変更することも検討できます。

ダイナミック消費電力の削減

デザインのダイナミック消費電力に寄与する係数 : $\sum (\alpha . f_{clk} . C_L . V^2)$

次に、上記の式に含まれる係数について説明します。

- アクティビティ (α, f_{clk})

  コンポーネントと信号のトグル数が少ないほど、消費電力も少なくなります。通常、I/O、クロック、演算器、メモリ、および一部のビット単位のロジックは、デザインで最もアクティビティが多くなるデザイン箇所です。さまざまなオプションを使用して、このアクティビティを減らすことができます。

  - ユーザーによる介入 : デザインの動作を理解することで、デザインに含まれているモジュールの出力が使用されていないときにこれらのモジュールの電源を切ることができます。新しいデータがキャプチャされないように入力をディスエーブルにできます。また、クロック イネーブル信号を追加して、デザイン ブロック全体、I/O インターフェイス、またはクロック ドメインにゲートを付けることも可能です。クロック周波数を減らすことは稀ですが、一部のアプリケーションでは入力データの有無によってクロック周波数を調整することも可能です。
  - ISE 消費電力最適化アルゴリズム : 階層の境界に関係なく、デザインのデーター フローで出力が使用されていないシーケンスを厳密に検出し、未使用のサイクルでクロックとロジックの両方またはいずれかがゲート処理されます。たとえば、セレクター値を予測することで未使用の乗算器入力をディスエーブルにしたり、読み取りまたは書き込み操作が不要なときに RAM ポートをディスエーブルにします。

- キャパシタンス (C_L)

  トグル イベントごとに駆動される必要があるキャパシタンスは、ロジック タイプ、ファンアウト、およびキャパシタンス、およびデザインで使用される配線リソースによって異なります。まず、デザインの制約を確認します。厳しく設定されすぎていない効率の高い制約では、インプリメンテーション ツールでタイミング クリティカルなパスのみが最適化され、ほかのパスで使用されるエリアおよび配線構造が最小限に抑えられます。

  - 信号配線 : どの配線リソースを使用するかは、各パスおよびその周囲ロジックで考慮する事項が多くあるため、配置配線ツールを使用して決定するのが最善策です。ただし、フロアプラン手法を使用して特定のクロック領域にまとめたり、インターコネクトの度合いが高いロジックを近くに配置したりすることができます。これにより、デザインに含まれるファンアウトの大きい信号やアクティブの度合いが大きいパスの長さを短縮できます。

- 電圧 (V^2)

  電圧は、主要な外部パラメーターでダイナミック消費電力に影響します。電圧 : 標準値の電源範囲では、デバイスが予期どおりに動作します。これらのレールに接続されている電源レギュレーターおよびその他のコンポーネントで電圧レベルを下げることができる場合、スタティック消費電力およびダイナミック消費電力の両方に影響するので、1 ～ 2 % 下げるだけでも FPGA の総消費電力に大幅に影響します。

- エレメント数 (∑)

  エレメント数を減らすと、切り替える総キャパシタンスが減ります。このためには、ターゲットおよびツール制約に対して HDL コードを効率的に記述する必要があります。グルー ロジックの一部を DSP やブロック RAM などのハード ブロックを使用するように変更するのも効果的です。シフト レジスタは特殊な LUT にインプリメントして、使用するレジスタ数および配線構造数を減らすことができることを覚えておいてください。時分割多重やパーシャル リコンフィギュレーションなどのエリアを減らすその他の手法も試すことも可能です。

- バランスの取れたアプローチ

  ほとんどの場合で、解析後にスタティックとダイナミックの両面で変更する方法が、消費電力を削減して許容バジェット内に抑える最も簡単な方法です。

電源バジェットが超えている場合

一部のアプリケーションまたは規格でプリント回路基板 1 つが消費できる最大電力が定義されているために、総消費電力バジェットを超えてしまう場合があります。また、選択した電圧レギュレーターで供給可能な最大電流にデザインが達してしまっているような場合もあります。FPGA はプログラム可能で、システム内に含まれるコンポーネントの中で最も制御できるので、FPGA の消費電力を最小限に抑えることができないかを確認するのは当然です。総消費電力には、熱電力に加えて FPGA を介して電源から供給される電力およびコンポーネントの外側に放散される電力も含まれます。内部消費電力を減らす手法は、上記の「熱バジェットが超えている場合」を参照してください。また、オフチップ電力には、FPGA が駆動している外部コンポーネントが主に寄与します。これらには通常、抵抗終端の負荷や特殊デバイス (パワー トランジスタ、LED、またはほかのコンポーネントなど) が挙げられます。オンチップ終端をオフチップ終端の代わりに使用すると、I/O インターフェイスの総消費電力の差を算出できます。FPGA では、オンチップ終端を使用しないときにディスエーブルにできます。その一方で、オンチップ終端を使用することで、デバイスで消費される電力が増え、その結果スタティック消費電力が増えてしまいます。各想定で総消費電力を確認し、アプリケーションに最適なトポロジを選択します。

手順 3 : 試行

上記の手順で特定した消費電力を最適化するデザイン箇所の候補リストを確認し、簡単なものから順に並び替えて、実行する最適化または試行を決定します。消費電力ツールでは、What If? 解析を実行できるので、デザイン変更を迅速に入力でき、コードや制約を実際に変更したりインプリメンテーションに戻らずに消費電力を予測できます。

- インプリメンテーション ツール内での試行

  合成およびインプリメンテーション ツールのオプションを見直し、デザイン全体または一部で消費電力およびエリアの削減をオンにします。ザイリンクス ツールでは、消費電力最適化アルゴリズムによりデザインのロジックが自動的にゲート処理されて、コアのダイナミック消費電力を約 15 % ～ 20 % も節約できます。この際、コードを変更したり論理検証を行う必要はありません。

- XPA 内での試行

  XPA では、変更してから解析に戻って消費電力への影響を確認できます。

  - [Environment] : 熱パラメーター、プロセス、または電圧が含まれています。

  - [Design Activity] : デザインに含まれるネットまたはインスタンスのアクティビティを調整します。1 つまたは複数のアイテムを同時に変更します。次も変更できます。
    - クロック ドメイン : スイッチング周波数を調整します。
    - グルー ロジック : ダイナミック アクティビティ レートを調整します。
    - I/O : スタティック アクティビティおよびダイナミック アクティビティの確立を調整します。負荷容量や近端ボード終端など、デバイス出力に接続されている外部コンポーネントのパラメーターも調整できます。
    - 信号 : データ信号のダイナミック アクティビティ レートを調整します。制御信号では、スタティック確立を変更してクロック イネーブル、セット、またはリセットの複数の想定下で消費電力を評価できます。
    - 特定ブロック : ダイナミック アクティビティ確立に加えて、ブロック RAM のポート イネーブルまたはライト イネーブルなどの制御信号のアクティビティも調整できます。
- XPE 内での試行

  複数のソースを使用して開発されている IP ブロックがデバイスでインプリメントされた後には、XPE に XPower Analyzer の結果をインポートして、これらの消費電力を確認できます。また、ネットリストを変更しなくてはいけない状況を評価し、実際にコードを変更しない場合の消費電力への影響を評価することもできます。XPE ではロジック エレメントまたは信号を個々に変更できないので、デザインのコア ロジックに対する作業の精度は XPA よりも低くなります。

  XPE では、次も試行できます。

  - リソース使用量 : リソース数を減らすことができるか試します。ロジックの一部をスライス ロジックから ブロック RAM や DSP などの専用ブロックにマップし直したり、またその逆を実行してみます。
  - リソースのコンフィギュレーション : デザインの I/O、ブロック RAM、クロック ジェネレーター、およびその他のリソースに別のコンフィギュレーション設定を使用してみます。
- PlanAhead RTL Power Estimator での試行

  RTL Power Estimator には、RTL コードを変更して消費電力を減らす必要があるときに、リソースごとまたはデザイン階層ごとに消費電力を確認したり、デバイスの消費電力に最も寄与するデザイン箇所を特定する機能があります。最初の解析から、合成または配置配線に導くデザイン制約およびツール オプションを簡単に引き出すことができます。たとえば、特定のステートマシンに別の方法でマップする指示や最も効率的な消費電力最適化オプションがあります。パイプラインを追加したり、キャリー チェーンや XOR ファンクションなどのアクティビティの高いロジックでの消費電力のリタイミングを実行してみたりすることが可能です。キャリー チェーンを持つ長いパスでは、クロック ドメインが遅くなる傾向がありますが、グリッチ アクティビティが増えてデザインの消費電力が増えてしまいます。これらのパスをリタイミングやパイプライン処理すると、有益になることがよくあります。

手順 4 : 変更を反映し、消費電力での節約を確認

時間、パフォーマンス、およびリソース制約での最適な変更を決定したら、これらを反映させます。一度に試すオプションや変更が多すぎると、競合や相互作用が発生する可能性があるため、結果が最適にならない可能性があります。時間がある場合は一度に試すオプションを限定して、ほかの変更を加える前に消費電力およびその他の制約への影響を評価する方法が最適です。

消費電力および温度の計測

このセクションでは、FPGA の消費電力または放熱量を計測するさまざまな方法について簡単に説明します。これらの方法の一部では、内部 FPGA リソースが使用され、ほかの方法ではボードのコンポーネントまたは外部コンポーネントが使用されます。消費電力と温度をアクティブに監視して開発後に変更する必要があるアプリケーションもあれば、プロトタイプおよび検証段階にラボでこれらの計測手法を使用するアプリケーションもあります。デザインに最も該当する点を考慮してください。

消費電力の計測

- 電流検出抵抗器 : レギュレーター出力と FPGA 間に直列で挿入されます。この抵抗を付けると、電圧が多少下がります。この量は、オームの法則で電流に比例しています。この電圧を計測すると、FPGA に供給されている電流が判明します。
- アドバンス レギュレーターおよびデジタル パワー コントローラー : 最新の評価キットには、アドバンス レギュレーターおよびパワー コントローラーが含まれており、これらを使用してレギュレーター出力の電流および電圧をキャプチャし、その情報を USB インターフェイスを介して監視しているコンピューターに送信できます。この方法が電力レールを監視する最も簡単で便利な方法です。ML605 および SP605 ボードには、Texas Instrument UCD92xx コントローラーが搭載されており、Fusions Digital Power Designer ソフトウェアから PMBus (I2C) - USB インターフェイス モジュールを介してアクセスできます。
- オンボード モニタリング : 最新のザイリンクス デバイス ファミリでは、内部センサーおよび最低 1 つのアナログ ー デジタル変換器があり、供給電圧およびデバイスの温度を計測できます。ChipScope ユーティリティでは、リアルタイムの JTAG アクセスが提供されており、デバイスのコンフィギュレーションの実行前と実行後にさまざまな電源電圧やデバイスのジャンクション温度を計測できます (図 3-4 を参照)。また、システム モニターまたは XADC コンポーネントをコードにインスタンシエートして、FPGA アプリケーションからこれらを計測することもできます。

熱の計測

- 外部モニタリング : デバイスのパッケージによりシリコンにアクセスできないため、ジャンクション温度は直接計測できません。しかし、ジャンクション温度はパッケージ、ヒート シンク、およびそのほかの熱伝対がある場所の温度を計測することで予測できます。また、熱カメラを使用してデバイスの温度および近隣コンポーネントおよびそれより大きい環境と相互作用する放熱を視覚化できます。
- オンボード モニタリング : 熱計測が可能で消費電力の計測と同じ方法が使用されます。コンフィギュレーションの実行前および実行後に ChipScope を使用 (図 3-4 を参照) するか、またはシステム モニター プリミティブをデザインに含めてデバイスのジャンクション温度を読み込みます。

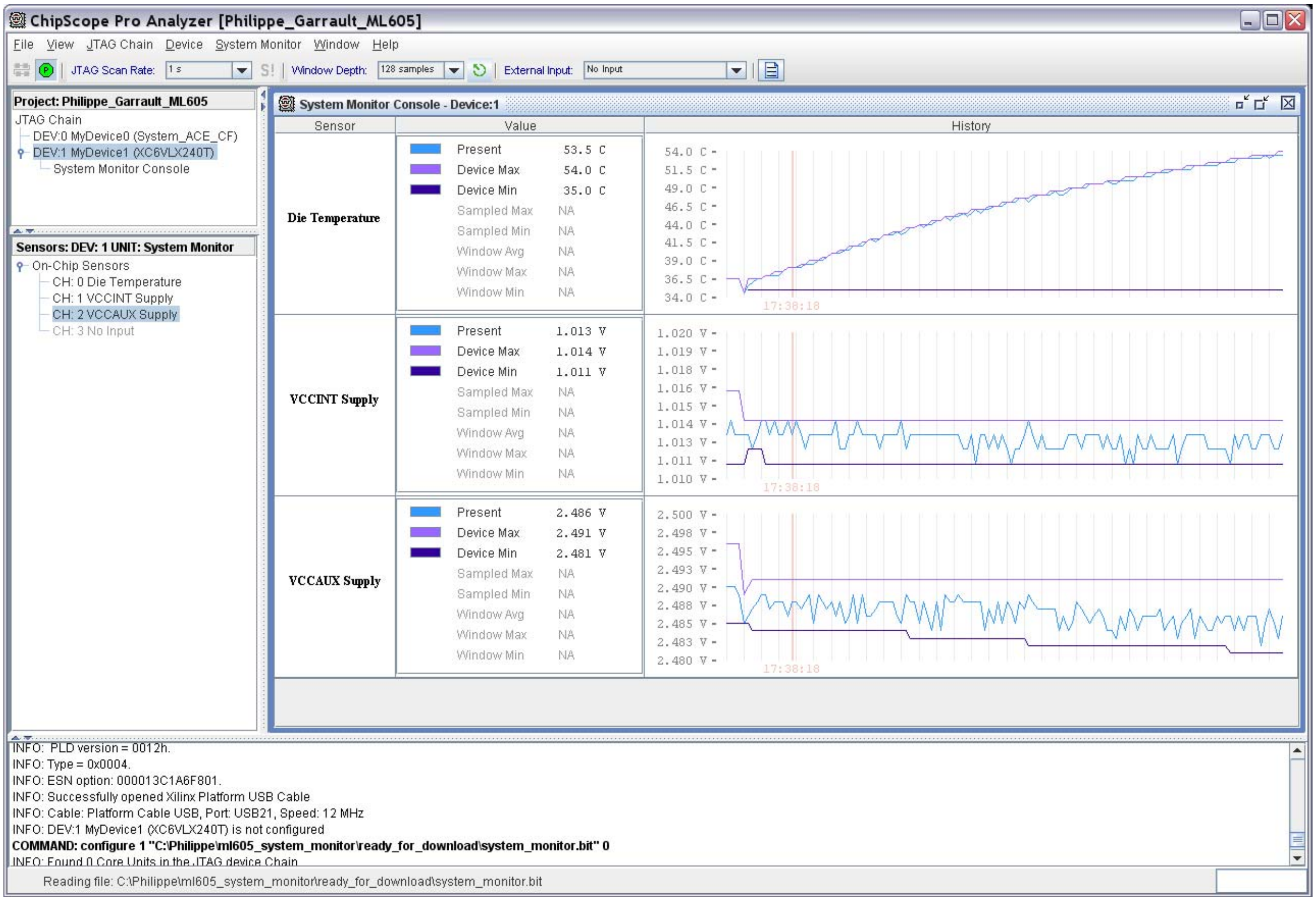

図 3-4 : ChipScope を使用した電圧およびジャンクション温度の監視

消費電力および温度の計測方法

デザインの総消費電力を構成する次の 3 つのコンポーネントを評価できるよう、計測前にデバイスのジャンクション温度を制御して安定化させる必要があります。これは、デバイスおよびデザインのスタティック消費電力がデバイスのジャンクション温度に大きく依存しているためです。

- デバイスのスタティック消費電力 : 最初に空のデザインをダウンロードして、ノイズがキャプチャされておらず、すべての内部ロジックおよびコンフィギュレーション回路が既知のステートになっていることを確認します。空のデザインにはゲートまたはフリップフロップが 1 個含まれており、いずれもトグルせずにすべての出力がトライステートになっています。ジャンクション温度が安定化するのを待ってから V_{CCINT}、V_{CCAUX}、およびその他の任意の電源を計測します。特殊な装置、単純なヒート ガン、または冷却スプレーを使用しても温度を変更でき、デバイスのスタティック消費電力への環境による影響を評価できます。
- デザインのスタティック消費電力 : FPGA にデザインをダウンロードして、すべての入力および内部アクティビティ (入力データ、外部および内部クロック生成) をオフにします。デバイスの温度が安定するまで待ってから、任意の電源レールの消費電力を計測します。これらの値からデバイスのスタティック消費電力を差し引くと、デザインで使用される特定のロジック リソースおよびコンフィギュレーションで消費されるスタティック消費電力 (デザインのスタティック消費電力) がわかります。
- デザインのダイナミック消費電力 : FPGA にデザインをダウンロードして、デザインのクロックおよび入力スティミュラスを供給します。ジャンクション温度が安定するまで待ってから、

任意の電源の消費電力を計測します。この消費電力は、デザインの瞬間的な総消費電力を表します。値は、各クロック サイクルのアクティビティによって変化します。

消費電力削減のためのヒントおよび手法

この章では、消費電力を削減する手法およびその手法で予期される総消費電力への影響を説明します。この情報は、時間、消費電力バジェット、使用可能リソース、およびデザインの変更許容度などに合わせて最適なオプションを評価する際に役立ちます。一部の手法は、内容の重複を避けるため、ここでは詳しく説明していません。詳細は、前章の「試行」のセクションを参照してください。

システム レベル

冷却ストラテジ

冷却ストラテジでは、デバイスで生成された熱が除去されて環境に吸収されるようにします。これらの冷却ストラテジは、デバイスのスタティック消費電力に大きく影響します。このストラテジは、通常デザイン開始時に使用でき、フロー後半では実行しずらくなります。

- エアフローを増やします。
- 周囲温度を下げます。
- ヒート シンク (さらに大きいヒート シンク) を使用するか、または別のレギュレーターを選択します。

電源ストラテジ

電圧は、スタティック消費電力およびダイナミック消費電力の両方に大きく影響します。電圧レベルのアクティブ制御により、指定の電圧がデバイスに使用されます。

- スイッチング レギュレーターを使用 : リニア レギュレーターと比べると消費電力が低くなりますが、コンポーネント数は多くなります。
- 調整可能なレギュレーターを使用 : 同じ電源で複数の FPGA に電力が供給される場合は、FPGA および消費電力が最も消費されるデバイスの電圧値にできる限り近い値になるよう調整します。
- 許容誤差が厳しいレギュレーターを選択します。

デバイスの選択

- 製品に最適なデバイスを選択 : ベンダー、集積度、機能、およびパフォーマンスは、デバイスを選択する際の主要な要素ではありません。機能をインプリメントする際にシステム レベルで決断することにより、全製品の消費電力を最小限に抑えることができます。
- デバイス数を最小限に制限 : 空間を節約し、I/O インターコネクトの消費電力、総リーク電力、およびその他の要素を抑えることができます。通常は、プロセッサと FPGA などの複数のコン

ポーネントをサイズが大きめの 1 つの FPGA と置き換えることで、スタティック消費電力を削減できます。

- 最小デバイスを選択 : リーク電力を削減できます。通常、1 つの FPGA ファミリでは異なるダイ サイズを含む同じパッケージがあります。たとえば、プロトタイプ中や生産前には大きめのダイを使用し、量産段階では小さめのダイを使用できます。
- 最大パッケージを選択 : 放熱量を増やすことができます。パッケージが大きいほど、ダイの熱を環境に放散するエリアが大きくなります。パッケージの上面に装着するヒート シンクを大きくすると、下面のボール グリッド アレイを介してプリント回路基板に放散される熱を増やすことができます。
- 低電圧デバイスを使用 : 一部のデバイス ファミリでは低電力オプションを使用できます。電圧要件を下げると、スタティック消費電力およびダイナミック消費電力を大幅に節約できます。
- リーク電力が低いデバイスを使用 : 一部のデバイス ファミリには、特定のスピード グレードまたは温度グレードを使用すると、低リーク電力または低スタティック消費電力オプションを利用できます。これらのデバイスの価格は多少高くなりますが、電気代、冷却ハードウェア、およびシステム管理費をその価格以上に節約できる可能性があります。

デバイス レベル

ザイリンクス設計チームは、デバイスの機能を向上し、お客様のアプリケーションにおける課題を満たすよう、革新的ではありながらも合理的なソリューションを提供することを目指しています。実際面では、製造プロセスおよび構造パラメーターといった 2 つの主要点において広範囲の研究および試験を行っています。これらのトピックは、第 1 章の「デバイスの消費電力の要因」を参照してください。

デザイン レベル

正確な消費電力情報を取得

消費電力を最小限に抑える箇所を特定できるよう、ツールによる消費電力予測が現実的になるようにします。

- 予期するデバイス動作条件を指定 : デバイス、その熱仕様、および適用される電圧は、デバイスの電源要件および熱要件を決定する際の重要な要素です。
- リソース使用量、コンフィギュレーション、およびアクティビティを指定
  - リソース使用量およびクロック/制御信号/主要入力のアクティビティの情報を可能な限りツールに供給します。
  - 以前のデザインのデータや情報を再利用したり、ツールのインポート機能を使用して、手動によるデータ入力を最小限に抑えます。
  - アクティビティを指定するときは、通常動作またはワーストケース動作に一致するようにしてください。デザインでデータがバースト処理され、その後に静止期間がある場合、長期間でアクティビティが正規化するようにします。熱および電源の影響は、内部スイッチング ロジックに比べてかなり時間がかかります。

リソースを効果的に使用

ロジック : 忙しいためにターゲット アーキテクチャを完全に理解できていないことがあります。アーキテクチャを理解すると、次を実行する際に役立ちます。

- デザイン記述を最適化
  - 合成ツールによりロジックを専用ブロックにマップできるようなコードを記述します。合成ツールでは、タイミング要件とリソース使用率要件を満たす最善のマップ方法を決定できます。たとえば、カウンターまたはステート マシンは分散ロジックまたはブロック RAM のいずれかにマップできます。シフト レジスタは、特定の LUT モードにマップしてエリアおよび消費電力を節約できます。
  - ロジックの最適化の妨げになり、配置配線リソースを多く使用する非同期の制御信号を最小限に抑えます。
  - 制御セット数を最小限に抑えます。制御セットは、クロック、クロック イネーブル、セット、リセット、ライト イネーブル (LUT RAM の場合) 信号の固有グループで構成されています。1 つのスライス内に配置されるレジスタ数は制御セット数に影響を受けます。これは、すべてのレジスタでクロック、セット/リセット、およびクロック イネーブル信号が共有されるためです。制御セット数が増えるほどデザインが拡散して消費電力が高くなります。FPGA アーキテクチャによって異なりますが、限度に達したときに近接した関連ロジックをパックできず、配線リソースが増加する場合があります。
  - パイプライン段を追加して、組み合わせロジック コーンのサイズを最小限に抑えます。これにより、各クロック サイクルで信号が最終ステートに到達するまで、レジスタ間のグリッチの伝搬を最小限に抑えることができます。
  - リソースのタイム シェアリングを使用します。この手法により、同じハードウェア リソースに異なるファンクションを時分割多重化することでデバイスのリソース使用量を最小限に抑えることができます。これにより、小さいデバイスを使用できるようになったり、配置配線の混雑を緩和してスタティック消費電力およびダイナミック消費電力を削減できます。
    - 低速で類似しているプロセスは、別のリソースを使用せずに同じリソースで実行できます。ただし、処理するデータのバッファ処理、マルチプレクサ処理、初期化、および制御方法を慎重に検討する必要があります。複数の入力センサーを処理するなど、並列処理が行われるアプリケーションでこのような最適化を実行します。処理ユニットを入力と同じ数にする代わりに 1 つの処理ユニットのみを使用して高速に実行できます。入力チャネルは順に処理され、各出力に対する応答時間は同じになります。XPower Estimator の What If? 予測を実行すると、この作業により消費電力を節約できるかが判断できます。
    - パーシャル リコンフィギュレーションこの手法を使用すると、機能をオンザフライで変更でき、全体をコンフィギュレーションし直してリンクを確立し直す必要がなくなります。パーシャル リコンフィギュレーションは、リソース使用量を最小限にするときに特に役立ち、これによりスタティック消費電力およびダイナミック消費電力の両方が削減されます。システムの現在の環境で必要なファンクションのみ、またはアプリケーションの全プロセスのうちの特定期間で必要なファンクションのみ FPGA に読み込むことができます。詳細は、次のウェブページから入手可能なパーシャル リコンフィギュレーションの手法ガイドを参照してください。

      http://japan.xilinx.com/tools/partial-reconfiguration.htm
    - DSP およびブロック RAM のオプションのレジスタを使用します。たとえば、DSP ブロックで乗算器または MREG レジスタをイネーブルにすると、クロック サイクル間の内部グリッチが最小限に抑えられて伝搬されるので、消費電力が最も抑えられたインプリメンテーションになります。
- リソース数を最小限に抑える：ロジック リソース数を最小限に抑えると、配線リソース数も削減されます。これにより、インプリメンテーション ソフトウェアでデザインをさらに効果的に配置配線できるようになります。

- デバッグ ロジックをディスエーブルにするのではなく、最終的なデザインから削除します。
- クロック生成コンポーネントの数を最小限にします。主要 IP ブロックではなくデザイン最上位の視点からクロック生成および管理を考えることで、不必要な複製や機能を回避します。クロック コンポーネントには複数の出力があることがよくあり、各出力にはそれぞれプログラム可能な周波数および位相シフト機能があります。プロジェクト リードは、2 つの IP ブロックに 1 つのクロック マネージャーをインスタンシエートする代わりに、1 つのクロック ジェネレーターから異なるクロック ドメインを生成できないか、評価できます。

- アクティビティを最小限に抑える : コンポーネントおよび信号配線のアクティビティは、デバイス ダイナミック消費電力に大きく影響します。
    - クロックまたはデータ パスにゲートを付けます。このよくある手法では、これらのパスが使用されないときにパスが停止されます。クロックにゲートを付けると、駆動されているすべての同期ロードが停止され、データ パスにゲートを付けると、スイッチする信号およびグリッチが発生する信号が次の同期エレメントまで伝搬されません。ソフトウェア ツールでは、記述およびネットリストが解析されてこのような状況が検出されます。それでも、ツールにはないアプリケーション、データ フロー、および依存性に関する情報があり、設計者のみが入力できます。

      ゲーティングの一般的なガイドライン :

        - ゲート付き信号の影響を受けるエレメント数を最大限にします。たとえば、クロック イネーブル信号と共に各ロードをゲートするよりも、その駆動ソースのクロック ドメインをゲートした方が消費電力の節約量が多くなります。
        - アクティビティまたはクロック　ツリー使用率を最小限に抑えるようクロック　ゲーティングを実行したり、またはクロックにマルチプレクサを付ける場合は、専用クロック バッファーのクロック イネーブル ポートを使用します。LUT を挿入したり、その他の方法でクロック信号をゲートするのは、消費電力とタイミングを考慮する場合、効率がよくありません。
        - 制御セット数を最小限に抑えます。ゲート付き信号を追加してデータまたはクロック パスを停止するには、ロジックと配線が必要になるので、本来の目的を達成できるよう、制御セット数を最小限にする必要があります。これらの余分リソースを配置配線すると、既存ロジックのインプリメンテーションを悪化させる可能性があります。配置が広がったり、複製が作成されたり、または配線が混雑する可能性があり、この結果ダイナミック消費電力が増加します。
        - ブロック RAM ポートをディスエーブルにします。アプリケーションでアレイからの読み出しまたはアレイへの書き込みが行われないときは、使用されていないメモリ ポートをディスエーブルにするようにイネーブル信号を記述します。
    - サスペンド　モードまたはスタンドバイ　モードを使用して、デバイスを使用されないときにディスエーブルにします。機能は、FPGA ファミリによって異なります。この手法は、バッテリー アプリケーションや処理するデータがバースト処理されてその後長期間アクティビティがないようなときによく使用されます。このメカニズムは簡単にインプリメントでき、デバイスのスタティック消費電力およびダイナミック消費電力を削減できます。デバイスがパワー ダウン モードから回復するときに、アプリケーションで起動時間が許容されるようにしてください。
    - アーキテクチャ コンポーネントそれぞれにパワー ダウン モードを使用します。デバイス コンポーネントのほとんどには、回路の電源をオフにしたり、クロックまたはデータ フローをディスエーブルにする機能があります (例 : ブロック RAM ポート イネーブルおよ

びトランシーバーの各種パワー ダウン モード)。アーキテクチャについて学び、専用の消費電力節約構造を最大限に利用します。

- 配置をガイド : デザイン ロジックの配置をソフトウェアに制御させるのが、通常は最善策です。小さいデバイス エリアにロジックを制約すると、配線リソースが短くて済むのでダイナミック消費電力が節約されると考えるでしょうが、逆に混雑を招き、この人工的に発生した混雑を回避するために最適ではないリソースが信号で使用される可能性があります。しかし、次のような場合には、配置をガイドすることが役立ちます。
    - タイミング制約またはその他の制約が緩く設定されているときにアクティビティが高いロジックを高密度で配置します。XOR などの演算器およびビット単位のファンクションでは、クロック サイクル間にグリッチが発生する可能性があり、ダイナミック消費電力が増えます。これらのファンクションを互いに近くに配置すると、配線リソースが短くなり、同じスライスまたは CLB にパスを含めることができ、消費電力が抑えられます。

I/O : I/O インターフェイスでは、寄生要素による影響を受ける可能性がある長距離を駆動する必要があるため、通常はデバイスの消費電力要件の大部分を占めます。

- 最小限の V_{CCAUX} を使用します。これにより、この電源のスタティック消費電力およびダイナミック消費電力の両方が最小限に抑えられます。
- 入力 : 内部で参照される入力規格の使用を制限します。
- 出力 :
    - 受信チップでサポートされるスルー /駆動強度/電圧の最小レベルを使用します。
    - 終端または直列終端ではなくパラレル終端を選択します。この決定には、シグナル インテグリティ シミュレーション ツールを使用できます。
    - デバイスの熱バジェット、システム コスト、およびボードのスペース要件を考慮して、オンチップ終端またはオフチップ終端のどちらを使用したらよいかを検討します。
    - 電圧幅の低い差動規格を使用できないか検討します。
    - アプリケーションで大型パラレル　バスの変わりにトランシーバーを使用できないか検討します。
    - IBUI、IO DELAY などの I/O 機能の要件を評価し、許容される場合はディスエーブルにします。

トランシーバー

- 低電力モードを使用して、一部の回路を使用されないときにディスエーブルにします。
- トランシーバーを可能な数だけ 1 つのタイルにパックして、サポート回路の複製を最小限に抑えます。

メモ : 作業しすぎないようにしてください。ワースト ケースのデバイス、プロセス、環境、およびデザイン アクティビティを必ず同時に設計する必要はありません。場合によっては、このような状況が発生する前にシステムのほかのコンポーネントが停止することがあります。このような場合、絶対定格のジャンクション温度を超えないようにし、デバイスが既知のステートから再起動できるようにする必要があります。このような状況でデータを処理することは、アップストリームまたはダウンストリームのボード コンポーネントが動作不可能な場合があるため、問題にならない可能性があります。

ソフトウェア設定およびアルゴリズム レベル

合成

合成ツールでは、エリア、パフォーマンス、ランタイム、および消費電力制約間でトレードオフを行います。集積度またはパフォーマンスの目標を満たすことが優先されます。クリティカルではないパスは、エリアおよび消費電力を最小限に抑えるように最適化されます。次のセクションでは、合成されたネットリストをさらに最適化する手法、フロー、および制約について説明します。

ここで説明するオプションおよび手法は、ザイリンクス XST を参照していますが、Synopsys Synplify または Mentor Precision などのその他の FPGA 合成ツールにも類似した機能があります。

一般

- 完成していて現実的なタイミング制約を供給

  これにより、クリティカル パスの最適化に焦点を絞ることができ、残りのパスを最適化してエリアおよび消費電力を最小限にできます。パフォーマンス要件を人為的に厳しく制約すると、複製が多数生成され、論理記述がハードウェア リソースに非効率的にマップされてしまいます。制約が緩すぎると、最適にマップされず、配置配線で予期するパフォーマンスを達成することが困難になります。

- ブラック ボックスを読み出す

  ブラック ボックスへのパスおよびブラック ボックスからのパスのタイミング情報およびブラック ボックス内のリソース使用量を取得できます。これにより、ロジック リソースの不必要な複製や最適ではない使用を回避できます。

XST による消費電力最適化

- 同時にアクティブなブロック RAM のポート数を最小限に抑える

  この最適化は、-power yes オプションによりイネーブルにされますが、複数のブロック RAM にまたがる RAM または ROM 記述が分解されます。タイミング制約を満たしながらアクティブなブロック RAM ポート数をクロック サイクルごとに最小限にするよう、アドレス ライン、ポート イネーブル、およびライト イネーブル制御信号が調整されます。

- パフォーマンスへの影響を考慮せずに最も消費電力を抑えるようにブロック RAM を強制的にマップ

  このメモリに関連したタイミング パスが重要ではないときに、block_power2 オプションを ram_style 制約に使用します。これにより、消費電力を 15% ～ 75% 節約できます。

XST によるエリア最適化

- area 最適化モードを使用

  可能な限り XST で area 最適化モードを使用します。これにより、リソース使用量が最低限に抑えられます。

- マップを強制的に決定

  特殊なロジック ブロックでは、パフォーマンスへの影響を慎重に予測した後、ツールで強制的にファンクションを専用ロジック リソースにマップできます。たとえば、カウンターを使用可能な DSP ブロックに強制的にマップできます。シフト レジスタは、スライスの SRL モードにマップ可能です。ワード数が少ないメモリがブロック RAM としてインプリメントされていて、その入力配線または出力配線が長い場合、配線の消費電力を削減できる可能性があるため、分散 RAM に強制的にマップするような場合もこの例です。

- リソース シェアリングの使用

  リソースを共有すると、算術演算子の数を最小限に抑えることができるので、デバイス使用率を削減できます。類似する算術演算子は、これらの出力が同時に使用されることがない場合、デバイスの共有リソースを使用してインプリメントできます。通常、リソース共有では因数分解された入力のいずれかを選択するために、マルチプレクサ ロジックが追加されます。因数分解を実行することでロジック使用量を最小限に抑え、ロジックの複製を回避できます。合成ツールでは、これらの最適化がデフォルトで実行され、潜在的なパフォーマンスへの副作用が制御されるので、この最適化をディスエーブルにする必要はほぼありません。

その他の XST オプション

- レジスタ バランス調整を使用してアクティビティを削減

  -register_balancing オプションを使用すると、組み合わせロジックに含まれるタイミング パスの長さが均一になるように、レジスタのタイミングが前方または後方に調整されます。このオプションは主にデザイン パフォーマンスの向上のために実行されますが、最長のタイミング パスを短くすることでグリッチの伝搬が最小限に抑えられるので、アクティビティが削減されます。レジスタ バランス調整は、演算器や幅の広いビット単位構造などアクティビティの高いロジックをインプリメントするパスで特に有益です。

- FSM エンコード方式

  大型ステート マシンのエンコード方式を多数試すことができます。たとえば、Gray エンコードを使用すると、ステート遷移間のビット変化数を最小限に抑えることができます。エンコード方式を変更すると、出力の生成に必要なロジック数に対して必要になる次のステートをデコードするためのロジック数に影響します。

インプリメンテーション

次のセクションでは、個別または組み合わせて使用することでさらに最適化を実行できるアルゴリズムについて説明します。

ネットリストの最適化

- アクティビティを意識した最適化 (高度なゲーティング) をイネーブル

  これらのアルゴリズムでは、論理式が解析されて、結果に影響しないソース レジスタがクロック サイクルごとに検出されます。ソフトウェアでは、FPGA ロジックに含まれている十分なクロック イネーブル (CE) リソースを使用して、無駄なスイッチイング アクティビティを回避する高精度のゲーティング信号が作成されます (図 4-1 を参照)。高度なクロックおよびデータ ゲーティングは、map -power high オプションを使用して制御できます。総ダイナミック消費電力は 15% 以上削減可能で、ほとんどの場合で挿入したゲーティング ロジックが原因でパフォーマンスが影響を受けることはありません。

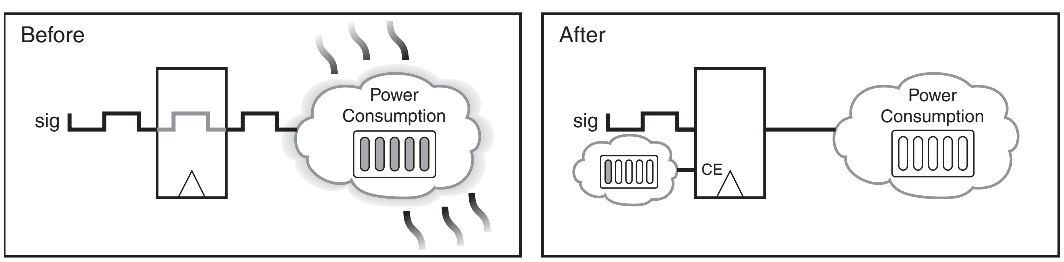

図 4-1 : 高度なクロック ゲーティングによるスイッチング消費電力の削減

- リソースを再配分する

  ネットリストの最適化では、ファンアウトの大きいネットが低消費電力の配線構造に再マップされます。たとえば、多くの負荷に分配されるリセット信号はグローバル ネットにマップされる可能性があります。グローバル ネットはパフォーマンスに対して最適化された専用配線で、キャパシタンス (容量) も比較的低い値です。また、残りのデータ パスを配置配線する配線リソースも解放されるので、配線の混雑が減り、全体的なキャパシタンスも削減されます。これらのすべての効果は map -power on オプションで制御され、ダイナミック消費電力を削減します。

配置

- キャパシタンスを意識した最適化
  - クロックの負荷をグループ化 : このプロセスではフリップフロップや DSP ブロックなどのエレメントの配置が再編成され、各クロック ネットの範囲が最小限に抑えられます。クロック負荷が最少数の水平または垂直クロック スパインに沿って配置されるとき、クロック領域内の未使用の分岐はソフトウェアでディスエーブルにできます。これにより、使用されるクロック リソース数およびバッファ要件が減り、ダイナミック消費電力が抑えられます。このプロセスは、map -power on オプションによって制御されます。
  - データ負荷をグループ化 : このアルゴリズムでは、パフォーマンス要件を満たしながらデザインに含まれるワイヤの全長が最短にされます。ダイナミック消費電力は、関連して増加するキャパシタンスが原因で、ファンアウトおよび配線構造の種類と長さと共に増加するので、データ負荷をグループにまとめると、消費電力を節約できます。このグループ化アルゴリズムも、map -power on オプションでイネーブルにされ、関連ロジックを隣接して配置することで消費電力の削減を達成できます。
- アクティビティを意識した最適化
  - シミュレーション結果からアクティビティを入力 : これにより配置プログラムで消費電力が最小限に済むように効率よくネットリストの優先順位が付けられてフロアプランされます。電圧およびキャパシタンスに加えて、アクティビティはダイナミック消費電力を決定する必須要素です。デフォルトでは、配置プログラムでデザインのパフォーマンスおよび配線目標を満たすように試みられます。シミュレーション結果のアクティビティ情報は、アクティビティが高いロジックおよびパスを配置するときに配置プログラムをさらにガイドします。これにより、これらの構造に対する内部 CLB 配線が増え配置密度が増すので、ダイナミック消費電力が削減されます。これらのアルゴリズムをイネーブルにするには、次のオプションを使用します。

    map -activity_file *file_name*.**saif**

    同一の階層セパレーター、最上位名などを使用して、ネットリストとシミュレーション出力のコンポーネントが効率よく一致するようにしてください。
  - **重要 :** キャパシタンスおよびゲーティング アルゴリズムの両方をイネーブルにする map -power xe オプションを使用することを検討してください。過去のデータに基づくと、このオプションがダイナミック消費電力を削減できる最も効率的なオプションです。

その他

- ツールのアップデートを確認

  ほとんどの IC コンポーネントでは、製造プロセスの成熟度に伴い、FPGA の消費電力特性が改善されます。FPGA 初期段階では、消費電力特性はシミュレーションから取得されています。エンジニアリング サンプルが入手可能になると、計測データを消費電力モデルに統合できます。デバイスが完全生産に進むと、複数の製造バッチにおけるプロセスのばらつきが完全に特性化

され、これらの計測で消費電力モデルがアップデートされます。消費電力予測および解析ツールでは、このアップデートされた情報を使用して精度の高い結果を出力できます。

- ソフトウェアの消費電力オプションを一括して試行

  ISE の SmartXplorer 機能または PlanAhead の [Design Runs] ビューを使用すると、消費電力に影響するさまざまなソフトウェア オプションを一括して試行できます。これらには、合成および配置配線ツールの設定を調整する定義済みのストラテジ セットがあります。独自のストラテジを追加するか、または既存のストラテジを編集して、複数のマシンで高速に異なる run を実行します。

デバイスまたはアーキテクチャを効率的に比較

デバイスまたはアーキテクチャを比較するには、次を実行します。

- 放熱係数を一致させる

  ツールにはそれぞれ異なるオプションおよびデフォルト設定があるので、比較するときは放熱係数を一致させる必要があります。これにより、スタティック消費電力を正しく比較できます。

- 使用されるリソースを一致させる

  アーキテクチャにはそれぞれ異なるリソース (シフト レジスタ、メモリ コントローラー、クロック マネージャーなど) および異なるサイズのリソース (LUT、BRAM、DSP ブロックなど) があるため、両方のツールで同等の数値を入力してください。たとえば、LUT SRL を使用してインプリメントされるシフト レジスタがサポートされないアーキテクチャでは、サポートするアーキテクチャと同じ機能を達成するためにフリップフロップ数が多く必要になります。

- I/O 設定を一致させる

  アーキテクチャではそれぞれ異なる I/O 規格、終端またはデータ キャプチャ、およびアライメント ブロックがサポートされる場合があります。比較するときは、同じ電圧レベルおよび機能になるようにしてください。

- アクティビティを一致させる

  ツールのデフォルトは、アーキテクチャおよびベンダーによって異なります。比較する前にこれらの情報を変更してください。

- 結果の値に何が含まれているかを理解する

  次のような質問を自問してください。何が含まれているか。何が除外されているか。すべての電圧の影響が含まれているか。オフチップ終端で散逸される消費電力は含まれているか。

第 5 章

まとめ

このガイドで説明した原則に従うと、デザイン サイクルを通してデザインの消費電力予測が実行しやすくなります。解析手法は総消費電力に影響する主な要因の特定を容易にし、ヒントおよび手法は消費電力が超えてしまっているときの最も効果的な対処方法を決断する際に役立ちます。当然デザインはそれぞれ異なるので、特定のデバイス、環境、プロセス、および締め切りに応じてこの資料を利用してください。

最終的には消費電力を最小限に抑えることで、同じ消費電力で実行できるアプリケーションの操作が増えます。エンド製品の動作コストを電気代、信頼性、および管理費のすべての面で削減できます。

その他のリソース

消費電力に関する資料

- ザイリンクスのウェブサイトで消費電力ソリューション ページを参照してください。

  http://japan.xilinx.com/power

- 『Lowering Power at 28 nm with Xilinx 7 Series FPGAs』(WP389)
- 『高度なクロック ゲーティングによるスイッチング電力の削減』(WP370)

ツール資料

- XPower Estimator (XPE)
  - 『XPower Estimator ユーザー ガイド』(UG440)
  - 『Seven Steps to an Accurate Worst-Case Power Analysis Using Xilinx Power Estimator (XPE) 』(WP353)
- XPower Analyzer (XPA)
  - グラフィカル インターフェイス : XPower Analyzer (XPA) ヘルプ
  - コマンド ライン ツール (`xpwr`) : 『コマンド ライン ツール ユーザー ガイド』(UG628)
- PlanAhead RTL 電力予測
  - 『PlanAhead ユーザー ガイド』(UG632) (第 5 章「RTL デザイン」の「消費電力予測」)

サポートおよびその他

- シリコン、ソフトウェア、IP に関する問題をアンサー データベースで検索したり、テクニカル サポートのウェブ ケースを開くには、次のザイリンクス ウェブサイトにアクセスしてください。

  http://japan.xilinx.com/support

- 内部リソースおよび I/O リソースの機能 :

  http://japan.xilinx.com/documentation の [デバイス] タブで [FPGA デバイス ファミリ] をクリック