7 System Level Design

低消費電力化設計と消費電力見積り

(株)インターデザイン・テクノロジー
荒木　大
araki.dai@interdesigntech.co.jp

松下電器産業(株)
竹村 和祥
takemura.kazuyoshi@jp.panasonic.com

三洋電機(株)
齊藤 博文
saitoh@ul.rd.sanyo.co.jp

低消費電力化設計への期待

携帯通信・モバイル情報機器を中心としてSoCのアプリケーションが急速に広がりつつある．製品の小型化に伴って，電池駆動で長時間動作が可能で低発熱が要求される．この一方で，製品の機能の多様化・高度化・高性能化を実現するためには，逆にSoCの規模や，SoCに実装される機能が増える傾向にあり，消費電力も増える要因が増している．

このようにSoCの低消費電力化への要求が厳しくなる中で，低消費電力化を達成するための設計技術の重要性が非常に大きくなった．

しかしながら，現状の低消費電力設計技術は，レイアウト・回路・プロセスレベルで行う手法や技術が中心であり，システムレベルの設計において適用可能な低消費電力設計技術は未成熟である．

現状を裏付けるデータとして，平成12年3月の「半導体技術動向に関する調査報告」[1]における低消費電力化技術の動向調査の抜粋を図-1に示す．

この報告から読み取れるポイントは2つある．1つ目のポイントは，システムレベルおよびアーキテクチャレベルの設計における低消費電力化手法による削減効果目標が，レジスタ・トランスファ・レベル（以下，RTLと呼ぶ）以下の下流設計における低消費電力化技術のそれよりも大きいことである．たとえばRTL設計や論理設計のレベルでの削減目標が1/2，プロセスレベルでの

設計フェーズ	最適化項目	最適化手法	削減目標	最も効果ある要因	実現時期
システム	チップ外通信低減	機能分割最適化	～1/100	speed	2005
	データ転送効率化	小振幅化		size	2002
	データ転送効率化	変調技術		size	2005
	データ転送効率化	チップ内データ圧縮伸張		size	2011
	電力予測	システムレベル電力予測		all	2005
アーキテクチャ	処理量低減	高効率アルゴリズム	～1/50	all	2002
	システム分割最適化	HW／SWの分担最適化		all	2005
	スイッチング頻度低減	データ表現最適化		size	2002
	スイッチング頻度低減	マルチクロック		speed	1999
	スイッチング頻度低減	非同期回路		speed	2005
	動作周波数低減	並列化		speed	1999
	不要動作停止	メモリバンク最適化		size	2002
	負荷容量削減	バス構造最適化		size	2005
	ソフトウェア最適化	メモリアクセス低減		size	2002
		低電力アーキテクチャ		all	2002
		低電力アルゴリズム		all	2005
	ソフトウェア最適化	低電力ソフトウェア		all	2008
	電力予測	ソフトウェアレベル電力予測		all	2008
		アーキテクチャレベル電力予測		all	2005

図-1　低消費電力化技術の動向（文献1）より抜粋）

目標が1/10に対して，アーキテクチャレベルは1/50でありシステムレベルでは1/100である．

2つ目のポイントは，システムレベルおよびアーキテクチャレベルの設計における低消費電力化技術は，RTL以下の下流設計における低消費電力化技術[2]と比べると未実現が多く，将来技術とされているものが多いことである．

本稿では，まず，システムレベル設計において低消費電力化を達成するための設計手法について解説をする．次に，システムレベル設計において設計対象の消費電力を見積るための手法を解説する．最後に，商用あるいは研究用として公開されている消費電力見積りツールあるいは消費電力最適化設計の具体例について解説を行う．

低消費電力化のための設計手法

まず，システムレベル設計において低消費電力化を達成するための設計手法をいくつか紹介する．この分類は，文献3)を基にしている．

(A) HW/SW partitioning

消費電力を考慮しながらハードウェアとソフトウェアの最適分割を決める手法である．通常，CPUやDSPなどは，専用のハードウェアに比べて電力消費が高い．低電力化の観点から言えば，専用ハードウェアに機能を押し込める方がよいが，設計効率，設計の柔軟性，設計コスト・期間など他の要因も考慮するとバランスのとれた機能分担が重要となる．後で紹介するAvalancheシステムは，このアプローチによって設計最適化を行う．

(B) Instruction-level power optimization

CPUの命令セットシミュレータ上でソフトウェアを実行して，命令発行状況をトレースすることでCPUの消費電力を見積り，アプリケーションソフトの設計変更やコンパイラの最適化技術などで消費電力の最適化を行う方法である．命令ごとの消費電力をあらかじめ計測しておくことで電力を見積る．通常，メモリアクセス時の電力消費が大きいので，キャッシュヒット／ミスのトレースが見積りの精度を高める上で重要である．

(C) Control-data-flow transformation

ハードウェアの動作合成技術の中で入力モデルのControl Data Flow Graph（以下，CDFGと呼ぶ）を変換することによって低電力なハードウェアを合成する手法である．

(D) Memory optimization techniques

使用するメモリのサイズやメモリへのアクセスを減らすことで，消費電力を減らす手法である．後で紹介するATOMIUMはこれに該当する．また，データの配置を変えてアドレスバスの遷移回数を最小化したりキャッシュミスを削減したりするアプローチもある[4]．

また，後で紹介するAvalancheもキャッシュ・メモリのサイズの最適化による設計の最適化を行う．

(E) Interface power optimization

バス上に流れるデータのエンコーディング方式を変えることで，バス上のトラフィックを軽減し，バス転送において発生する消費電力を減らす手法である．

(F) Variable-voltage techniques

システム稼働中に，動的に電圧制御を行うことによって低消費電力化を達成する手法である．

(G) Dynamic power management

非稼働状態では，システムを低電力のスリープ状態にさせる手法である．

(H) Approximate signal processing

演算精度を下げることによって低消費電力化を達成する手法である．特定のアプリケーションの性質に依存した設計手法と言える．

(A)から(E)は，従来のシステムアーキテクチャの枠組みの中で，設計によって低消費電力化を図るアプローチである．一方(F)から(H)は，低消費電力化が期待できる新しいシステムアーキテクチャを指向する立場にあるといえる．この意味において，本稿では，システムレベル設計としてのアプローチと考えられる(A)から(E)を実現するために必要な消費電力見積り技術にフォーカスして解説する．

消費電力の見積り手法

CMOS回路の主な消費電力は負荷容量の充放電時（回路の出力が0⇔1で遷移するとき）によるものであり，その消費電力は次式で表される．

$$P = CV^2 f \alpha \qquad (式1)$$

ただし，C：負荷容量，V：電源電圧，f：動作周波数，α：スイッチング率，である．

しかし，システムレベルの設計の段階において，このような回路レベルでの電力見積りを正確に行うことは到底できない．そこで，設計コードなどの情報から，実機

を実装する前の段階でシステムとしての消費電力量をモデル化して見積るための技術と手法が重要になる．

システムアーキテクチャの観点から考えると，消費電力の見積りは，「CPUやDSPなどのプロセッサ部」「ASICなどのカスタムハードウェア部」「メモリ部」「各コンポーネントをつなげるバス部」に分けて考えることができる．それぞれに対する，一般的なアプローチは次の通りである．

プロセッサ部での消費電力は，命令セットシミュレータを使用してソフトウェアを実行した際に呼び出される命令をトレースし，用意しておいた電力係数をかけて見積るのが一般的な手法である．

カスタムハードウェア部での消費電力は，ハードウェア中のコンポーネント端子（粒度はさまざまだが，たとえば論理ゲートの出力端子）のトグル率（秒あたりの遷移数）をシミュレーションによってトレースして，ゲートレベルでの電力見積り式である（式1）をベースにして推定するのが一般的な手法である．

バス部の消費電力は，シミュレーションによってバス上で転送されるデータのトグル率を求めることによって，ここから消費電力を見積るのが一般的手法である．

メモリ部の消費電力は，シミュレーションによってメモリのアクセス回数をトレースして，これとメモリの消費電力モデルを使って見積る方法が一般的手法である．

以降で，消費電力の見積り手法を「CPUやDSPなどのプロセッサ部」「ASICなどのカスタムハードウェア部」「メモリ部」「各コンポーネントをつなげるバス部」に分けて，一般的アプローチ以外の手法も含めて紹介する．

■プロセッサの電力見積り手法

（A）平均キャパシタンスによる見積り

CPUが活性化している時の平均キャパシタンスからCPU全体のパワーを見積る．プロセッサの平均的な特性を表した電力モデルを利用することで電力を見積る方法である．モデルが簡単であるが，見積りの精度はそれほど期待できない．

（B）バススイッチング率による見積り

データ，アドレス，命令のスイッチング率を計測し，CPU全体のパワーを見積る．これも，電力モデルを利用する方法だが，CPUバス上のスイッチング率を計測して，パワーを見積る点で（A）の手法よりは精度が高い見積りとなる．

（C）インストラクションレベルモデルによる見積り

命令セットレベルの電力モデルを用意して消費電力の見積りを行う方法である．

（方式1）
消費電力を

$$P = \sum_i (\beta_i \times N_i) + \sum_{i,j} (O_{i,j} \times N_{i,j}) + \sum E$$

で見積る．ただし，β_i：命令iのエネルギーコスト，N_i：命令iの呼び出し回数，$O_{i,j}$：命令iから命令jに遷移する際のエネルギーコスト，$N_{i,j}$：命令iから命令jに遷移する回数，Eは命令以外の要因（パイプラインストール，キャッシュミスなど）である．

アプリケーションを実行した命令発行状況をトレースし，各命令発行回数，命令遷移，キャッシュミスやパイプラインなどの発生をカウントし，電力係数をかけることで消費電力を見積る．

（方式2）
インストラクションレベルパワーを

$$I_s = \sum_j (IF_j \times AS_j)$$

で求める[5),6)]．

ただし，IF_j：機能ブロック別（ALU，Load/Store，Register write，branch，fetch & decode）の平均電流，AS_j：命令ごとのActivation係数（各命令が実行された際に動く前述の機能ブロック）である．

機能別に命令を分類しておき，分類ごとの実行回数と係数をかけることで見積る方法である．

（D）プロファイルドリブン合成手法[7)]

アプリケーションの設計コードをそのまま使ってシミュレーションを行うのは時間がかかるために，電力消費の特性を近似させた小さなソフトウェアコードを使って消費電力の推定を行う手法である．

具体的には，まずアーキテクチャシミュレータ上でソフトウェアを実行し，分岐やキャッシュミスなどのプロファイルをとる．次にこのプロファイルの特性に合致し，かつコード量が千分の一以下になるような小さな擬似ソフトウェアを作る．この擬似ソフトウェアをRTLシミュレータ上で実行することによって，オリジナルのコードを使うよりは短時間に消費電力の見積りが可能になる．また，オリジナルのアプリケーションの設計コードの場合の消費電力もこの結果から推定できる．

オリジナルのプロファイル特性に合致した擬似ソフトウェアを生成するのが技術的には難しいと考えられる．

■カスタムハードウェアの電力見積り手法

カスタムハードウェアの電力見積り手法について解説

する．この分類は，文献8)を基にしている．

(A) Fast Synthesis

動作合成あるいは論理合成ツールを使ってもとの設計よりも抽象度の低いモデルに自動変換を行って電力見積りを行うアプローチである．回路データが詳細化されるので，精度は増すが，検証に時間がかかるため，大規模回路の電力見積りには一般的には向かない．たとえば，論理合成ツールを使用してRTL設計からゲートレベルのネットリストを自動生成して論理シミュレータにかけてゲートレベル電力見積り手法を使用して電力を求めることができる．動作合成ツールを使ってビヘイビアレベル設計をRTL以下のモデルに変換して電力を見積るアプローチも登場してきている．

(B) Power models of high-level building blocks [9), 10)]

ALU，乗算器，メモリ，レジスタ，加算器，コントローラなどの演算器ごとにハードウェアの電力モデルを作成しておき，シミュレーションのプロファイルから得られたブロックごとの入出力とスイッチング率を使って電力を見積る方法である．アーキテクチャの変更に対する柔軟性が少なく，モデルの再構築が必要になる．

(C) Semantics of high-level language primitives [8)]

HDLなどで書かれた設計コードから，電力見積りの要素となるプリミティブを抽出し，シミュレーションによってこれらのプリミティブごとのスイッチング回数を計測して，電力係数をこれに乗じて消費電力を見積る方法である．

(D) Rules of thumb [11)]

回路の入出力ピン数や活性化率（エントロピー）から，面積と全ノードの遷移密度を見積り，消費電力を推定する．（式1）において，Cは回路の面積と置き換えて考える．回路の面積と遷移密度（$f\alpha$）は，RTLシミュレーション結果から求めた回路の活性化率（エントロピー）を使って算出する．

回路の構造情報を用いないので精度が粗いため，使用前にチューニングをする必要がある．同様なアプローチには文献12)もある．

■バスの消費電力見積り手法

バス上で消費される電力の見積りは，命令セットシミュレータとシステムバス上のトランザクションアクティビティをトレースするツールを使用して，バス上のトランザクションを観測して，ソフトウェアを実行した際のバスのアクティビティ率をトレースすることによって，バスで消費される電力を見積ることができる．具体例は文献13)などがある．

■メモリの消費電力見積り手法

メモリの電力を見積るためのモデリング手法には以下のアプローチがある．

(A) 信号レベルのモデルによる見積り [14)]

入力信号や出力信号の遷移確率から電力を見積るモデリング手法である．

(B) 命令レベルのモデルによる見積り [15)]

メモリの構造的なパラメータを使って電力を見積る手法である．メモリ生成ツールとの相性が良い．電力モデルは，メモリを構成するプリチャージ部，センスアンプ部，Address transition detection, バッファ部，行デコーダ部，chip enable MUXs, メモリセルなどのコンポーネントごとの電力の計算式からなる．計算式は,「パラメータ×係数」の総和できまる．係数はSpiceネットリストを使用したシミュレーションにより求める．

消費電力見積りツール

次に，システムレベル設計における消費電力見積り，あるいは低消費電力化設計を支援するツールの紹介を行う．

■ATOMIUM

ATOMIUMは，ベルギーの国立研究機関であるIMECで開発された，メモリアクセスの解析と最適化によって消費電力の改善を行うツールである[16)～18)]．

ATOMIUMの入力は，C言語のソースコード（アルゴリズム・機能レベル）である．タイミングの制約条件とともに与えると，電力消費の観点で改善されたC言語コードが出力される．

内部的には，実際の機能モデルのCコードをシミュレーション実行してメモリ（データ構造や配列）などへのアクセスサイズを解析する動的解析と，メモリアドレス計算の最適化などの静的解析があり，解析手段により動的・静的解析を使い分ける．入力するC言語のコードに特段の制約がない点でターゲットのアーキテクチャには非依存なツールである．

ATOMIUMは図-2に示すAnalysis/SBO/MC/RACEの4つのツールから構成される．

- Analysis：機能レベルのC言語コードを入力として，メモリアクセスのボトルネック解析を行う．

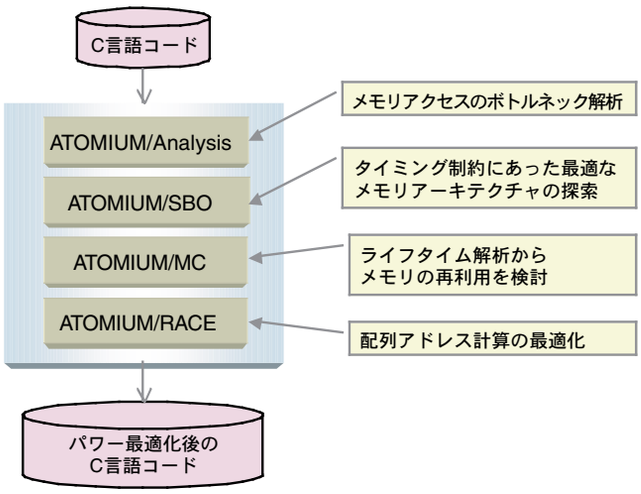

図-2　ATOMIUMのツール構成

- SBO（Storage Bandwidth Optimization）：タイミング制約にあった，最適なメモリアーキテクチャを探索する．
- MC（Memory Compaction）：変数のライフタイムやメモリ空間を考慮したメモリ配置の最適化を行う．
- RACE（Reduction of Arithmetic Cost of Expressions）：配列（＝メモリ）のアドレス計算の最適化を行う．

これら4つのツールを使うことによって，入力したC言語コードに対して，電力消費の観点で改善されたC言語のコードを得ることができる．

なお，ATOMIUMの一部機能はPowerEscape社が商用ツール化している[19]．

■ ORINOCO

ORINOCOは，ChipVision Design Systems AG社の，カスタムハードウェア設計を対象とした消費電力見積り／最適化ツールである[20)，21)]．

図-3に示すように，ORINOCOへの入力は，Cあるいは SystemCで記述した設計コードである．メモリやIPコンポーネントでの消費電力の情報については，別途ライブラリを用意して入力する．このライブラリ構築のためのツールもORINOCOに含まれている．

ORINOCOの見積り手法は，「カスタムハードウェアの電力見積り手法」の項で説明した，（A）Fast Synthesisである．スケジューリング，アロケーション，およびバインディングを行ってデータのフローを考慮した特定のアーキテクチャにマッピングした状態で，ソースコードのシミュレーションを実行して電力消費を予測する．

ORINOCOによって，異なるアルゴリズムやアーキテクチャの間での消費電力の違いを比較することができる．また，メモリへのアクセス状況やシステムのどの部分で電力消費が大きいか等の情報をグラフィカルな画面で表示させることで，解析に必要な情報を把握することができる．図-4がORINOCOの解析結果の画面例である．最終的にORINOCOで行ったアーキテクチャ・マッピングの結果は，制約ファイルのかたちで動作合成ツールに渡すことができる．

ORINOCOは，図-3に示す3つのツール（DALE，RIO，BEACH）から構成されている．

ORINOCO-DALEは，ORINOCOの中核をなすツールで，これを用いて電力見積り，最適化を行う．ORINOCOはマクロ・ベースの設計を仮定しており，

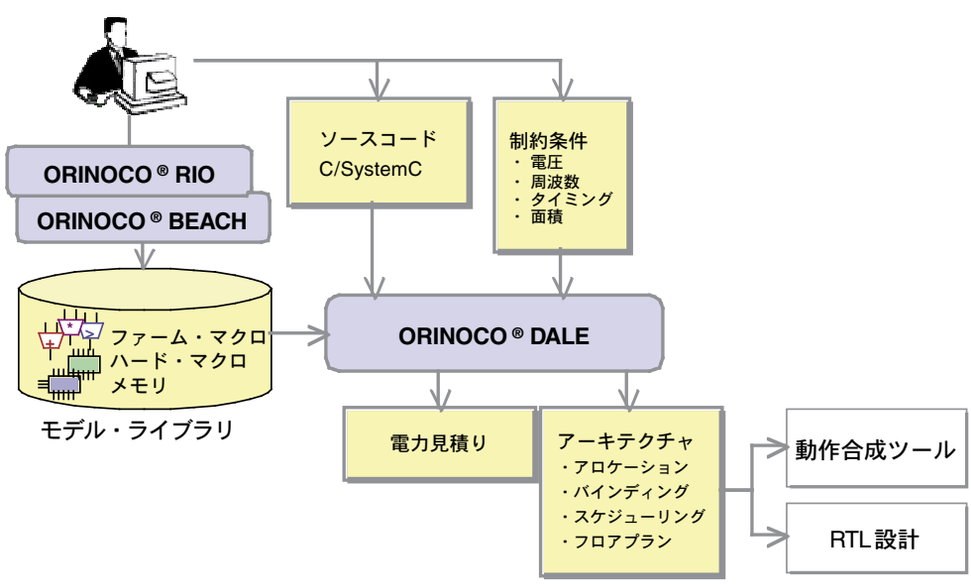

図-3　ORINOCOのツール構成

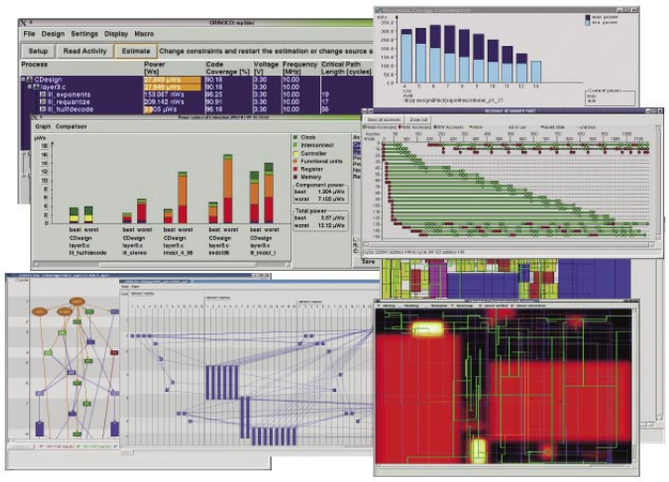

図-4　ORINOCOの画面例

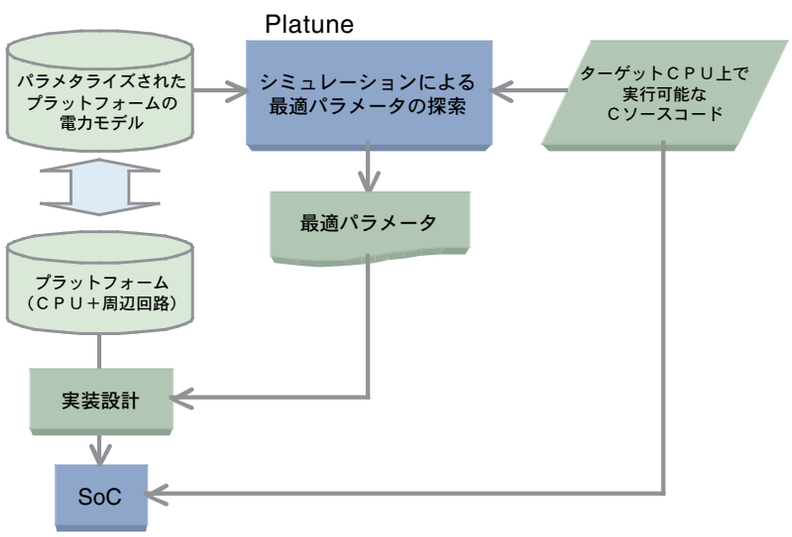

図-5　Platuneのデザインフロー

メモリやIPコンポーネントの電力はモデルライブラリを作成して見積る．

モデルライブラリの構築には，残りの2種のツールを使用する．ORINOCO-RIOは，RTLで記述されたコンポーネントのライブラリを作成するためのツールで，ORINOCO-BEACHが，メモリやIPコンポーネントのライブラリを作成するためのツールである．

■ Platune[22)]

Platune（Platform Tuner）は電力消費が少ないアーキテクチャ・マッピングを探索・最適化するツールである．図-5にPlatuneによるデザインフローを示す．Platuneでアーキテクチャ探索が行えるSoCは，MIPSのプロセッサ＋メモリ＋バス＋周辺ペリフェラルで構成されて，いくつかのパラメータによってアーキテクチャの構成の変更が可能である範囲に限定している．PlatuneにC言語のソースコードを与えると，そのコードを実行するために最適なパラメータの構成を，自動的に探索する．

Platuneで最適化できるパラメータには，CPUの電圧，キャッシュのサイズ・ラインサイズ・セットサイズ，バス幅・バスエンコーディングの方法，UARTのバッファサイズ，DCTの解像度などがある．

消費電力の見積りは，各モジュールの電力モデルを利用したシミュレーションによって行うが，Platuneでは，既存の手法を使って消費電力の見積りを行っている．

CPUは文献23）の方法で消費電力を見積る．命令ごとの消費電力をゲートレベルシミュレーションであらかじめ計測してライブラリ化しておく．ターゲットのアプリケーションを命令セットシミュレータで実行したトレース結果を使って見積る．直前に実行した命令との関連を考慮してライブラリ化しておく点が特徴である．キャッシュ・メモリの消費電力は文献24）の方法で，サブコンポーネントごとのキャパシタンスのモデルを用意しておいて，シミュレーションでスイッチング回数をトレース

して見積る．バスは，バスのキャパシタンスをモデルとして用意しておいて，シミュレーションでバスのスイッチングをトレースできるようにしている．ペリフェラルは，プロセッサと同様な手法を用いている．

Platuneは以下の手順でアーキテクチャ探索を行う．ターゲットコンパイラで設計コードをコンパイルして，命令セットシミュレータ上でシミュレーションを実行する．性能モデルと電力モデルを利用してソフトウェアをシミュレーションすることにより，消費電力と実行時間などの性能値を見積る．

アーキテクチャの探索は，設計者が変更可能と指定したパラメータ候補の組合せに対して探索を行うが，すべての組合せでのシミュレーションを実行すると多大な計算時間がかかるので，分枝限定法を使って探索空間の枝刈りを行いながら高速に探索を行う．Platuneは，消費電力と遅延時間のパレート曲面に存在する解候補を，最適解の候補集合として列挙して出力してくれる．設計者は，この中から1つを選んで，そのパラメータ設定でプラットフォームを実装すればよい．

Avalanche [25)]

Avalancheは，消費電力の観点で最適なHW/SW分割を行うことを目的にしており，合成ツール，シミュレータ，消費電力見積りツールを利用したツールチェーンによってこれを実現している．

インストラクションごとの，リソースの利用率を計算して，アプリケーション（Cレベルの記述）の中でHW化が有望なクラスタ（制御文，関数などの構造）をいくつか見つける．各候補に対して，HW/SW分割後の消費電力を，プロセッサコア，ASICコア，キャッシュ，メモリの各々の消費電力を見積もって評価することで，最良の分割ポイントを見つける．

また，Avalancheは，与えられたパフォーマンスの制約下で，キャッシュとメインメモリ（外部メモリ）の消費電力が最小になるキャッシュサイズを見つけることも行う．

Avalancheの設計フローを図-6に示す．

設計フローにおけるHW/SW分割手順と見積りの役割を説明する．

① アプリケーション記述（C言語）をクラスタに分割する．クラスタとは，ループ文，if文などの構造あるいは関数呼び出しなどを意味する．
② 各クラスタに対して，HWに割り当てた時に発生するバス転送に伴う電力消費を見積る．これが大きいクラスタは，HW化する候補から削除する．
③ 各クラスタをHWに実際に割りつけた場合の優劣を比較する．
（A）ASICのリソース（ALU , multiplier , shifterなど）をallocateして，候補クラスタをスケジューリングする．
（B）候補クラスタを実行した時のリソースの利用効率を，プロセッサで実行した場合とASICで実行した場合で比較する．プロセッサで実行した場合の利用効率はISSを使って見積る．
（C）ASICとして実装する方が利用効率が良い場合は，この利用効率と，リソースごとの平均電力消費（これは所与）を使って，ASICコアでの電力消費量を見積る．
④ 最大の利用効率が得られるクラスタを，ASICとして動作合成，論理合成する．
⑤ ゲートレベルでの電力消費の詳細見積りを行う．

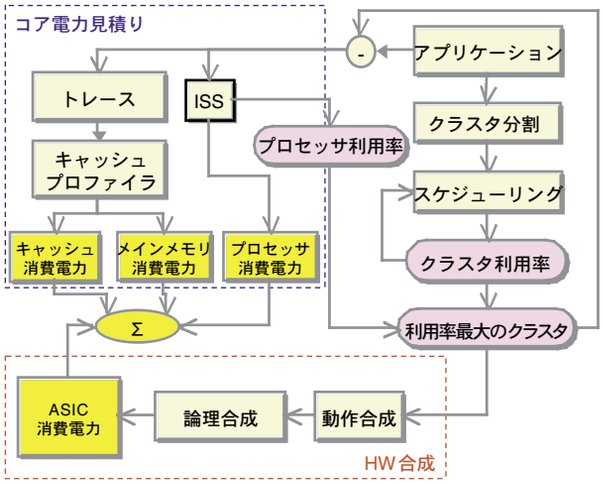

図-6　Avalancheのデザインフロー（文献25）より）

具体例として，SPARCLiteをプロセッサコアとするターゲットで，MPEG2エンコーダなどの6つの5Kから230Kバイトのcコードのサンプルで評価実験を行い，電力消費を30％から90％まで削減できるHW/SW分割ポイントを発見できたことが報告されている．

システムレベル設計における消費電力見積りの展望と課題

動作レベルの設計記述を入力とする電力見積り技術あるいはシステムレベルでの低消費電力化設計技術は，この数年の学会等の調査でも論文数が多く，さまざまなアプローチがある．一般的には，システムレベル設計における見積りは動的解析（シミュレーションによる解析）

のアプローチが多い．つまり，シミュレーションの実行に必要な設計情報が揃っていないと見積りができない．

しかしながら，システムレベルでの見積り技術は，実設計に応用した場合の成功例の報告が少なく，いまだ技術としての発展期であるといえる．この面で，それぞれの技術が実用性・実効性の観点でどこに課題があるのかが見えない状況にある．

筆者らは本特集の「システムレベル設計フローと設計言語」で説明するシステムレベル設計のための設計フローの上で消費電力見積りの技術あるいはツールを使ってゆく上での検討と課題の分析を行った[26]．この結論を説明することによって，本稿のまとめとしたい．

一般に，システムレベル設計での消費電力見積りは，より下流レベルでの見積り手法と比較すると見積り精度を高めることが難しい．しかし，各技術の利用法を工夫することで実用性が期待できる．

たとえば，システム全体の絶対値としての消費電力見積りの精度に期待するのではなく，アーキテクチャの最適化を行う上でのトレードオフを見るという目的で利用することは有効利用法の1つである．

プラットフォーム・ベース・デザインや流用設計が繰り返されるドメインにおいては，リファレンスとなる見積りデータを蓄積して再利用するとともに，見積りデータベース自身も継続的に洗練させてゆく仕組み作りが重要となる．これらの結果として見積りの精度も高めることができる．

また，システムレベル設計の機能決定のフェーズで行った見積り結果を，アーキテクチャを決定するフェーズにおいても活用する，さらにアーキテクチャ決定で行った見積り結果をさらに下流の設計でも活用するといった情報の受け渡しが必要である．一方で，下流設計で行った見積り結果を上流での設計にどうフィードバックするかに関しては，技術的にも研究例・実施例が少ない点で，将来のテーマとして注力すべきアイテムの1つと考える．

SoCのシステムレベル設計における電力見積りの1つの問題としては，アナログ部分（イメージセンサなど）の電力を高い抽象度でどう見積るかという課題がある．アナログ回路の低消費電力設計をも含めたかたちでの最適化設計の手法も今後の課題の1つといえる．

謝辞　本稿の執筆にあたって，JEITA/EDA技術専門委員会／SLD研究会の橋本毅久（ソニー（株）），木村仁（三菱電機（株）），黒坂均（NECエレクトロニクス（株））の各氏のご協力をいただきました．また，石原亨氏（Fujitsu Laboratories of America）およびNTTアドバンステクノロジ（株）のご協力をいただいたことに感謝いたします．

参考文献

1）半導体技術動向に関する調査研究報告，平成12年3月，日本電子機械工業会．
2）栗田敏明，竹本光雄：LSIの低消費電力設計，沖テクニカルレビュー第188号，Vol.68，No.4，pp.36-39．
3）Benini, L. and De Micheli, G. : System-Level Power Optimization: Techniques and Tools, ACM TODAES, Vol.5, No.2, pp.115-192（Apr. 2000）．
4）Panda, P.R., Dutt, N.D. and Nicolau, A. : Memory Issues in Embedded Systems-on-Chip: Optimizations and Exploration, Kluwer Academic Publishers, Boston, Massachusetts（1998）．
5）Brandolese, C. et al. : An Instruction-level Functionality-based Energy Estimation Model for 32-bits Microprocessors, DAC2000.
6）Sato, T., Otaguro,Y., Nagamatsu, M. and Tago, H. : Evaluation of Architecture-level Power Estimation for CMOS RISC Processors, Proc. of the Symposium on Low Power Electronics, pp.44-45（1995）．
7）Hsieh, H., Pedram, M.: Microprocessor Power Estimation Using Profile-Driven Program Synthesis, IEEE Trans. on CAD, pp.1080-1089（Nov. 1998）．
8）Lippens, P. et al.: High-level Power Estimation Methodology Applied for Processor-level DTSE, In“Unified Low-power Design Flow for Data-dominated Multi-media and Telecom Applications”, Catthoor, F., Kluwer Academic Publishers（2000）．
9）Landman, P.E. and Rabaey, J.M. : Architectural Power Analysis: The Dual Bit Type Method, IEEE Trans. on VLSI, pp173-187（June 1995）．
10）Katkoori, S. and Vemuri, R.: Architectural Power Estimation Based on Behavior Level Profiling, Journal on VLSI Design, Special Issue on Low Power Design, Vol.7, No.3, pp.255-270（Aug. 1998）．
11）Nemai, M. and Najm, F. N.:Towards a High-Level Power Estimation Capability, IEEE Trans. on CAD, pp.588-98（June 1996）．
12）Marculescu, D., Marculescu, R. and Pedram, M.: Information Theoretic Measure for Power Analysis, IEEE Trans. on CAD, Vol.15, No.6, pp.599-610（June 1996）．
13）Fornaciari, W. et al. : Power Optimization of System-level Address Buses based on Software Profiling, DATE2000 .
14）Bernacchia, G., Papaefthymiou, M. : Analytical Macromodeling for High-Level Power Estimation, ICCAD1999 .
15）Coumeri, S., Thomas, D. : Memory Modeling for System Synthesis, IEEE Trans. on VLSI, pp.327-334（June 2000）．
16）Balasa, F.，Catthoor, F. and De Man, H. : Practical Solutions for Counting Scalars and Dependences in ATOMIUM − A Memory Management System for Multi-dimensional Signal, Processing IEEE Trans. on CAD, Vol.16, No.2, pp.133-145（Feb. 1997）．
17）Miranda, M., Catthoor, F., Janssen, F. and De Man, H. : High-level Address Optimisation and Synthesis Techniques for Data-Transfer Intensive Applications, IEEE Trans. on VLSI Systems, Vol.6, No.4, pp.677-686（Dec. 1998）．
18）http://www.imec.be/design/atomium/
19）http://www.powerescape.com
20）Nebel, W. : Predictable Design of Low Power Systems by Pre-Implementation Estimation and Optimization, ASP-DAC2004, pp.12-17.
21）http://www.chipvision.com
22）Givargis, T., Vahid, F., Henkel, J. : System-Level Exploration for Pareto-Optimal Configurations in Parameterized System-on-a-Chip, IEEE Trans. on VLSI, Vol.10, No.4（Aug. 2002）．
23）Tiwari, V., Malik, S. and Wolfe, A. : Power Analysis of Embedded Software: A First Step Toward Software Power Minimization, IEEE Trans. VLSI Syst., Vol.2（Dec. 1994）．
24）Evans, R.J. and Franzon, P.D.: Energy Consumption Modeling and Optimization for SRAMs, IEEE J. Solid-State Circuits, Vol.30（May 1995）．
25）Henkel, J. and Yanbing Li. : Avalanche, An Environment for Design Space Exploration and Optimization of Low-Power Embedded Systems, IEEE Trans. on VLSI，Vol.10，No.4（Aug. 2002）．
26）JEITA EDA技術専門委員会SLD研究会：研究・標準化動向分析にもとづくシステムレベル設計手法の提案，DAシンポジウム2003．

（平成16年3月22日受付）