デジタル・オーディオLSIのCベース設計

岡田　敦彦　　数馬　晋吾
岩渕　信正　　槙　和彦

Cベース設計手法の適用により，短期間でシステムLSIが開発できるようになってきた。沖電気でも仮想プラットフォームを基本とするCベース設計手法を開発し，短納期のシステムLSI開発を実現している。しかし，演算の複雑なアルゴリズムをシステムLSIへ実装しようとすると，ハードウェアモデルの詳細化に工数がかかり，期待するCベース設計の効果が未だ得られていない。

そこで我々は，従来のCベース設計手法に動作合成ツールを取り入れることで，複雑なアルゴリズムをシステムLSIへ実装する手段を実用化した。本稿では，この動作合成ツールを適用したアルゴリズムの実装手段についてまとめ，具体例としてデジタル・オーディオ製品への適用と今後の展開について紹介する。

Cベース設計手法による開発期間の短縮

図1は，Cベース設計手法がシステムLSIの開発期間を短縮できるという利点について説明したものである。システムLSIの高機能・高速化にともない，机上で検討していたシステム仕様では，システム検証工程において，性能未達が発覚するというリスクが存在していた。

我々は，後工程で発生するリスクを最小限に抑えるべく，C++言語を拡張したシステム記述言語SystemCをベースにして，仮想プラットフォームを用いた設計手法を開発した[1)]。この設計手法を適用することで，システム検証結果を開発段階の早い時期にフィードバックすることができ，後工程における性能未達のリスクを低減し，システムLSIの開発期間を30%削減することに成功した[2)]。

ところが，音響処理のような複雑なアルゴリズムをシステムに実装する場合，期待するハードウェア開発期間の短縮効果が得られなかった。これは，最初に抽象度の高いハードウェアのモデルでシステム検証を行った後，LSI実装レベルへの詳細化は人手で行っていたため，詳細化のためのモデルの変換工数と変換後の検証工数が増大するためである。特に，演算量が多いアルゴリズムの場

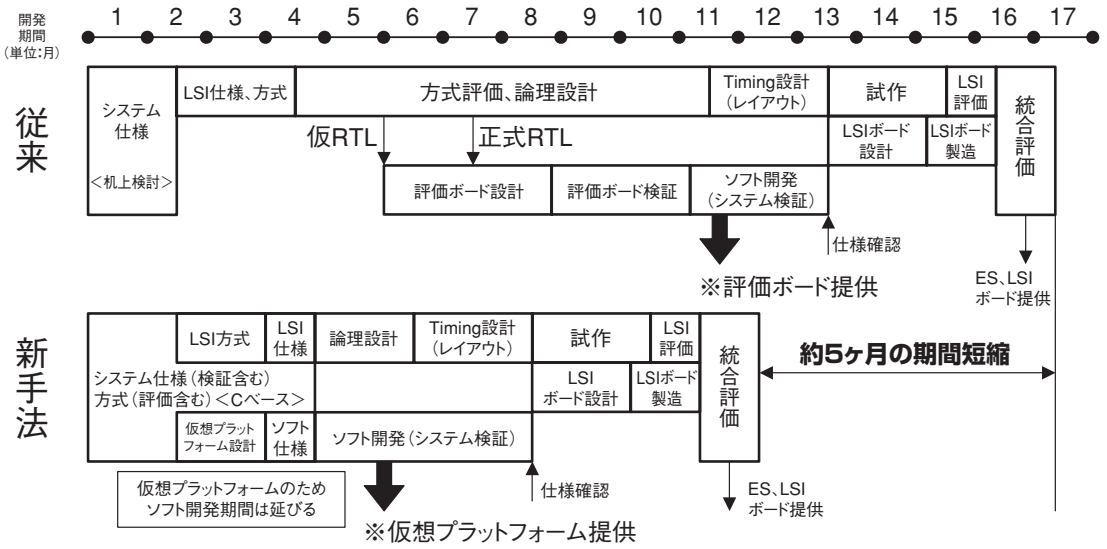

図1　従来との設計手法の比較

合，新手法における論理設計工程の開発工数は，従来手法での方式評価・論理設計工程と変わらない。そこで，動作合成ツールを適用し，検証効率を改善することで，ハードウェアモデルを短期間に詳細化し，アルゴリズムを実装するシステムLSIの開発期間を短縮する。

動作合成ツールを適用したCベース設計手法

図2に，アルゴリズムをシステムLSIへ実装するための設計フローを示す。

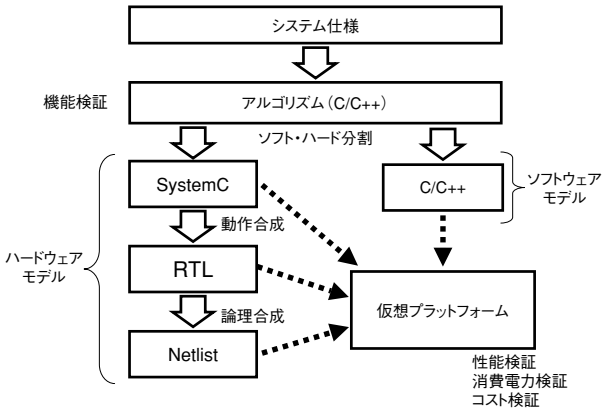

図2　設計フローと検証戦略

システム仕様が決まると，機能・性能・消費電力・コストの目標仕様を決定する。

次に，機能仕様を満たすアルゴリズムを開発する。そして，アルゴリズムからドライバとハードコアのモデルを分割し，システムをエミュレートする仮想プラットフォームを構築する。

ハードコアのモデルは，抽象度の高いSystemCモデルに変換した後，ドライバの開発と並行して動作合成・論理合成と段階的に詳細化していく。

詳細化されたハードコアのモデルを仮想プラットフォームに実装し，ドライバ開発およびシステム検証を進める。

以下に，仮想プラットフォームの構築，そしてアルゴリズム，ハードコア，ドライバの各開発の設計フローとその検証戦略について説明する。

(1) 仮想プラットフォーム

動作合成ベースの設計手法においては，ハードコアを開発してからドライバを開発すると，開発期間の短縮が望めない。従って，システムをエミュレートする仮想プラットフォームには，ドライバとハードコアがコンカレントに開発できる構成が必要となる。そこで，独自に開発した設計検証用クラスライブラリC-PLATをベースとした仮想プラットフォームを構築し，ハードコアのモデルを実装する。C-PLATクラスライブラリは，主に抽象度の異なるハードウェアのモデルを接続するためのインタフェースを提供している。

このC-PLATクラスライブラリを利用して，開発対象となるハードコアモデルのインタフェース部をSystemCのBCAレベルで実装する（**図3**）。

SystemCのBCAレベルはバス転送サイクル精度を持つため，バスマスタからのアクセスのサイクル精度が実際のシステムのサイクル精度と同等になる。また，ハードコアのモデルのインタフェース部をSystemCのBCAレベルで実装さえしておけば，仮想プラットフォームでシステムをシミュレートするのに，アルゴリズム部のアーキテクチャが最終決定されている必要はない。つまり，C-PLATクラスライブラリを利用した仮想プラットフォーム上では，ドライバ開発において実LSIシステムと同等の環境でありながら，ハードコアの開発を並行して行うことができる。

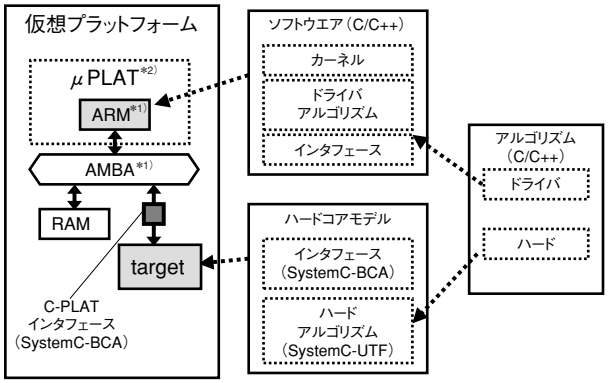

図3　仮想プラットフォームへのアルゴリズム実装

(2) アルゴリズム開発

アルゴリズムの開発は，C/C++言語を用いてパーソナルコンピュータかワークステーション上にて行う。高速で演算することができ，豊富な解析ツールがそろっているため，演算結果の精度や特性を含めた全機能を十分検証することができる。

(3) ハードコア開発

ハードコア開発では，次の3つのステージに分かれる。

① C/C++からSystemCへ変換
② SystemCからRTLへ動作合成
③ RTLからNetlistへ論理合成

＊1）ARM, AMBAは英国ARM Ltd.の英国およびその他の国の登録商標です。 ＊2）μPLATは沖電気工業(株)の日本，ドイツ，英国における登録商標です。

①では，前工程で検証されたソースに対して，C/C++言語からSystemCのUTFモデルへ変換し，バスインタフェース機能をSystemCのBCAレベルで実装する。新たに追加されるインタフェース機能については十分機能検証しておく必要はあるが，ハード化されるアルゴリズムについては，変換部の等価性だけを検証し，検証工数を短縮する。

②では，SystemCからRTLへの変換作業として，たとえば，配列変数のメモリへのマッピングや演算器の共有化といったリソースの抽出や，演算をどのサイクルで実行するかといったスケジューリングを行う。こうしたリソースの抽出とスケジューリングを人手で行うと，従来と変わらない変換工数が発生し，アルゴリズム開発と同じ検証ベクタを全て実行しないと機能等価性が保てない。しかも，人手による変換作業のため，バグの混入とその検証工数が増大することになる。

そこでForte社の動作合成ツールCynthesizer*3) を採用し，動作合成に適用することで，SystemCからRTLへの変換工数，およびバグ混入を検出する検証工数の低減を可能とする。さらに，人手による作業を極力排除することで設計品質が向上できる[3)]。

また，一般的に自動変換ツールによる処理では，変換工数の低減はできてもリソースの共有化と処理サイクル数の短縮がトレードオフになり，ゲート規模が増大するか性能が低下する。Cynthesizerは，リソースの指定や処理サイクル数の制御が容易にでき，しかも合成時間は，30kGのゲート規模であればSystemCからRTLまで20分程度と，非常に高速である。そのため，リソースや処理サイクル数を繰り返し調整しながら，短期間のうちに，性能とゲート規模を最適化することができる。

最終的にハードコアのゲート規模と消費電力の検証は，③のフェーズで行う。このフェーズは，従来の論理合成ツールおよび消費電力解析ツールを用いることで容易に検証することができる。

動作合成ツールを導入することで，ハードコアのモデルの変換工数だけでなく，機能・性能・消費電力・コスト検証工数の短縮が可能となった。

(4) ドライバ開発

ドライバ開発では，アルゴリズムから分割したCプログラムに対して，ハードウェアへのアクセス用インタフェースの追加，システムのカーネルへの組み込み，性能・消費電力・コードサイズ等の最適化を行う。

ドライバ開発における機能検証は，ドライバアルゴリズム部については等価性を，ハードウェアへのアクセス用インタフェース部については十分な機能検証を行うことで，検証工数を低減することができる。カーネルへの組み込みについては，従来の結合テストを行う。

性能・消費電力といったシステム検証は，その検証目的に応じた抽象度のハードコアモデルを仮想プラットフォームに実装して行う。たとえば，システム全体の性能評価にはSystemCモデル，消費電力評価にはNetlistといったように，目的に応じたハードコアモデルを選択して検証を進めることになる。

ハードコア開発に動作合成ツールを適用していることで，システム検証によって機能バグや性能未達が判明しても，短期間でフィードバックができる。

オーディオLSIへの適用

今回，3Dサラウンドと5バンドイコライザの音響アルゴリズムについてCベース設計を適用し，モバイル・パーソナル向けオーディオLSI ML2602（**写真1**）を開発した。

これら音響アルゴリズムをハードコア化するに当たり，C-PLATクラスライブラリを用いたモバイル・パーソナル

TIPS 【基本用語解説】

SystemC：
C++言語を拡張したシステム記述言語。インタフェースと機能を分離してモデリングすることができ,抽象度の異なるモデルを混在してシミュレーションすることが可能。一般に抽象度が高いほどシミュレーションは高速だが,サイクル精度は低い。以下に,主な抽象度レベルを紹介する。

UnTimed Function（UTF）レベル：
時間概念を持たない抽象度レベル。

Timed Function（TF）レベル：
時間概念を含む抽象度レベル。

Bus Cycle Accurate（BCA）レベル：
バス転送サイクルの時間概念をもつ抽象度レベル。

Register Transfer（RTC）レベル：
レジスタ転送サイクルに合わせた時間概念をもつ抽象度レベル。現在の論理設計手法に用いられるRTLと同じレベル。

*3)CynthesizerはForte Design Systems社の登録商標です。

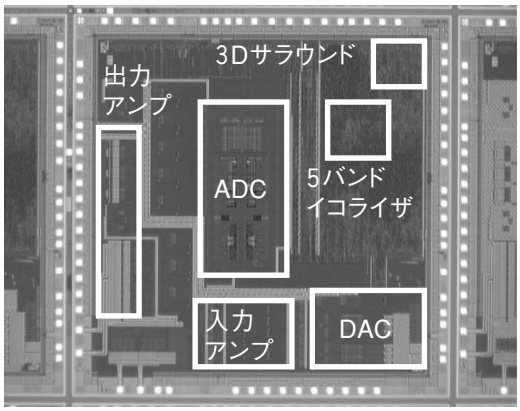

写真1　音響アルゴリズムを搭載したML2602

向け仮想プラットフォームLPMA（Low Power Mobile Audio）を構築した。**図4**にブロック図を示す。

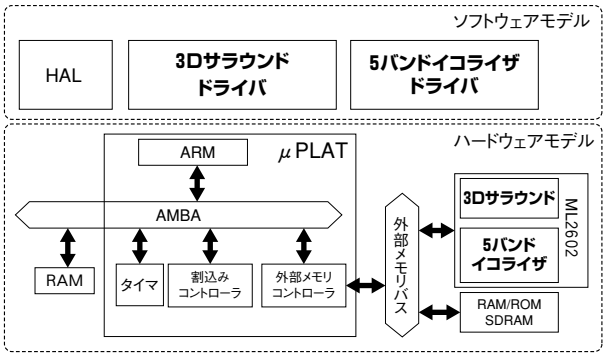

図4　LPMAブロック図

LPMAは，μPLATを中心にRAMモデルや外部メモリバス経由で3Dサラウンドと5バンドイコライザのハードコアモデルを接続している。ソフトウェアは，それらのドライバとタスクスケジュール機能を持ったシステムカーネルHAL（Hardware Abstraction Layer）から構成されており，LPMA上に搭載されている。

動作合成ツールを用いて開発した音響アルゴリズムのハードコアの開発期間と諸元を**表1**に示す。開発工数は設計当初6人月を見込んでいたが，実際には2人月でアーキテクチャ開発を完了し，ハードコアの開発期間を1/3に短縮することに成功した。ゲート規模は設計当初の見積り通りとなり，人手によるRTL設計と同程度に実装することができた。消費電力が見積り値に対して40%から16%と小さいのは，机上検討での見積り精度が低いためと思われる。

なお，イコライザの開発において，システム検証の結果，特性不良が発覚し，全演算のビット幅を変更しなければならない事態が発生した。しかし，仕様変更からアルゴリズムの検証（機能検証，全バンド周波数特性評価込み），動作合成，論理合成までを4.5時間で完了できた。従来のRTL設計なら4〜5日は工数を要したであろう。本手法の真価が十分に発揮された事例である。

表1　開発期間と諸元

	設計工数［人月］	ゲート規模［kG］	消費電力［mW］
3D サラウンド	1.0 (3.0)	15 (18)	1.5 (9.5)
5バンド イコライザ	1.0 (3.0)	18 (16)	2.3 (5.8)

※（）内は設計当初の見積り値。ゲート規模見積りは、アルゴリズムのCソースから演算量を抽出し、リソースを推定。消費電力見積りは、演算回数とアキュムレータのビット幅からトグルする最大データビット数を抽出して計算した。

最　後　に

我々は，アルゴリズムのシステムへの実装を含めた開発期間を短縮すべく，動作合成ツールを適用したCベース設計手法の開発に取り組んできた。

この手法を音響アルゴリズムに適用したところ，きわめて短期間で，しかも人手で設計した場合と同等のローコスト・低消費電力なハードコアを実現できた。

今後は，“音の沖電気”を目指し，モバイル・パーソナル向けサウンド系IPをCベースで設計することにより，ローコスト，低消費電力なシステムLSIを短納期で提供していく。◆◆

■参考文献

1)「ハード - ソフト協調検証，短期開発SoCの必需品へ」，日経エレクトロニクス，2004 12-6，no.888，日経BP社，pp.65–72，2004年
2)「沖電気，システムLSIの開発期間を1/3削減」，OKIプレスリリース，2004年11月29日
3)「沖電気，システムLSI開発の設計期間を3分の1に短縮」，OKIプレスリリース，2005年4月14日

●筆者紹介

岡田敦彦：Atsuhiko Okada. シリコンソリューションカンパニー デザイン本部 SoC設計部 デジタルサウンド開発チーム
数馬晋吾：Shingo Kazuma. シリコンソリューションカンパニー デザイン本部 SoC設計部 デジタルサウンド開発チーム
岩渕信正：Nobumasa Iwabuchi. シリコンソリューションカンパニー デザイン本部 ADCソフトウェア開発部 民生応用第一チーム
槙和彦：Kazuhiko Maki. シリコンソリューションカンパニー デザイン本部 プラットフォーム設計部