

9N-6 トップダウン方式によるLSIの自動論理設計

山口 雅之、野田 浩明、竹田 信弘、藤本 徹哉、神戸 尚志

シャープ株式会社 生産技術開発推進本部 生産技術研究所

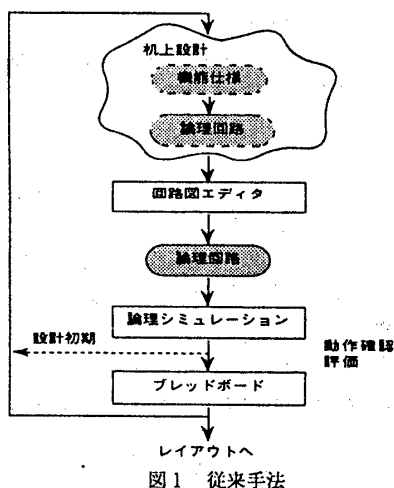
1. はじめに

LSIの設計においては、アーキテクチャ設計などシステム設計が最終的なLSIの性能やコストに大きな影響を与えるが、現在のCAD技術ではシステム設計の自動化は達成されていない。また、実際のLSI開発ではブレッドボード試作段階においてシステム検証を行なっている。目標とするシステム性能やコストを得るためには、システムレベルの設計から試作検証までを繰り返すことになる。よって、そのターンアラウンドタイムの短縮がCAD技術にとって重要な課題である。

当社では、自動論理合成とシステム検証を組み合わせたトップダウン設計手法を採用した。これにより、LSI設計のシステム設計から試作までのターンアラウンドタイムを大幅に短縮することが可能となる。この手法を用いたLSI開発では、システムレベルの改良を繰り返すことにより短期間のうちにLSIの動作速度やチップサイズ面で人手設計に比べて性能を大幅に改善することができた。本稿では、適用結果とその評価について述べる。

2. トップダウン設計手法とその設計環境

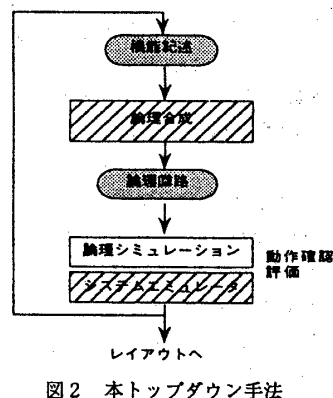
図1に従来手法、図2に本トップダウン設計手法に



おける設計フローを示す。本手法の設計環境の大きな特徴は(1)ハードウェア記述言語による機能設計および論理合成の採用と、(2)FPGAによるエミュレーション技術を利用したブレッドボードを用いない試作検証の採用である。

従来手法では、システム設計は机上で行なわれ、詳細設計を回路図エディタを用いて行なっていた。この方法ではシステム仕様の変更は論理回路に大きな変更を及ぼすため、システム性能やコストの最適化には多大な時間がかかる。またブレッドボード試作によるシステムの動作検証は初期コストが大きい上に修正に多くの手間を要するため、設計が成熟するまで試作検証が開始できず、検証時期が遅れることになる。また、設計から試作検証までのターンアラウンドタイムも長くなる。

本トップダウン手法ではハードウェア記述言語により機能レベルで設計を行ない、論理合成によってゲートレベルのネットリストを得る。設計はシステムのアーキテクチャレベルで行なわれるため、設計者の意図するアーキテクチャを記述することができ、ネットリストが得られるまでの時間も短い。また、得られたネットリストはシステムシミュレータ上のFPGAに短時間でプログラムできる。これによりブレッドボード検証では不可能であった初期設計回路から試作検証を行なうことができ、設計から試作検証までのターンアラウンドタイムも短縮することが出来る。



Automatic Logic Design for LSI by Topdown Method
 Masayuki YAMAGUCHI, Hiroaki NODA, Nobuhiro TAKEDA,
 Tetsuya FUJIMOTO, and Takashi KAMBE
 Production Technology Research Laboratory,
 Production Technology Development Group, SHARP Corporation

3. 論理合成

本手法では自社開発の論理合成システムを採用した⁽¹⁾。論理合成システムはUDL/I記述⁽²⁾から、その動作

(1)

を実現するためのデータバス、レジスタ、および有限状態機械として実現される制御回路を合成する。合成対象は多相クロック同期式回路である。

4. システム検証

システム検証は、従来の論理シミュレータやシミュレートデバッグの他に、再書き込み可能なFPGAを用いたエミュレーションシステムを採用した^[1]。エミュレーションシステムではLSIのネットリストをシステム上にロードし、ブレッドボードと同様の環境で動作チェックができる。この検証方法では、ハード的な作業をほとんど行なうことなく、論理シミュレーションでは困難なさまざまな検証・評価をブレッドボードと同様に実速度に近い動作速度で行なうことができる。後述するLSI開発例では、エミュレーションシステムで試作した回路にLSIの入力を与えて音声を出力し、システム評価を行なった。

5. 実LSI設計への適用

今回本手法を適用したLSIは音声信号処理用であり、ソフトウェアによりプログラム可能な専用DSPである。

本開発では、まず人手設計と同じアーキテクチャで機能記述したものを本手法により論理合成し、動作検証を行なった。しかし、この回路では、当初目標としていた性能を満たすことができなかつたため、アーキテクチャレベルの大幅な設計改良・試作検証を数回にわたり行なった。

[試作1] 人手設計と同じアーキテクチャによる合成

人手設計と同じアーキテクチャでRTL記述を行ない、論理合成を行なった。

[試作2] データバスアーキテクチャの変更

各命令のデータバスについてクリティカルパス解析を行ない、性能を満たさないデータバスを分割することによって1命令の処理時間を向上させた。これによって以前は1命令で行なえた処理が2命令になるなどソフトウェアにも影響を与えたが、ソフトウェア処理量の増加は影響のない範囲に抑えた。

[試作3] パイプラインアーキテクチャの変更

分割による高速化が不可能なデータバスについては、ALUや乗算器のパイプライン化やパイプライン段数の追加を行なうことにより高速化を図った。また、命令の解釈実行部もフェッチ・デコードを分割し、パイプライン段数を増やして処理性能を向上させた。

[試作4] DSP処理構造の変更によるコスト削減

試作1のシステムでは全体の処理を1サイクル内で行なうために、RAMを介して2つのブロックに分けて処理を行っていた。試作3までの改良によりDSPの処理能力が向上したため、2つのブロックを融合し中間RAMを削除することによりコストを削減した。これは大きなシステム構造の変更であり、この統合によりDSP

のISA(Instruction Set Architecture)も変更を行なったため、それに伴うソフトウェアにも影響を及ぼしたが、この変更によってチップサイズは大きく削減された。

[試作5] データ出力部とISAの改良

出力データアクセス部とISAの改良を行うことで、さらに処理性能の向上とコスト削減を行なった。

6. トップダウン設計の効果

一連の試作におけるシステムレベルの変更は回路全体に大きな影響を及ぼすので、従来の設計手法では簡単には行なうことが出来ず、変更を行なったとしても大きなターンアラウンドタイムが必要であった。

しかし、本手法ではクリティカルパス分割やパイプライン化などのアーキテクチャの変更は機能記述を数行変更するだけで行なうことが出来る。また、ISAの変更はハードウェアとソフトウェアの両面に影響を及ぼすが、アーキテクチャレベルの記述では大きな修正にはならない。ソフトウェア開発もシミュレートデバッグを併用することにより比較的簡単に修正を行なうことができる。このため、変更箇所が決定してからエミュレーションシステムによる試作検証まで数日で行うことができた。

試作2から5はほぼ5ヵ月で試作を完了し、試作1に比べ動作速度44%向上、チップサイズ40%向上という成果が得られた。表1に動作速度、チップサイズの比較を示す。

7. まとめ

本トップダウン設計手法を用いることにより、システムレベルでの改良を短時間で繰り返し行なうことが可能になる。これによって、システムの抜本的な改良を短期間に行なうことが出来る。今回のLSI開発では本手法の採用により、最初の試作に比べて大きな性能向上を果たすことができた。

[参考文献]

- [1] 野田他, "レジスタ転送レベルからのVLSIの合成手法", DAシンポ'92 (Aug. 1992).
- [2] 星野他, "主要なハードウェア記述言語の特徴と標準化状況 UDL/I", 情報処理 Vol.33 No.11 pp.1244-1249 (Nov. 1992).
- [3] S. Walters, "Computer-Aided Prototyping for ASIC-Based Systems", IEEE Design & Test of Comp. (Jun. 1991).

表1 各試作の動作速度とゲートサイズ

	ゲート数 (gates)	ROM, RAM (byte換算)	クリティカルバス(ns)
試作1	31,326	34943.5	35.7
試作5	22,400	26512.0	20.0