設 計 自 動 化 89－13
（1998. 9. 22）

システムLSIのコンカレントデザイン環境
— 1チップMPEG-2 MP@MLエンコーダLSIへの適用 —

落合 克幸　岩崎 裕江　長沼 次郎　遠藤 真

NTT ヒューマンインターフェース研究所
〒239-0847 神奈川県 横須賀市 光の丘 1-1
電話: 0468-59-8954 / E-Mail: {kach,hiroe,jiro,endo}@nttvdt.hil.ntt.co.jp

あらまし　近年、コアCPUと専用ハードウェアを融合したシステムLSIの開発が盛んである。しかし、このようなLSIの開発では、ハードウェア開発とともに、ソフトウェア開発が期間の大きな割合を占める。この問題を解決するため、我々は、コンカレントデザイン環境を提案するとともに、この環境を実現するソフトウェアプラットホームを開発した。このプラットホームは、命令セットレベルシミュレータと機能レベルのシミュレータから構成され、各々、コアCPUと専用ハードウェアを高速にシミュレーションする。本プラットホームはハードウェア記述を必要としないため、この記述の完了を待たずに、ソフトウェアの動作検証を実現できる。本プラットホームをMPEG-2 MP@MLエンコーダLSI開発へ適用し、本手法の有効性を実証した。

キーワード　システムLSI, コンカレントデザイン, 内蔵ソフトウェア, 動作検証, シミュレーション,MPEG

Concurrent Design Environment for System LSIs
— A Single-chip MPEG-2 MP@ML Video Encoder LSI —

Katsuyuki OCHIAI,　Hiroe IWASAKI,　Jiro NAGANUMA,　and　Makoto ENDO

NTT Human Interface Laboratories
1-1, Hikarinooka Yokosuka-shi, Kanagawa, 239-0847 Japan
Telephone: +81 468 59 8954 / E-Mail: {kach,hiroe,jiro,endo}@nttvdt.hil.ntt.co.jp

Abstract

This paper proposes a new hardware and software concurrent design environment exploiting a software-platform for embedded system LSIs (system LSIs), and evaluates its benefits on a practical single-chip MPEG-2 MP@ML video encoder LSI design. The software-platform consists of an instruction set level simulator of core CPU and a function level simulator of application-specific hardware. This design environment provides sufficient performance for simulating several to several ten millions cycles of a large embedded software without hardware description.

key words　system LSI, concurrent design, embedded software, validation, simulation, MPEG

1 はじめに

近年の様々なマルチメディア関連アプリケーションをタイムリかつ低コストで提供するため、キーデバイスとなる画像処理LSIやプロトコル処理LSIなどのシステムLSI開発が盛んである[1]。システムLSIは、コアCPUと専用ハードウェアを融合することにより、ソフトウェアの柔軟性とハードウェアの高速性を兼ね備えており、大規模化、複雑化するLSI開発のトータルなソリューションを与えている。我々も、システムLSI構成法をMPEG-2チップセット[2] [3]開発に適用し、その有効性を示した。

システムLSI開発は、ハードウェア開発とともにコアCPU上のソフトウェア開発も不可欠である。ソフトウェアは、チップ上の複数の機能ブロックのリアルタイム制御が要求されるため、極めて複雑なものとなってきている。このため、ハードウェア開発とともに、ソフトウェア開発が全LSI開発期間の大きな部分を占め、その長期化が深刻な問題となっている。

これらの問題の主な原因は、ソフトウェアの動作検証環境の不十分さである。従来のソフトウェアの動作検証環境としては、(1)機能・論理シミュレータ[4]、(2)ハードウェア論理シミュレータ、(3)ASICエミュレータ[5]、(4)ハード/ソフト協調シミュレータ[6]等が存在する。しかし、これらは、ハードウェアのRTL(Register Transfer level)記述以降にしか使用できず、LSI開発の後期にソフトウェアの動作検証作業が集中するため、システムLSIとしてのハードウェア/ソフトウェアの協調した動作検証が十分に実施できないという問題が生じる。

このような問題を解決するため、RTL記述前に適用可能なコンカレントデザイン環境を提案するとともに、この環境を実現するソフトウェアプラットホームを開発した[7][8][9]。また、本プラットホームを1チップMPEG-2 MP@MLエンコーダLSI[10] [11]開発に適用し、本手法の有効性を実証した。本プラットホームでは、システムLSIの各機能を高いレベルで抽象化して表現し、シミュレーションの高速化を実現することにより、ソフトウェアの動作検証の効率化を図っている。さらに、RTL記述完了前にソフトウェアとの協調の動作検証を行うことにより、システムLSI開発期間の短縮を図っている。

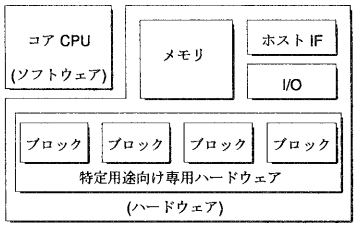

図1: システムLSIの概要

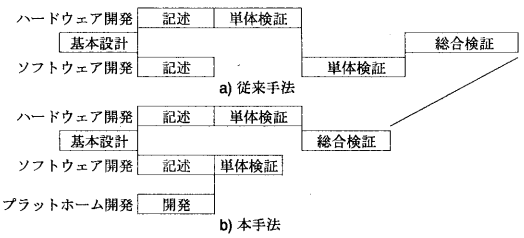

図2: コンカレントデザインのフロー

本稿では、まず第2節でシステムLSIの構成と設計フローについて述べ、コンカレントデザイン環境を示す。次に第3節でその環境を提供するソフトウェアプラットホームの詳細な構成法を示す。最後に第4節で1チップMPEG-2 MP@MLエンコーダLSI開発への適用とその評価結果を示す。

2 システムLSIのコンカレントデザイン手法

システムLSIの構成を示し、コンカレントデザインの従来手法と本手法を示す。また、ソフトウェアプラットホームの概略を示す。

2.1 システムLSIの構成

システムLSIは図1に示すように、ソフトウェアを処理するコアCPUと、メモリや演算器などの特定用途向けの専用ハードウェア機能ブロックから構成される。このシステムでは、コアCPUが専用ハードウェアの機能ブロックを統括的に制御する。また、ソフトウェアを搭載する事で、システムの柔軟性や拡張性も実現している。一方のハードウェア機能ブロックは、各々の所定の機能を高速に処理する。システムLSIでは、両者が協調してシステムの機能を実現している。

2.2 システムLSIの設計フロー

コンカレントデザインにおける従来手法と本手法の設計フローを図2に示す。ハードウェアとソフトウェアのコンカレントデザイン環境では、LSIのハードウェア設計とその上で動作するソフトウェアの開発が並行に進められる。

従来の設計では、ソフトウェアの検証に、開発がほぼ完了したRTL記述を使用する。このため、ハードウェア設計がかなり進んだ段階になるまでは、ソフトウェアの検証を進めることができない。

本手法では、RTL記述を必要としないシミュレーション環境を提供し、RTL記述完了前にソフトウェアの基本的な動作の検証を行う。これにより、ソフトウェアとハードウェアの単体検証が同時に進行し、各々の検証が進んだ段階で両者を結合し総合的な検証へと移行できる。その結果、システムLSIの開発期間を短縮できる。

2.3 ソフトウェアプラットホーム

2.3.1 開発思想

システムLSI開発の初期段階に用いる本プラットホームの目的は以下の3点である。

- ソフトウェア動作検証への適用
- 専用ハードウェアの機能ブロックの入出力期待値生成
- システムLSIアーキテクチャの検証と定量評価

ここで、ハードウェアとソフトウェアのコンカレントデザイン環境を実現するには、RTL記述完成前に、プラットホーム自身の開発を完了しなければならない。このように短期間でプラットホームを開発するため、

- コアCPU部は命令セットレベル
- 専用ハードウェア部はC言語で高い抽象レベル

でモデル化を図った。

2.3.2 システム概要

ソフトウェアプラットホームは、コアCPU部のシミュレータとC言語でモデル化された特定用途向け専用ハードウェア機能ブロックのシミュレータから構成される。コアCPU部には、既開発の命令セットレベルシミュレータ[12]を用い、ハードウェア機能ブロックとの情報の授受は通信を用いて実現する。一方のハードウェア機能ブロックのシミュレータは、高いレベルで抽象化し、機能レベルの入出力のシミュレータとして実現している。

3 ソフトウェアプラットホームの構成法

システムLSIのシミュレーションを実現する本プラットホームの構成とシステムLSIの動作モデル、実現した機能を示す。

3.1 システムLSIの動作モデリング

3.1.1 コアCPU部

コアCPU部は、命令セットレベルのシミュレータとして実現する。このシミュレータは、メモリマップトI/Oレジスタに値を書き込むことで専用ハードウェア機能ブロック部へ情報を渡し、I/Oレジスタを監視することで、機能ブロック部からの情報を得る。このシミュレータと専用ハードウェア機能ブロック部のシミュレータとは、TCP/IPによる通信を用いて接続し、共通に参照できるメモリマップトI/Oを介して、互いに同期して動作する。

3.1.2 専用ハードウェア機能ブロック

専用ハードウェア機能ブロックのモデルを図3に示す。このモデルでは、ハードウェアの構成要素を3つの要素で表す。

1. システムLSIを構成しているある機能のまとまりを抽象化し機能モジュールとしてC言語の関数やプログラムで表現する。
2. 機能ブロック間を流れるデータは、インターフェースを介して共通に参照できる変数として表す。ここでは、データへのアクセス権を制御するハンドシェークフラグやメモリマップトI/Oレジスタを介するデータも同様の変数として表現する。
3. 機能ブロック等からのデータへのアクセスは、予め決められたポインタを介した変数へのアクセスとして表し、変数への値の書き込みと読み込みによって機能ブロック間のデータ転送を実現する。

データの授受、及び関数の駆動判定には、システムLSIに実装されるハンドシェークフラグによる

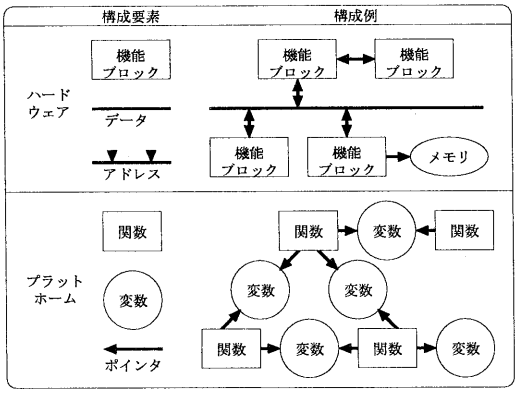

図 3: システム LSI の動作モデル

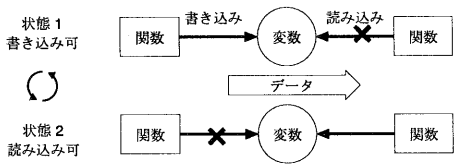

図 4: データアクセスの排他制御

排他制御を用いる (図 4)。この図で、状態 1 は書き込みが可能で読み込みが禁止の状態である。状態 2 は、逆の状態であり、結果としてデータは左から右の機能モジュールに流れる。

機能モジュールの駆動は、データの授受と同様に、ハンドシェークフラグの状態によって決定される。各々の機能モジュールが、このフラグの状態に基づいて局所的な依存関係で動作し、結果として全体の同期と順序性が維持される。本モデルでは、プラットホーム全体の制御を意識する必要は無い。

3.2 システムの構成

プラットホームの全体構成を図 5に示す。プラットホームは、データ管理部と複数の関数モジュールやプログラムモジュール、および両者間のインターフェースで構成される。

このように機能をモジュール化し、それらを統一したインターフェースで接続することにより、システムの拡張性とモジュール間を流れるデータの監視性を高める。また、関数モジュールはハードウェア機能ブロックの入出力に対する関数として扱い、関数の内部表現をハードウェア構成とは独立に設計することで高速化を図る。コア CPU 部には、既開発の命令セットレベルのシミュレーションモジュール

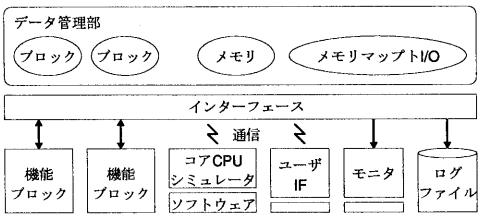

図 5: プラットホームの構成

を用いる。

これらのモジュールは、排他制御により同期して動作し、共通のインタフェースを介してデータにアクセスする。データへのアクセスは、履歴としてファイルに記録され、モジュール間を流れるデータは、モニタモジュールによりウィンドウ上にリアルタイムで表示される。このモニタモジュールも他のモジュールと同様な仕組みで動作し、必要に応じてデータにアクセスする。さらに、関数モジュールの内部状態をモニタする機能を付加しプラットホーム自身の開発を効率化している。これらのプログラムは、C/C++ 言語で記述され、汎用ワークステーション上で動作する。

3.3 実現機能

3.3.1 デバッグ / モニタ機能

本プラットホームで実現するデバック / モニタ機能の概要を図 6に示す。図中の命令セットレベルシミュレータでは、会話型のユーザインターフェースを介してコア CPU を操作し、ソフトウェアの起動や停止などの制御、あるいはレジスタやスタックなどの内部状態の設定や観察を行うことができる。また、機能ブロックシミュレータでは、メモリマップト I/O や機能ブロック間のデータを監視し、ユーザインターフェースを介した操作や、リアルタイムで表示される情報を用いてソフトウェアの解析とデバッグを行う。

3.3.2 入出力期待値生成機能

プラットホームは、図 7に示すように、各機能モジュール間の入出力データをファイルに出力する。このファイルに出力された情報は、各機能ブロックの RTL 記述のシミュレーション結果との比較による機能検証に用いられる。

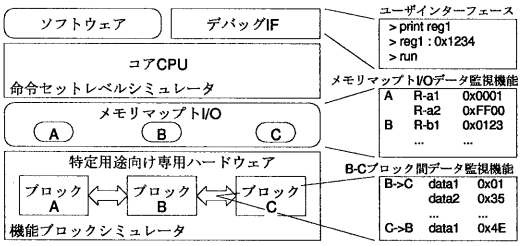

図 6: デバッグ / モニタ機能

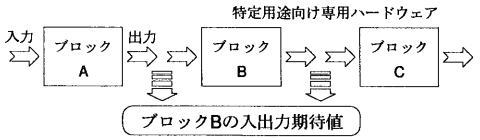

図 7: 入出力期待値生成機能

4 評価

本プラットホームを MPEG-2 MP@ML エンコーダ LSI へ適用し、その機能と性能について評価する。

4.1 適用評価

4.1.1 対象 LSI の構成

適用の対象としたシステム LSI は、図 8に示す映像符号化用途の MPEG-2 MP@ML エンコーダ LSI [10] [11] であり、デジタル放送などの通信系システムや DVD などの蓄積系システムに使用される。その諸元を表 1に示す。

このシステム LSI は、MPEG-2 に特有な動き補償用のサーチエンジン部や SIMD プロセッサ部、DCTQ/IDCTQ・Q/IQ 部、可変長符号化部 (VLC) などの特定用途向け専用ハードウェア機能ブロックとこれらを制御するコア CPU、メモリなどから構成される。ソフトウェアを処理するコア CPU は、メモリマップト I/O レジスタを介してハードウェア機能ブロックを制御し、画像符号化単位であるマクロブロックレベルで全体が同期して動作する。ソフトウェアは、C 言語で開発し、コンパイルしたコードをシステム LSI に実装する。ソフトウェアの規模は約 10k line(C 言語) である。

4.1.2 ソフトウェア開発デバッグ

プラットホームのデバッグ / モニタ機能を用いて、ソフトウェアの動作検証を行った。ワークステーション上でのプラットホームの実行状況を図 9に示す。図中左下は、命令セットレベルシミュレータであり、コア CPU の動きやメモリマップト I/O レジスタへのアクセスを操作する。また、図中左上は、メモリマップト I/O レジスタの値を示している。さらに、図中右上は各ハードウェア機能ブロックの動作状態を表し、図中右下は、機能ブロック間を流れるデータを視覚的に表示している。これらの複合的な情報を活用し、ソフトウェアのデバッグを効率的に進め、初期の段階で多くのバグを修正することができた。

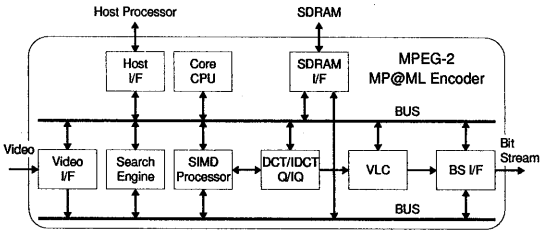

図 8: 対象 LSI の構成

表 1: 対象 LSI の諸元

Technology :	0.25μm 4-level metal CMOS
Gates :	5.0 million transistors
Area size :	9.8×9.8 mm^2 die size
Clock :	81MHz
Power :	1.5W@2.5V/3.3V
Package :	208-pin QFP

4.1.3 機能ブロックの期待値作成

プラットホームの動作の中から、各機能ブロックの入出力データを自動的に抽出し、RTL 記述のシミュレーション結果との比較による検証を行った。

例えば、DCT(Discrete Cosine Transform) や量子化演算等を行う DTC/Q 機能ブロックへの適用では、多くの比較検証を行うことで、精度が求められる演算機能の確認を行った。また、可変長符号を生成する VLC 機能ブロックは、入力データや制御パラメータの組合せが多く、機能の検証が複雑であるが、期待値を用いて多くの入出力の比較検証を行った。

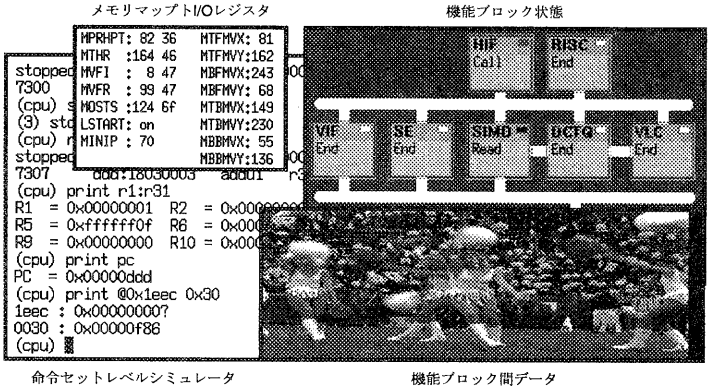

図 9: 実行状況

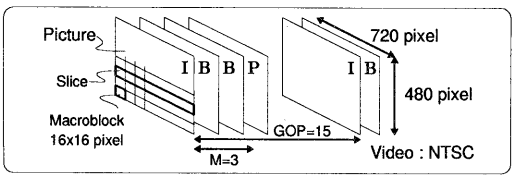

図 10: MPEG-2 の符号化単位

表 2: 処理時間

画像符号化単位	CPU 時間	画像: NTSC
マクロブロック	0.06 sec	(720 × 480 画素)
スライス	2.8 sec	GOP 構造:M=3,N=15
ピクチャ	85 sec	計算機: Sun Ultra2
GOP	21 min	(200MHz × 2CPU)

4.2 性能評価

ソフトウェアの動作検証は、図10に示す符号化単位でマクロブロック単位からGOP単位まで、より大きな符号化処理へと進められる。数千万サイクルのシュミレーションにより、システムLSIの基本的な動作を確認した。

各画像符号化単位の処理に要する実行時間を表 2に示す。マクロブロックやスライス単位の処理に要する時間は数秒以内であり、処理時間を気にせずにメモリマップトI/Oレジスタなどの値の確認や操作ができる。このため、会話型のユーザインターフェースを介して動作検証が快適に進められた。

また、ピクチャやGOP単位の処理に要する時間は20分程度である。ファイルに出力されたログファイルを元に、ビットストリーム解析やレート制御などのピクチャ間の検証も効率良く実現できた。全体として、デバッグの短TAT化が図れており、特に、バグを多く含む設計初期のソフトウェア動作検証への適用は効果的であった。

4.3 考察

4.3.1 実現機能

本プラットホームで実現したデバッグ / モニタ機能により、メモリマップトI/Oや機能ブロック間のデータ、機能ブロックの状態が観察、解析できた。また、ウィンドウ上に視覚的に表示される情報は、ユーザに直観的に伝わるため、バグの所在の特定に効果的であった。これらの情報を用いる事で、ソフトウェアのバグを効率的に修正できた。また、機能ブロックの期待値生成機能を用いたハードウェア記述の検証では、多くの入出力データの組合せをテストすることで、記述の完成度が高められた。

4.3.2 実行速度

本プラットホームの処理速度は、RTL記述完了後に使用できる市販のコンパイル型ソフトウェア機能 / 論理シミュレータの数百倍以上である (図11)。また、FPGAベースの市販ASICエミュレータと比較して1/10程度である。この処理速度は、設計の初期段階で使用できるソフトウェア動作検証用のシミュレータとしては極めて高速である。

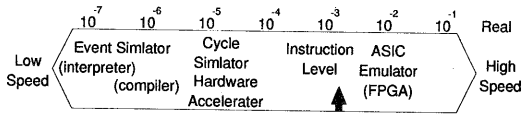

図 11: 従来の検証系との速度比較

4.3.3 設計の短 TAT 化

本プラットホームの適用により、ソフトウェアの動作検証の効率化が図れ、ハードウェア / ソフトウェア設計の初期段階から RTL 記述完了を待たずにソフトウェアの動作検証が始められた。このため、両者を結合した総合検証へと円滑に移行でき、ハードウェアとソフトウェアから構成されるシステム LSI 開発期間の短縮が図れた。

5 おわりに

コア CPU と専用ハードウェアを融合したシステム LSI のソフトウェア開発の効率化をターゲットとして、RTL 記述完了前に適用可能なコンカレントデザイン環境を提案するとともに、この環境を実現するソフトウェアプラットホームを開発した。本ソフトウェアプラットホームを、1 チップ MPEG-2 MP@ML ビデオエンコーダ LSI 開発のハードウェア / ソフトウェア切り分け後の RTL 記述完了前のソフトウェア開発に適用し、本手法の有効性を実証した。

本ソフトウェアプラットホームでは、システム LSI の各機能を高いレベルで抽象化して表現し、シミュレーションの高速化を実現することにより、ソフトウェアの動作検証の効率化を図った。また、ここで採用した動作モデリングは、本プラットホーム開発自体の効率化と短 TAT 化も併せて達成している。さらに、本プラットホームは RTL 記述を必要としないため、RTL 記述完了前にソフトウェアとの協調の動作検証を実現することにより、システム LSI の開発期間の短縮を実現できる。

このように、本ソフトウェアプラットホームをシステム LSI のコンカレントデザインに適用することにより、システム LSI のソフトウェアのデバッグ期間が短縮でき、システム LSI 開発の短 TAT 化が図れた。今後、機能ブロック間のインターフェース等の汎用化と規格化を図り、本手法を他のシステム LSI の開発にも適用していく。また、各機能ブロックのシミュレータをネットワーク上に分散させ、さらなる処理の高速化と高機能化を図っていく予定である。

謝辞

本研究を進めるにあたり、終始御指導、御助言を頂いた NTT ヒューマンインタフェース研究所 画像通信研究部 一之瀬 進 部長、ならびに NTT 入出力システム研究所 第一プロジェクト 笠井 良太 プロジェクトリーダに感謝致します。また、種々の有益な御討論を頂いた 小倉 武 グループリーダ、ならびに研究グループの方々に感謝致します。

参考文献

[1] "第 1 回 システム LSI 琵琶湖ワークショップ," Nov. 1997.

[2] T. Kondo, K. Suguri, M. Ikeda, T. Abe, H. Matsuda, T. Okubo, K. Ogura, T. Tashiro, N. Ono, T. Minami, R. Kusaba, T. Ikenaga, N. Shibata, R. Kasai, K. Otsu, F. Nakagawa, and Y. Sato, " Two-Chip MPEG2 Video Encoder ," IEEE Micro, pp.51-58, April 1996.

[3] M. Inamori, J. Naganuma, and M. Endo, "A Memory-based Architecture for MPEG2 System Protocol LSIs," ED&TC 96, Mar. 1996.

[4] "Chronologic VCS Reference Manual (Release 4.1)," Synopsys, Inc. Mar. 1998.

[5] "System Realizer Reference Manual (Version 5.1)," Quickturn Design Systems, Inc., Dec. 1997.

[6] "Seamless CVE Reference Manual (Release 2.4) ,"Mentor Graphics Corporation, Jul. 1998.

[7] 長沼, 岩崎, 落合, 遠藤, "エンベデッドシステム LSI のコンカレントデザイン環境 — 開発の目的とソフトウェアプラットホーム," 春季信学全大,Mar.1998.

[8] 落合, 岩崎, 長沼, 遠藤, "エンベデッドシステム LSI のコンカレントデザイン環境 — ソフトウェアプラットホームの構成," 春季信学全大,Mar.1998.

[9] 岩崎, 落合, 長沼, 遠藤, “エンベデッドシステムLSIのコンカレントデザイン環境— 実現機能と適用例,” 春季信学全大, Mar. 1998.

[10] T. Minami, T. Kondo, K. Nitta, S. Suguri, M. Ikeda, T. Yoshitome, H. Watanabe, H. Iwasaki, K. Ochiai, J. Naganuma, M. Endo, E. Yamagishi, T. Takahashi, K. Tadaishi, Y. Tashiro, N. Kobayashi, T. Okubo, T. Ogura, and R. Kasai, “A Single-Chip MPEG2 MP@ML Video Encoder LSI with Multi-chip Configuration for a Single-board MP@ML Encoder ,” Hot Chips 10, Aug. 1998.

[11] 近藤, 南, 新田, 村主, 池田, 岩崎, 落合, 渡辺, “HDTVへの拡張が可能なMPEG2映像符号化LSI ,” 信学集積回路研究会, Aug. 1998.

[12] 遠藤, 長沼, “ MPEG2システム多重 / 分離LSIのファームウェア”, 春季信学全大, Mar. 1996.