

画像処理向けシステム LSI 設計における 設計ナビゲーションを考慮した HW/SW 分割システム

小島 洋平[†] 戸川 望[†] 橘 昌良[‡] 柳澤 政生[†] 大附 辰夫[†]

[†] 早稲田大学理工学部コンピュータ・ネットワーク工学科

[‡] 高知工科大学工学部電子・光システム工学科

本稿では、(1)IP 再利用、(2) 定性指標、(3) 設計ナビゲーションを考慮した、画像処理向けシステム LSI を対象とする HW/SW 分割システムを提案する。(1) により設計期間が短縮し、(2) により設計/テストコストの低い解を選ぶことができる。また、(3) により改良工程に取り掛かる設計者をサポートすることができる。これらの有効性を計算機実験によって確認した。

A hardware/software partitioning system with design navigation for a system LSI design

Yohei KOJIMA[†], Nozomu TOGAWA[†], Masayoshi TACHIBANA[‡],
Masao YANAGISAWA[†], and Tatsuo OHTSUKI[†].

[†] Department of Computer Science, Waseda University.

[‡] Dept. of Electronic and Photonic Systems Engineering, Kouchi University of Technology.

In this paper, we propose a new hardware/software partitioning system based on (1) IP reuse, (2) a qualitative objective evaluation and (3) design navigation. Design period of system LSIs is shortened by taking (1) into consideration. low design cost or low test cost designs can be realized by introducing (2). (3) helps the designer improve his/her design. We confirmed the effectiveness of the proposed system through computer experiments.

1 まえがき

システム LSI とは、従来では別個のパッケージで作られていた LSI を、1 つのチップ上に集積したものであり、現在プロセッサも集積可能である。したがって、システム LSI 設計においては、ハードウェア (HW) 部分とソフトウェア (SW) 部分の両方を設計する必要がある。

システム LSI における HW 部分と SW 部分を同時に設計し、最適化を図ることで、個別に設計するよりも短期間かつ低コストで設計する手法として HW/SW 協調設計がある。

HW/SW 協調設計にはこれまで様々な手法 [1, 2, 3, 4, 5, 6, 9] が提案されている。しかし、これらの手法は、分割時の評価として動作速度や面積、消費電力など定量的な数値のみを注目しており、「テストのしやすさ」などのいわゆる定性的な側面については深い考察がなされていない。

また、一般に、システム LSI の大部分を新規設計とすることは設計期間、コストの面において困難である。したがって、IP (設計資産) を用いて、設計期間の短縮や低コスト化を目指すことが重要となる。

そこで本稿では、IP 再利用、定性指標を考慮した HW/SW 分割システムによる協調設計手法を提案する。本システムでは IP の再利用として、既存の IP を元に「処理の並列度」や「処理の反復性」を考慮し、新たな IP を仮想的に類推し、列挙する類推列挙機構を導入している。類推列挙により、仮想的に面積/処理時間制約を満たす設計を増やすことができる。定性指標としては「設計のしやすさ」と「テストのしやすさ」の 2 項目を導入する。定性指標

を導入することにより、面積/処理時間制約を満たす中で、特に定性的に優れた設計を出力することが可能になる。

また、多くの既存手法では制約を満たす解が存在しない場合について言及していない。そこで、本手法では解がない場合のサポート機能として設計ナビゲーションを導入する。設計ナビゲーションは改良工程に取り掛かる設計者に改良の目安となる指標を提示することでサポートする。

2 HW/SW 分割システム

本節では、提案する HW/SW 分割システムの概要を説明する。以下、2.1 節に本稿で使用する用語、2.2 節にシステムモデル、2.3 節にシステム構成を示す。

2.1 用語の定義

機能モジュール 色空間変換や DCT といった、ひとまとまりの処理を示す。

機能ブロック図 機能モジュールをノードとし、機能モジュール間のデータの流れをエッジとした、データフローグラフである。JPEG エンコーダを例とした機能ブロック図を図 1 に示す。

アルゴリズム 機能モジュールで実行される処理を表す。

アーキテクチャ アルゴリズムを実装する手段であり、HW/SW の区別 (HW であれば並列度も示す) を表す。アーキテクチャは、面積、処理時間および定性指標といった属性を持つ。

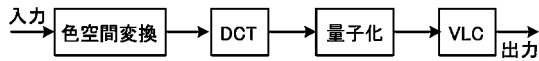


図 1: JPEG エンコーダの機能ブロック図

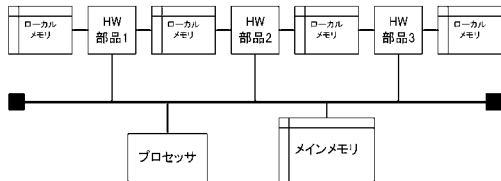
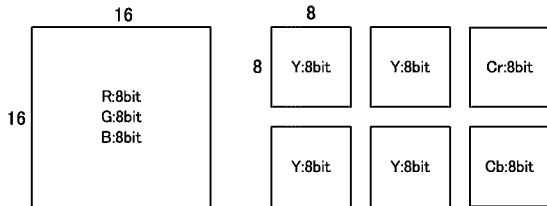


図 2: システムモデル



(a)色空間変換の基本ブロック (b)色空間変換を除くアルゴリズムの基本ブロック

図 3: 基本ブロック

割り付け アルゴリズムのアーキテクチャを決定する操作を表す。

2.2 システムモデル

本システムが対象とするシステムモデルを図 2 に示す。このモデルは以下の特徴をもつ。

- HW モジュール同士は、共有メモリを介して通信する。
- 2 つ以上の SW モジュールは同時に実行不可能である。
- HW-SW 間の通信にはバスを使用する。

また、ターゲットアプリケーションは JPEG, JPEG2000, MPEG-4 エンコーダとし、処理の単位を図 3 に示す基本ブロックとする。図 3 のように、色空間変換前の基本ブロックは RGB 信号 8×8 画素 $\times 4$ ブロックであり、色空間変換後は色差信号 4 画素分を 1 画素にまとめるため輝度信号 8×8 画素 $\times 4$ ブロック、色差信号 8×8 画素 $\times 2$ ブロックとなる。HW-SW 間の通信をする場合、1 基本ブロック分のデータを転送するために以下に示す時間を要する。

$$T_{\text{bus}} = \frac{1 \text{ 基本ブロック分のデータ量}}{\text{バスクロック} \times \text{バスビット幅}} \quad (1)$$

2.3 システム構成

本システムの概要図を図 4 に示す。本システムは、アーキテクチャデータベース、実現可能割り付け系、設計ナビゲーションから構成されている。

アーキテクチャデータベースにはアーキテクチャを蓄積しており、蓄積されたアーキテクチャは IP として再利用可能である。また、蓄積されたアーキテクチャを元にして仮想的なアーキテクチャを類推する類推列挙機構を導入している。これにより解の探索空間を広げることが可能となる。

実現可能割り付け系では、入力されたアプリケーションの機能ブロック図の各アルゴリズムに、デー

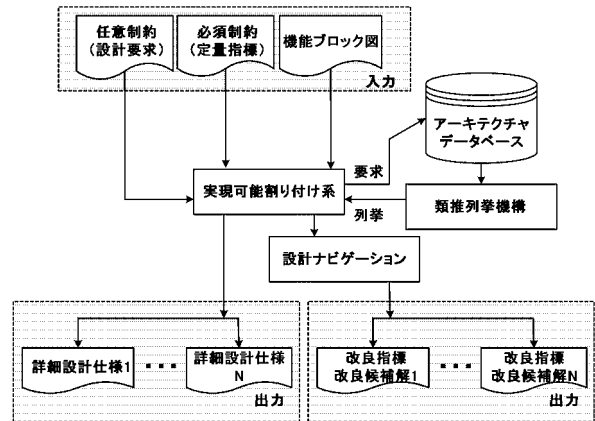


図 4: HW/SW 分割システム

タベースから列挙されたアーキテクチャを割り付ける。この割り付け結果を解候補と呼び、特に入力された制約条件 (面積/処理時間) を満たすものを実現可能解と呼ぶ。さらに定性的な設計要求が入力された場合には、実現可能解の定性指標 (設計/テストのしやすさ) を評価し、要求を満たすものを詳細設計仕様として出力する。なお、以降設計要求という単語は定性的な設計要求という意味で用いる。また、面積/処理時間制約は必須入力であり、設計要求は任意入力である。

設計ナビゲーションは、実現可能解が存在しない場合に、改良を試みる設計者をサポートする機能である。制約オーバーの小さい解候補を改良候補解として入力し、これを改良するための指標を出力する。

3 アーキテクチャデータベース

アーキテクチャデータベースは、入力アプリケーションを構成するアルゴリズムを入力とし、入力されたアルゴリズムのアーキテクチャを全て出力する。

LSI を設計する際に、アーキテクチャデータベース中に、必要となるアルゴリズムのアーキテクチャがあれば再利用可能であり、なければ新たに用意する必要がある。アーキテクチャデータベースを利用した設計を繰り返すことにより、データベースが充実し、HW/SW 分割システムの完成度が向上する。

以下、3.1 節にアーキテクチャの構成、3.2 節に類推列挙機構、3.3 節に定性指標を示す。

3.1 アーキテクチャの構成

各アーキテクチャは、面積、処理時間および定性指標を持つ。ただし、処理時間は 1 基本ブロック分の処理に要する時間とする。

アーキテクチャが HW の場合、面積および処理時間は VHDL で実装した結果を用い、論理合成には synopsys 社の Design Compiler および日立製作所の $0.35[\mu\text{m}]$ セルライブラリを用いる。

アーキテクチャが SW の場合、面積は ROM と RAM の面積の和とする。ROM の面積は $600[\mu\text{m}^2/\text{byte}]$ に ROM 使用容量 [byte] を掛け合わせて算出し、RAM の面積は $820[\mu\text{m}^2/\text{byte}]$ に RAM 使用容量 [byte] を掛け合わせて算出する。ここで、1 バイトあたりの ROM, RAM の面積は、そ

表 1: 演算命令種類別の平均 CPI 値

演算	平均 CPI 値
加減・論理演算	1
乗算	1
除算	8
ジャンプ	1.2
分岐	1.8
メモリアクセス	1.4

れぞれローム 0.35[μm] プロセスの ROM (面積: 38300[μm^2], 容量: 64[byte]) および SRAM (面積: 33000[μm^2], 容量: 40[byte]) を元に算出した。また, 処理時間 T_{software} [s] は, 式 (2) に示すように, プログラムの実行クロック数 N_{clock} とプロセッサの動作周波数 f [Hz] から算出する。ここで, 実行クロック数は, アセンブリ言語の命令種別ごとの数と, 表 1 に示した値との積とする。

$$T_{\text{software}} = \frac{N_{\text{clock}}}{f} \quad (2)$$

使用するプロセッサは TX39 (東芝製) であると仮定する。TX39 のアーキテクチャは MIPS 製 R3000A と同じであり, 面積は 8280000[μm^2] である。この値は, 文献 [7] および [8] からプロセッサの等価ゲート数を計算し, これと VDEC 提供の日立製作所 0.35[μm] セルライブラリの 2 入力 NAND の面積を用いて換算した値である。本システムのプロセッサ動作周波数は 10[MHz] であると仮定する。

3.2 類推列挙機構

本システムでは, アーキテクチャを再利用する方法として, 既存のアーキテクチャを元に, 並列度などの異なるアーキテクチャの面積や処理時間を類推し, 列挙する類推列挙機構を導入した。本稿では, 以下の 2 種類の類推手法を適用した。

- 処理の並列度を変化させ, 面積, 時間を類推。
- 処理の反復回数を変化させ, 面積, 時間を類推。

並列度を変化させる場合, アルゴリズム毎に類推方法が異なる。例えば, 色空間変換, 量子化, 逆量子化などのアルゴリズムは, 画素単位での処理であるため, 画素単位で並列度を変化させることが可能である。実現可能な並列度は基本ブロックに含まれる画素数の約数である。つまり, 量子化, 逆量子化の場合は $8 \times 8 \times 6 = 384$ 画素の約数 $\{1, 2, 3, 4, 6, 8, 12, 16, 24, 32, 48, 64, 96, 128, 192, 384\}$ となる。色空間変換の場合は 4 画素をまとめて処理するため, $8 \times 8 \times 4 = 256$ 画素の約数かつ 4 の倍数である $\{4, 8, 16, 32, 64, 128, 256\}$ となる。また, DCT, 逆 DCT, VLC などのアルゴリズムは 8×8 画素のブロック単位の処理であるため, ブロック単位で並列度を変化させる必要がある。実現可能な並列度は基本ブロックに含まれる 8×8 画素のブロック数の約数 $\{1, 2, 3, 6\}$ である。ここで, 上記のアルゴリズムでは, 並列度を N 倍にした時, 面積は N 倍, 処理時間は $1/N$ 倍に変化するものとする。

これに対し, 動き予測処理の場合, 処理の並列度に対して面積/処理時間が線形に変化しない。動き予

表 2: 動き予測器における類推

予測手法	探索点数	処理時間	面積
完全探索手法 (実 IP)	1089	1	1
3 段階層探索手法	33	0.0303	1
2 次元対数手法	57	0.0523	1
one-at-a-time 手法	64	0.0588	1
1 次元完全探索手法	74	0.0680	1

表 3: 類推列挙適用前後でのアーキテクチャ数

アルゴリズム	種類 (類推前)	種類 (類推後)
色空間変換	2	17
DCT	5	11
量子化	4	34
逆量子化	4	34
逆 DCT	5	11
可変長符号化	3	9
動き予測	4	20

測は動画像におけるあるブロックの動きベクトルを検出する処理であるが, 探索範囲内でブロックマッチングを繰り返すことで実現している。したがって, 動き予測では処理時間は探索点数にはほぼ比例すると考えられるため, 処理の反復回数を変化させる類推手法を適用する。現在, アーキテクチャデータベースに存在する動き予測器は, 探索範囲を 48×48 画素とした完全探索手法であるが, 探索点数を間引くことにより処理を高速化する以下のような手法が存在する。

- 3 段階層探索手法
- 2 次元対数手法
- one-at-a-time 手法
- 1 次元完全探索手法

上記の各手法が全探索手法と異なる点は, 探索点数および探索箇所のみである。探索点数に比例して処理時間は変化すると考えられる。また, 探索箇所が変化した場合, レジスタ, 比較器および加算器等の面積オーバーヘッドが生じるが, 本稿ではこれらのオーバーヘッドを無視し, 上記の各手法の面積も全探索手法と等しいものとする。

以上の議論をもとに, 探索範囲を 48×48 画素とした完全探索手法から, 上記の 4 種類の探索手法の処理時間および面積を類推し, 完全探索手法に対する比率を表 2 に示す。また, 表 3 に類推列挙を適用前後での, 各アルゴリズムのアーキテクチャ数を示す。

3.3 定性指標

各アーキテクチャは, 面積や処理時間といった定量指標に加え, 定性指標を持つ。定性指標を考慮することにより, 後段の詳細設計やテストが容易な設計を抜粋することが可能となる。

本稿では, 定性指標として「設計のしやすさ」および「テストのしやすさ」の 2 種類を取り扱う。

3.3.1 設計のしやすさ

「設計のしやすさ」では, 類推 IP と比較して実 IP の方が設計しやすいという考えに基づき, 表 4 に示すような 2 段階の順序付けを設定した。

表 4: 定性指標 (設計のしやすさ)

区別	定性指標
実 IP	優
類推 IP	良

表 5: 定性指標 (テストのしやすさ)

区別	定性指標値
SW	優
HW(BIST あり)	良
HW(BIST なし)	可

3.3.2 テストのしやすさ

「テストのしやすさ」については以下の点に着目し、表 5 に示すような 3 段階の順序付けを設定した。

1. SW モジュールと HW モジュールを比較すると、HW モジュールの方が幅広いテスト環境と多くのテストデータが必要である。
2. HW モジュールの場合、BIST を組み込むことにより、少ない HW コストで HW テストを容易化することができる。

4 実現可能割り付け系

実現可能割り付け系では、実現可能割り付けにより解候補を作成し、続いて解候補が面積/処理時間制約を満たしているか定量指標評価を行う。設計要求が入力された場合には、最後に定性指標評価を行い、詳細設計仕様または設計ナビゲーションへの入力となる改良候補解を出力する。

以下、4.1 節に実現可能割り付け、4.2 節に定量指標評価、4.3 節に定性指標評価について示す。

4.1 実現可能割り付け

実現可能割り付けでは、機能ブロック図のアルゴリズムにデータベースから列挙されるアーキテクチャを割り付けることにより、解候補を作成する。その際、図 5 に示すような、深さをアルゴリズム、ある深さの節点をそのアルゴリズムを実装するアーキテクチャとした木構造を生成する。この木構造において、同一の深さの同一の番号の節点は同一のアーキテクチャを表している。この木構造を各アルゴリズムにおいて処理時間の小さいアーキテクチャから順に深さ優先探索し、根から葉までの経路に含まれるアーキテクチャの組み合わせが解候補となる。

解候補を作成したら定量指標評価を行い、次の解候補を探索する。各アルゴリズムにおいて最も処理時間の大きいアーキテクチャの組み合わせを探索したら処理は終了となる。なお、探索の際に分枝限定法を用い、あらかじめ制約を満たさないと判断できるものは探索しないことにより、探索時間を短縮している。

4.2 定量指標評価

定量指標評価では、解候補の面積および画像 1 フレーム分の処理時間を算出し、面積/処理時間制約を満たしているか評価する。

解候補の面積は、解候補における各アーキテクチャの面積と、プロセッサの面積の総和である。なお、解候補が全て HW である場合も、プロセッサは必要で

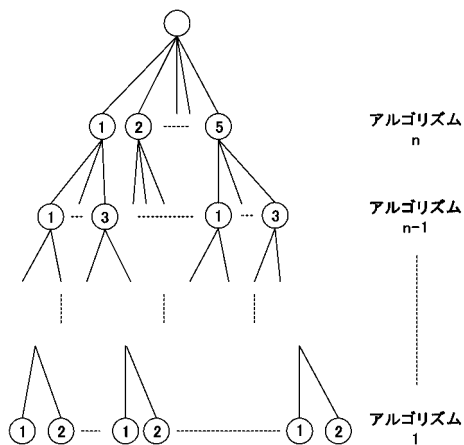


図 5: 木構造

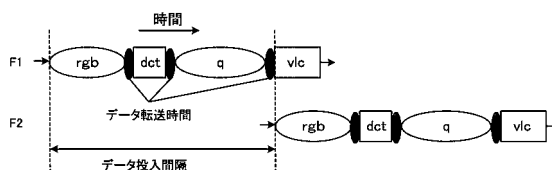


図 6: データの投入間隔

あるものとする。処理時間に関しては、本システムでは 1 フレームに含まれる N 個の基本ブロックをパイプラインでエンコード処理する。したがって、1 フレーム分の処理時間 $T_{\text{frame}}[\text{s}]$ は、解候補を構成するアーキテクチャの処理時間の和 $T_{\text{all}}[\text{s}]$ 、1 イタレーションにおけるデータの転送回数 M 、式 (1) で得られるバスのデータ転送時間 $T_{\text{bus}}[\text{s}]$ およびデータの投入間隔 T_{in} を用いて、式 (3) のように算出される。

$$T_{\text{frame}} = (N - 1) \cdot T_{\text{in}} + T_{\text{all}} + M \cdot T_{\text{bus}} \quad (3)$$

データ投入間隔は、同一のアルゴリズムまたは SW 同士の処理が重ならないよう各イタレーションをずらししてスケジューリングすることで求められる。JPEG エンコーダを例としたデータの投入間隔を図 6 に示す。ただし、図 6 において、方形の機能モジュールを HW モジュール、円形の機能モジュールを SW モジュールとする。

入力された面積/処理時間制約を解候補が満たしている場合は、実現可能解として出力する。制約を満たさない場合でも、制約オーバーの小さい解候補は改良候補解として定数個保持しておき、最終的に実現可能解が存在しない場合に設計ナビゲーションへの入力として出力する。ただし、制約オーバーの大きさを表す値には、解候補の面積/処理時間がそれぞれの制約をオーバーした分の制約に対する割合の和を用いる。

4.3 定性指標評価

設計要求が入力された場合には、実現可能解または改良候補解に対して定性指標評価を行う。

設計要求は、「DCT の設計のしやすさ：優」、「量子化のテストのしやすさ：良」というようにアルゴリズムごとに与えられるものとし、これを満たす定

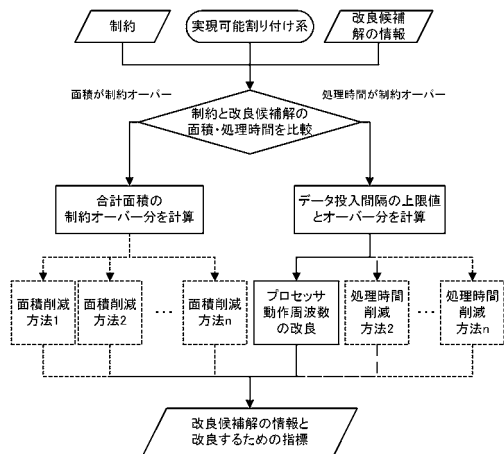


図 7: 設計ナビゲーションフロー

性指標を持つ解を詳細設計仕様または設計ナビゲーションの入力として出力する。設計要求を満たす解がない場合には、設計者が各定性指標項目の優先順位や要求を与えた評価関数を設定し、評価値の高いものを出力する。

5 設計ナビゲーション

実現可能割り付けの結果、制約を満たす実現可能解が得られなかった場合、設計者は改良を施すかどうか判断すると思われる。その判断の目安となること、および改良工程に取り掛かる設計者をサポートすることを目的とするのが設計ナビゲーションである。

設計ナビゲーションへの入力には改良コストが小さいと考えられる改良候補解と制約条件であり、出力は改良候補解を改良するための指標である。現在導入している指標は以下の3項目である。

- 合計面積の制約オーバー
- データ投入間隔の上限値・上限値オーバー
- プロセッサ動作周波数の変化量に対する処理時間の変化量

上2項目はどれ程改良する必要があるのかという指標であり、3つ目の項目はパラメータをどのように改良すれば良いかという指標である。設計ナビゲーションのフロー図を図7に示す。図7のように、まず制約をどれ程オーバーしているのかを求め、続いてそのオーバーしている量をパラメータをどのように改良すれば削減できるのかを求める。ここで、複数のパラメータの改良を考慮できるようになることが理想的である。

1つ目の指標である合計面積の制約オーバーの値は、改良候補解の面積と面積制約の差で求められる。以下残りの指標について、5.1節にデータ投入間隔の上限値、5.2節にプロセッサ動作周波数の変化量に対する処理時間の変化量の求め方を示す。

5.1 データ投入間隔の上限値

本システムでは画像1フレームを基本ブロックを単位にパイプライン処理するため、改良候補解の処理時間 T_{frame} は式(3)で表されるようにデータの

投入間隔 T_{in} に大きく依存する。そのため、処理時間の制約オーバーを示すよりも、データ投入間隔の上限値を求めそれをオーバーしている量を示したほうが、ボトルネックがわかりやすく改良する目安となると考えられる。

データ投入間隔の上限値 T_{inmax} は、式(3)の T_{frame} に処理時間制約 $T_{constraint}$ を代入し、逆算することで求められる。

$$T_{inmax} = \frac{T_{constraint} - (T_{all} + M \cdot T_{bus})}{N - 1} \quad (4)$$

5.2 プロセッサ動作周波数の変化量に対する処理時間の変化量

SWモジュールの処理時間を削減する一方法として、プロセッサの動作周波数の改良が挙げられる。そこで、改良候補解にSWモジュールが含まれる場合には、動作周波数をパラメータとしてどのように改良したら改良候補解全体の処理時間がどれ程改善するのかという値を出力する。

改良候補解に含まれる全SWモジュールの合計処理時間 T_{SW} は、全SWプログラムの合計実行クロック数 N_{clk} とプロセッサの動作周波数 f を用いて次式のように算出される。

$$T_{SW} = \frac{N_{clk}}{f} \quad (5)$$

したがって、プロセッサ動作周波数の変化量 Δf に対する処理時間の変化量 ΔT は、次式で表せる。

$$\begin{aligned} \Delta T &= \frac{N_{clk}}{f + \Delta f} - \frac{N_{clk}}{f} = \frac{f \cdot T_{SW}}{f + \Delta f} - \frac{f \cdot T_{SW}}{f} \\ &= -\frac{\Delta f}{f + \Delta f} \cdot T_{SW} \end{aligned} \quad (6)$$

6 計算機実験結果

提案システムを、C++言語を用いて計算機上に実装した。例題アプリケーションとしてMPEG-4エンコーダ(176×144画素)およびJPEGエンコーダ(2048×1536画素)を適用した。ここで、本システムでは、両者の処理内容が類似している点に着目して、まずMPEG4エンコーダを設計し、さらにMPEG4エンコーダのIPを再利用することによりJPEGエンコーダの設計期間を短縮することを実現した。また、バスアーキテクチャは、バスクロック:10[MHz]、バスのビット幅:128[bit]とし、アーキテクチャデータベースは表3に示したものを適用した。

まず、制約を満たす実現可能解の分布を調べた。JPEGエンコーダを入力アプリケーションに、面積制約:15[mm²]、処理時間制約:30[s]を与えた場合の実現可能解のトレードオフ曲線を図8に示す。図8において、実線のグラフは実IPのみを適用した結果であり、破線のグラフは類推IPも含めて適用した結果である。これより、類推列举機構を導入することで、処理時間が同等、かつ、約20%小面積の設計を出力できることを確認した。

次に、制約を満たす実現可能解が得られない場合に、改良候補解100個の分布を調べた結果を図9に示す。図9において、「+」は面積制約30[mm²]、時

表 6: 定性指標評価

アプリケーション	制約 (処理時間 [s]/面積 [mm ²])	実現可能解数	設計要求 (定性指標)	詳細設計仕様数
JPEG	300/25	34029	rgb_des:優, dct_des:優, q_test:優	252
JPEG	30/12	6591	vlc_des:優, q_test:優	136
JPEG	1/10	444	vlc_des:優	136
MPEG-4	1.5/12	20506	idct_des:優, iq_des:優, vlc_test:優	432
MPEG-4	1/12	10130	rgb_test:優, q_test:優, vlc_des:優	336
MPEG-4	0.4/13	28328	rgb_des:優, q_des:優, iq_des:優	48

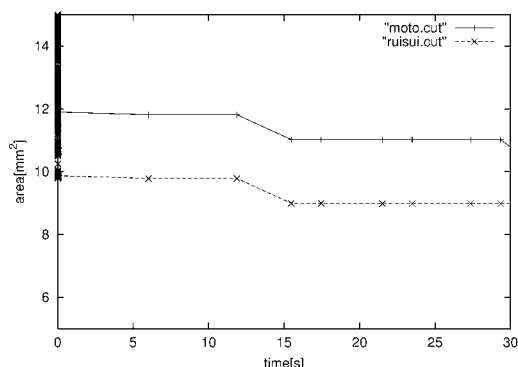


図 8: トレードオフ曲線

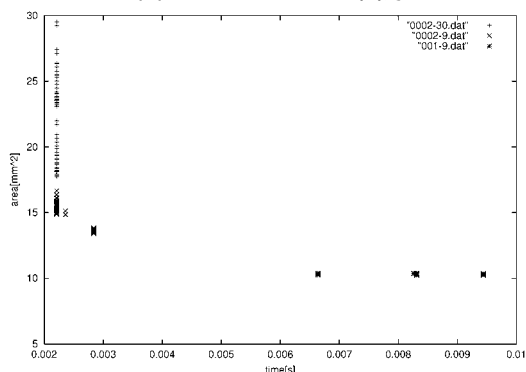


図 9: 改良候補解の分布

間制約 0.002[s], 「×」は面積制約 9[mm²], 時間制約 0.002[s], 「+」は面積制約 0.002[mm²], 時間制約 0.01[s] の場合である。これより、実現可能解がない場合には、制約条件に近い指標を持つ改良候補解が出力されることを確認した。

また、実現可能解数と定性指標評価後の詳細設計仕様数を表 7 に示す。ここで、表 7 において「A.B:C」は「アルゴリズム A における定性指標 B の制約が C」であることを表している。ただし「des」を「設計のしやすさ」、「test」を「テストのしやすさ」とする。表 7 より、定性指標制約を与えることで、面積/処理時間制約を満たす設計の中から、特に定性的に優れた設計を抜粋できたことを確認した。

7 あとがき

本稿では、「IP 再利用」「定性指標」「設計ナビゲーション」を導入した HW/SW 協調設計手法を提案し、その有効性を計算機実験によって示した。今後の課題としては、設計ナビゲーションにおけるパラ

メータの他項目化、ユーザーインターフェースの検討が挙げられる。

謝辞

本研究に関し、有用な議論および討論をいただきました。株式会社東芝 西尾誠一氏、相原雅己氏、影島淳氏、宮岡祐一郎氏および小田雄一氏に感謝致します。本研究の一部は、日本学術振興会科学研究費補助金 (若手研究 B, 課題番号 15700071), 電気通信普及財団研究調査助成金, 早稲田大学特定課題研究助成費 (2005A-872) の援助を受けた。

参考文献

- [1] Karam S. Chatha and Ranga Vemuri, "MAG-ELLAN: Multiway Hardware-Software Partitioning and Scheduling for Latency Minimization of Hierarchical Control-Dataflow Task Graphs," CODES'01, pp.42-47, 2001.
- [2] Bharat P. Dave, Ganesh Lakshminarayana and Niraj K. Jha, "COSYN: Hardware-software cosynthesis of embedded systems," in Proc. 34th DAC, pp.703-708, 1997.
- [3] Rajesh K. Gupta, Claudionor Nunes Coelho, Jr. and Giovanni De Micheli, "Synthesis and simulation of digital systems containing interacting hardware and software components," in Proc. 29th DAC, pp.225-230, 1992.
- [4] Joerg Henkel, Rolf Ernst, Ullrich Holtmann and Thomas Benner, "Hardware-software cosynthesis for microcontrollers," Proc. ICCAD-94, pp.96-100, 1994.
- [5] Joerg Henkel, "A Low Power Hardware/Software Partitioning Approach for Core-based Embedded Systems," Proc. 36th IEEE/ACM Design Automation Conference, pp.122-127, 1999.
- [6] Jinhwan Jeon and Kiyong Choi, "Loop Pipelining in Hardware-Software Partitioning," in Proc. ASPDAC, pp.361-366, 1998.
- [7] 東芝, ASIC データブック TC220 シリーズ プリミティブセル I/O セル (非線形遅延モデル), 1997.
- [8] 東芝, ASIC デザインマニュアル 32 ビット TX system RISC TX39 デザインガイド, 1998.
- [9] Steven Vercauteren, Bill Lin and Hugo De Man, "Construction application-specific heterogeneous embedded architectures from custom HW/SW applications," in Proc. 33rd DAC, pp.521-526, 1996.