2007－EMB－6 (8)
2007／12／4

SystemCを用いたハードウェア／ソフトウェア協調設計方式とその評価

遠藤 聡† 清尾 克彦‡ 三井 浩康† 小泉 寿男† 神戸 英利††
†東京電機大学
‡株式会社ゼネテック
††三菱電機株式会社

製品開発における理想モデルは短い開発期間と低い開発コストで高品質の製品を開発することである．競争が激化する組込みシステムの分野では，設計・開発を理想モデルに近づけ，設計生産性を向上させることが重要である．設計生産性を向上させる手法の一つとして，ハードウェアとソフトウェアを統一された環境で協調しながら設計をおこなうハードウェア／ソフトウェア協調設計方式が提唱された．

本稿では，ハードウェア／ソフトウェア協調設計の更なる設計生産性の向上を目的とした高抽象モデルの自動詳細化手法，および明確な指標によってシステムをハードウェアとソフトウェアに分割するトレードオフ解析を提案し，それらに対する評価について述べる．

Hardware/Software Co-design method of SystemC and estimation

Satoshi Endo† Katsuhiko Seo‡ Hiroyasu Mitsui† Hisato Koizumi† Hidetoshi Kanbe††
†Tokyo Denki University
‡Genetec corporation
††Mitsubishi Electric corporation

An ideal model in the development of products is to develop the product in a short period and a low cost and a high quality. In the field of embedded system which has intensified competition, it is important that designers develop the products close to the ideal model to improve productivity. Hardware/Software co-design method is gaining attention as one of the techniques for improving the design productivity in the environment in which hardware and software were united. In this paper, we propose an automatic refinement method of the high abstraction level model to improve further productivity of hardware/software co-design, and we also propose a trade-off method that divide the system into hardware and software according to the automatic refinement method. And we describe the evaluation of the proposed method.

1. はじめに

近年の組込みシステムの発展は目覚しく，組込みシステムは大規模化・複雑化している．携帯電話を例に挙げると，カメラやワンセグなど多彩な機能が続々と付加させているにも関わらず，製品のライフサイクルは約半年と短い．短い開発期間で製品を開発しなければならないにも関わらず，品質はある程度の水準を保たなければならない．このような状況下における設計者の負担は増大している．これらに対する解決策として新しい設計手法の確立やソフトウェア部品の再利用が挙げられる．本稿ではハードウェア(以下，H/W)／ソフトウェア(以下，S/W)協調設計方式に着眼する．H/W・S/W 協調設計方式は統一された環境下で各設計を協調させながら同時に進めていくので，H/W，S/W 間の検証やインタフェース設計を容易化でき，それに伴い，開発期間の短縮化を見込める．H/W・S/W 協調設計を実現するシステムレベル言語として，SpecC や SystemC 等がある．本稿では SystemC を用いる．SystemC を用いた H/W・S/W 協調設計は抽象度の異なる段階的なモデリングや機能・通信の独立設計をサポートする．

従来，提案されている H/W・S/W 協調設計方式に，本稿で提案する高抽象モデルの自動詳細化手法，および H/W・S/W トレードオフ解析を取り入れ，H/W・S/W 協調設計方式の設計生産性の向上を図る．

H/W を設計する際，設計初期段階で回路レベルの記述をおこなうことは難しい為，一般的には設計初期段階ではインタフェース(以下，I/F)や機能部の厳密な規定をおこなわない抽象度の高いモデルを記述し，徐々に抽象度を下げながら設計をおこなっていく．本稿ではこの抽象度を下げる作業を効率化する為に高抽象モデルの自動詳細化手法を提案する．

要求仕様を満たす適切なアーキテクチャの構築をお

こなう為に，もう一つの提案項目である H/W・S/W トレードオフ解析を提案する．

これら 2 つの提案項目について，実用性の評価をおこなう．

2. H/W・S/W 協調設計と SystemC

H/W・S/W 協調設計は統一された環境下で各設計を協調させながら進めていく手法であり，その実現にはシステムレベル言語の SpecC や SystemC が使用される．本稿では SystemC を用いて H/W・S/W 協調設計を実現する．

従来の H/W 設計，S/W 設計，I/F 設計は複数の異なったツールを用いて設計していた．その為，一人ですべての設計を取り扱うには多くのツールに精通していなければならなかった．また，一般的にデバイスが完成しないとデバイスドライバの設計に着手できず，各設計の連結は設計最終段階にならないと実行が困難であった．こののような問題点を解決する為に H/W・S/W 協調設計では統一されたツールで各設計を並行しておこなう．これにより，一人ですべての設計に従事することが可能になり，設計者間の意見の相違を抑制することができる．また，H/W と S/W の間の協調性検証を設計初期段階からおこなうことができ，開発期間の短縮化が図れる．

SystemC は C 言語をベースとしたシステムレベル言語(正確には C 言語のライブラリ)であり，H/W 記述の為の並列動作や同期，通信，タイミング等の構文が含まれる．図 1 のように SystemC は"モジュール"と"モジュール間の繋がり(チャネル)"で表現される．また，SystemC は異なる抽象レベルの記述をサポートしており，高い抽象レベルから設計を始めることができる．SystemC 記述の構造としてモジュールの機能部と通信部は独立されている為，機能部と通信部の設計の分離が可能である．機能部と通信部の独立は部分的な設計に役立ち，設計対象のシステムを部分的に詳細化していくことができる．抽象レベルを徐々に下げながらおこなう設計方式は，設計者の理解・把握を助ける．

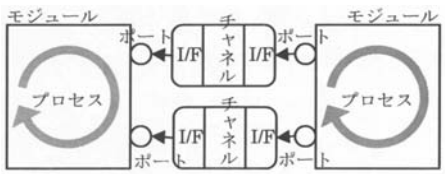

図 1 SystemC の構造

3. SystemC による H/W・S/W 協調設計方式

SystemC を H/W・S/W 協調設計に適用した場合の設計フローを図 2 に示す．

一般的な組込みシステムは CPU と複数の専用 H/W，リアルタイム OS(以下，RTOS と略す)上で動作する複数のタスクで構成される．H/W・S/W 協調設計ではこれらを総括して設計をおこなう．

(1) UTF モデル

モデリングの第一段階として，最も抽象レベルの高い UnTimed Functional(UTF)モデルを SystemC で記述する．UTF モデルは時間的要素を考慮せず，機能面からシステムを複数のモジュールに分割するモデルである．このモデルでは入出力，アルゴリズム処理がゼロ時間で処理される．モジュールの機能部は S/W モデル(UnTimed)でアルゴリズムが記述され，モジュール間は FIFO(First In First Out)で接続される．シミュレーションによってこのモデルが仕様を満たしていることを確認する．

(2) TF モデル

システムモデリングの第二段階として，UTF モデルに時間的要素を付加し，TF モデルに昇華させる．UTF モデルとは違い，ゼロ時間で処理はされず，時間経過に従って処理が進められる．この TF モデルによって性能シミュレーションをおこなう．

(3) H/W・S/W 分割

定量的評価に基づいてモジュールを H/W または S/W に割り振る．TF モデルの性能シミュレーションによって，モデルの改良を繰り返し，システムの最適

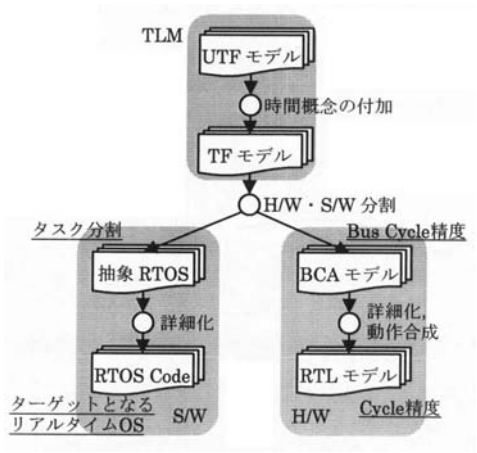

図 2 SystemC を用いた H/W・S/W 協調設計

な構成を模索する.

(4) H/W 設計／BCA モデル

H/W に割り振られたモジュールは BCA モデルに抽象レベルを下げる. SystemC によって H/W モジュールの通信部のみを RTL と同じサイクル精度で記述する. 次の RTL を生成する過程において, 動作合成をおこなう為には BCA モデルで動作合成可能な記述をおこなう必要がある.

(5) H/W 設計／RTL モデル

BCA モデルを動作合成, または設計者による RTL 設計により RTL モデルを生成する. RTL モデルは H/W モジュールの通信部に加えて機能部も RTL 化される. 更にこれを論理合成によってゲートレベル化し, ネットリストを生成する.

(6) S/W 設計

S/W は RTOS 上でタスクとして動作する. タスクは並列動作せる処理の単位である. S/W モジュールは融合・分解され, 複数のタスクになる.

4. 高抽象モデルの詳細化手法

4.1. 概要

SystemC による H/W・S/W 協調設計は異なる抽象度のモデルを使って, 段階的にシステムの詳細化をおこなっていく. その手順は, 機能部・通信部の抽象的なモデリング(UTF・TF モデル)→通信部の詳細化(BCA モデル)→機能部の詳細化(RTL モデル)となっている. SystemC での記述は機能部と通信部が独立している為, 片方のみに変更を加えることは容易である.

提案する高抽象モデルの詳細化手法は UTF・TF モデルを RTL モデルまで抽象度を下げる作業を自動化する. この高抽象モデルの詳細化手法を取り入れる為には, 高抽象モデルを記述する段階で制定された記述制約に則った記述をおこなわなければならない. 以下, 高抽象モデルの記述制約と詳細化手法について述べる.

4.2. 高抽象モデルの記述制約

高抽象モデルを詳細化する為には, 高抽象モデルを記述する段階で, 記述制約を与える必要がある. 高抽象モデルである UTF・TF モデルは, 機能部・機能部ともにタイミングや並列性等を考慮しない記述をおこなう. 通信部は FIFO(First In First Out)によって実現する.

高抽象モデルの詳細化手法は後述するシステムの H/W・S/W 分割を含み, 高抽象モデルの記述には H/W, S/W の区別はつけない.

基本的に詳細化は機能部と通信部を独立させて考える. その為, 機能部と通信部の明確な分割が必要になる. 機能と通信の分解の例として RGBtoYUV 色変換モジュールを挙げ, 図 3 に示す.

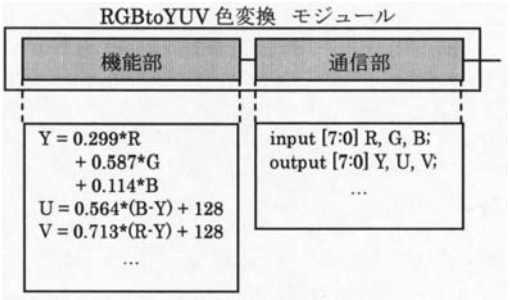

図 3 RGBtoYUV 色変換システムの機能部と通信部

抽象モジュールの詳細化にあたって, 抽象モジュールが満たすべき記述の制約は以下の項目が挙げられる. 以下に挙げる項目は全体の一部であり, 実際には更に多くの制約に縛られることになる.

- 制約内の演算子のみで構成する.
- ポインタを用いた動的な配列宣言は不可
- 決められた範囲内の C/C++構文サブセットを利用する.
- 入力・出力データを明確にする.

4.3. 詳細化システム[1]

高抽象レベルで記述されたソースファイル(*.c, *.cpp), ヘッダファイル(*.h)を読み込み, 低抽象レベルの記述に変換するシステムを構築する. 現在, これを実現するツール“ShaperSystem”を作成している.

(1) 変換法

抽象モジュールを H/W, S/W に自動変換する場合, 機能部の変換に加え I/F を生成しなくてはならない.

まず, 各モジュールの機能部と通信部を明確に分割する. 通信部は I/F の詳細化, 機能部は H/W(HDL), S/W(C/C++)への変換をおこなう.

S/W 化モジュールの変更項目を以下に示す.

- H/W へのアクセス機構 (Read/Write)
- H/W からのアクセス機構 (割込み)
- 機能部の S/W 化

H/W 化モジュールの変更項目を以下に示す.

- S/W へのアクセス機構 (割込みの要求)
- S/W からのアクセス機構 (Read/Write)
- 機能部の H/W 化

例として SystemC で記述された高抽象レベルの RGBtoYUV 色変換システムを論理合成可能な Verilog HDL の記述へ詳細化する過程を説明する. SystemC で記述された高抽象レベルの RGBtoYUV 色変換システムの一部をリスト 1 に示し, 変換後に想定される

Verilog HDL の記述の一部をリスト 2 に示す.

リスト 1 SystemC による高抽象レベルの記述

```
SC_MODULE(RGBtoYUV)
{
  sc_fifo_in<unsigned char> R, G, B;   //…①
  sc_fifo_out<unsigned char> Y,U.V;   //…②
・・・(略)・・・
};
・・・(略)・・・

void RGBtoYUV::Conv_RGBtoYUV(
  unsigned char r, unsigned char g, unsigned char b,
  unsigned char *Y, unsigned char *U, unsigned char *V)
//…③
{
  unsigned char y, u, v;
  y =0.299 * r + 0.587 * G + 0.114 * B;   //…④
        ・・・(略)・・・

  //結果
  *Y = y;
        ・・・(略)・・・
}
```

リスト 2　Verilog HDL による回路レベルの記述

```
module RGBtoYUV(CLK, RESET, RGB_VALID,
                YUV_EN, R, G, B, Y, U, V);
input CLK, RESET, RGB_VALID;
output YUV_EN;
input unsigned [7:0] R, G, B;   //…①
output unsigned [7:0] Y, U, V;   //…②
reg Y_EN, U_EN, V_EN;
reg unsigned [23:0] Y_, U_, V_;

parameter FIX_0_29900 = 19595;
   ・・・(略)・・・

always @(posedge CLK) begin
  if(RESET == 1'b1) begin
    Y_EN = 1'b1;
  end else if(RGB_VALID == 1'b0) begin
    Y_ = FIX_0_29900*R + FIX_0_58700*G
       + FIX_0_11400*B + ONE_HALF;
    Y_ = Y_ >> 16;   //…③
    Y_EN = 1'b0;
  end
end

   ・・・(略)・・・

assign YUV_EN = Y_EN | U_EN | V_EN;
assign Y = Y_;
assign U = U_;
assign V = V_;
endmodule
```

本稿の高抽象モデルの詳細化手法では，詳細化後の記述は Verilog HDL 記述に限定する．抽象レベルを下げても，基本的にはモジュールの粒度は変更しない．

通信部の詳細化では，リスト 1 の①②で FIFO として記述されている入力信号の R，G，B 信号，および出力信号の Y，U，V 信号を回路レベルのリスト 2 の①②の記述に変換する．入出力信号のビット幅は変更をしない．また変換後のモジュールはすべてクロック同期型とし，すべてのコンポーネントにクロック信号を入力として与える．また，内部の状態をリセットするリセット信号等やモジュールの有効／無効を切り替える信号などを付加する．

関数の引数と戻り値を，低抽象レベルのモジュールの入力信号と出力信号に対応させるのが理想だが，出力信号は複数の値を持つことが多く，戻り値に複数の値を持たせるには構造体を利用するなどの工夫が必要である．そこで，リスト 1 の③にように引数に出力信号を格納する変数へのポインタを入れる．この記述制約を設けることで，ポインタでない引数を入力信号，ポインタである引数を出力信号に対応させることができる．

機能部の詳細化では並列性およびアルゴリズムの変換について考慮する．RGBtoYUV 色変換システムでは，Y 値，U 値，V 値は並行して算出することが可能である為，リスト 2 ではそれぞれを always 文で分けて並列動作させている．一方，変換前のリスト 1 の記述では，Y 値，U 値，V 値は逐次的に求めている．並列動作可能か否かはプログラム内の変数の相互依存関係によって判断する．例えば，A=B という代入文があるとき，D=A という式は A=B と並列動作できないが，D=B ならば A=B と並列動作可能である．よってリスト 1 のような SystemC による高抽象モデルの記述の段階では並列性は考慮せず，自動変換に任せることができる．

RGBtoYUV 色変換では出力される YUV 値は整数であるが，計算には小数を含んでいる(リスト 1 の④参照)．このような場合，小数を左ビットシフトによって整数に変換し，計算後に右ビットシフトによって整数解を求める方法をとることで回路規模を小さくすることができる．リスト 2 の③では小数を 16 ビット左シフト(2 の 16 乗)し，演算後に 16 ビット右シフト(2 の-16 乗)している．現在，作成している高抽象モデルの詳細化システムではこのような場合に限って小数計算をビットシフトで簡易化する．H/W に合わせた計算式の変換は小数計算の簡易化の他にも様々なものが考えられる為，本研究において，今後，検討していくべき課題の一つとなっている．

5.　H/W・S/W トレードオフ解析[2][3]

5.1.　概要

各抽象モジュールを H/W または S/W に割り振ると

き，どのように割り振るかの指標を決定する．組込みシステムにおいて，H/W は専用回路，S/W はリアルタイム OS 上のタスクとして実装される為，一般的には H/W は S/W と比べ性能の面では勝るが，コストの面では劣る．性能向上とコスト削減のような二律背反した条件のもとで，要求された仕様を満たすアーキテクチャを構築する為にトレードオフ解析をおこなう．

5.2. 解析方法

まず，トレードオフ対象モジュールを抽出する．トレードオフ対象モジュールに対して性能解析と全体的なシミュレーションを繰り返し，H/W・S/W 割当てを決定する．もう一つの提案事項である高抽象モジュールの詳細化手法で開発している ShaperSystem の一部を利用して，各トレードオフ対象モジュールを H/W，S/W それぞれに割り当てたときの性能解析をおこなう．トレードオフ対象モジュールとは H/W に割り振るか S/W に割り振るかが決定していないモジュールのことで，トレードオフ解析によって H/W もしくは S/W に割り当てられる．トレードオフ対象モジュールは存在するモジュールの中から設計者が任意に決定するものとする．

高抽象モジュールの詳細化手法の項目で開発している ShaperSystem は SystemC を用いて高抽象レベルで記述されたモデルを Verilog HDL で記述された論理合成可能なモデルへ変換するプログラムであるが，提案する H/W・S/W トレードオフ解析手法はこの低抽象レベルのモデルの自動生成を利用して，各モジュールの性能解析をおこない，その結果をもとにトレードオフ解析をおこなう．

トレードオフ解析をおこなうにあたって，トレードオフ項目を設定をおこない，どのようなアーキテクチャを構築するのかを決定する．トレードオフ項目としては以下の項目が挙げられる．要求されるシステムに応じて優先項目を決定する．

- 性能
- コスト
- 消費電力
- 柔軟性

仕様を決定する段階で満たすべき性能を明確に決定していれば，その性能を最低限満たす H/W・S/W 割り当てをおこない，そのパターンの中でコストや消費電力が最小となるパターンを探索すれば良い．例として，一つの S/W モジュールと 2 つのトレードオフ対象モジュールからなるシステムの H/W・S/W 分割パターンを図 4 に示す．トレードオフ解析では図 4 に示すそれぞれのパターンにおいての性能を解析し，比較する．

コストと性能をトレードオフ項目の優先事項とした

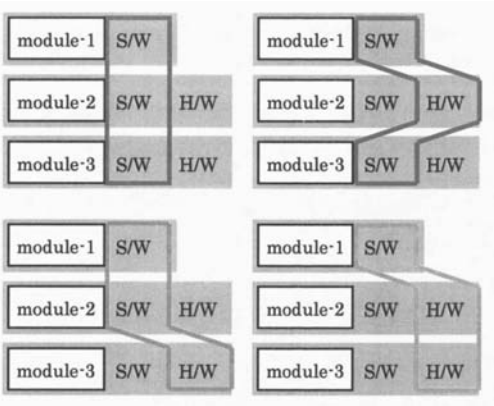

図 4 H/W·S/W 分割パターン

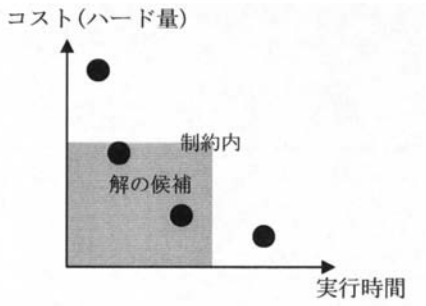

図 5 トレードオフ解析の例

場合の解析の例を図 5 に示す．制約としてコストと実行時間が決められているとき，解の候補はその両方の制約を満たす H/W・S/W 分割パターンとなる．解析の結果，制約を満たす H/W・S/W 分割パターンが複数ある場合は，設計者によって設定されたトレードオフ項目の優先順位に従って決定する．

6. 評価

6.1. 高抽象モデルの詳細化手法の評価[4]

実際にシステム開発を提案した手法に適用し，評価をおこなう．評価は，高抽象モデルの詳細化手法によって生成した回路レベルの記述が論理合成可能でかつ入力に対して正しい出力を出すかを調べる．

高抽象モデルの詳細化手法の評価をおこなう為の環境を表 1 に示す．

高抽象モデルの詳細化手法の評価をおこなう為の対象システムとして JPEG エンコーダ(基本方式)を採用する．JPEG エンコーダの概要を図 6 に示す．

JPEG エンコーダは RGBtoYUV 変換，DCT(離散コサイン変換)，量子化，エントロピー符号化の 4 つのモジュールに分割される．

表 1　開発環境

開発プラットフォーム	Windows XP
プログラミングツール	Microsoft Visual C++
システムレベル言語	SystemC 2.0
SystemC のシミュレーションツール	GTKWave
デバイスのコンフィグレーションツール	QuartusII Web Edition
デバイスの S/W 開発ツール	NiosII IDE
デバイス	NiosII CycloneII (EP2C35F672C6N)

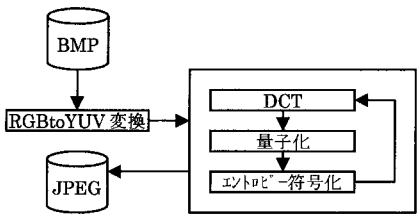

図 6　JPEG エンコーダの概要

手順としては，提案する詳細化手法が適用可能な高抽象モデル(UTF，TF モデル)の記述をおこない，詳細化手法を適用(同時に H/W・S/W トレードオフ解析をおこなう)し，実機上で動作検証をおこなう．UML 等を用いて仕様書からシステムのモデリングをおこなう作業は，本研究の主旨としていないので省略する．

DCT を H/W，その他のモジュールのが S/W の場合においての動作検証をおこない，高抽象モデルの詳細化手法を実現するプログラム ShaperSystem の動作を確認した．結果，S/W モジュール，H/W モジュールともに正しい出力を出し，H/W については動作合成可能であった．しかし，現段階では ShaperSystem の適用範囲は狭く，改善の余地がある．

6.2.　H/W・S/W トレードオフ解析の評価

H/W・S/W トレードオフ解析の評価は高抽象モデルの詳細化手法と同じ評価対象を用いる．

H/W・S/W トレードオフ解析の評価は JPEG エンコーダのシステムを H/W と S/W に分割する部分において評価をおこなう．

JPEG エンコーダの各モジュールは性能解析の結果，図 7 に示すように DCT が H/W モジュール，その他のモジュールが S/W として分割された．

設計したシステムの動作確認は前項の高抽象モデルの詳細化手法の評価でおこなったが，H/W・S/W トレードオフ解析の評価はトレードオフ解析をおこなった結果，それによって適切なシステムの分割がおこなわれたかを確認しなければならない．その評価方法として考えられるのは，シミュレーションと実機検証の比較である．複数の H/W・S/W 分割パターンに対してシミュレーションと実機検証を比較して，その差異がどれほどのものなのかを確認する．この評価については実施途中であるので，今後の課題とする．

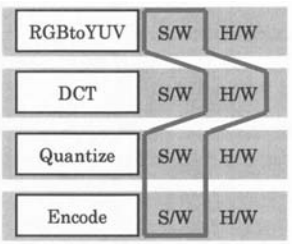

図 7　JPEG エンコーダの H/W-S/W 分割

7.　まとめと今後

高抽象モデルの自動詳細化手法および H/W・S/W トレードオフ解析を提案し，H/W・S/W 協調設計方式に取り入れた．現状では高抽象モデルの自動詳細化手法は限られた範囲の記述しかサポートしていない為，実用性を向上させる為にサポートの範囲を広げていく予定である．また，トレードオフ解析についても複数の適用対象に適用し，その有効性を示していく予定である．

参考文献

[1] 木村正裕，小林憲貴，山崎亮介，吉田紀彦，"リファクタリングに基づく段階的詳細化設計の GSM ボコーダへの適用"，ESS2006，pp.83-87 (2006)

[2]遠藤祐，小泉寿男，清尾克彦，"ハードウェア・ソフトウェア協調設計方式と ITS 画像処理開発への適用検証"，T.IEE Japan Vol.120-D No.10 (2000)

[3] 遠藤祐，吉田健，井上聡，飯田庸介，小泉寿男，"ITS 画像処理系・制御系開発におけるハードウェア・ソフトウェア協調設計方式"，情報処理学会論文誌 Vol.43 No.12 (2002)

[4] 大村正之，深山正幸，"C/C++による VLSI 設計-SystemC による JPEG コーデック設計-"，pp.107-138，2003 年発行