Vol.2010-EMB-19 No.1
2010/12/6

M 言語ベース設計によるフィルタ設計事例

松田昭信† 石原亨†

近年，組込みシステムのハードウェア設計のアルゴリズム開発及び検証において，M 言語を用いた設計手法が広まりつつある．M 言語とは，配列全体を処理する言語である．これは，規模の増大と複雑さが増しているアルゴリズム開発において，設計工数短縮と検証の容易さが求められているためである．よって，本稿ではハードウェアにおけるフィルタ設計において，信号処理アルゴリズムを M 言語によって設計した事例を示した．それにより，M 言語ベース設計フローの有効性が判明した．

A Case Study of Filter Design for Using a M-Based Language Design Flow

Akitoshi Matsuda† and Tohru Ishihara†

Recently, M language is becoming to be used for algorithm development and verification in embedded system design. M language is the language which processes the whole array simultaneously. It is expected that the M-based language design can reduce the design turn-around-time without losing the quality of design. This paper presents a case study of filter design for signal processing. In the filter design, M-based language design flow is used. The results of the case study demonstrate high efficiency of the M-based language design flow.

1. はじめに

現在，デジタル家電等の高機能化及び大規模化に従い，これらを構成する組込みシステムにおけるハードウェア部においては，開発期間・開発コストの増大や品質劣化の問題が多発している．これら複雑化するハードウェアの開発効率向上への取組みの中で，システムレベル設計及びアルゴリズム設計において，MATLAB/Simulink を用いた設計手法が注目を浴びている[1]．この MATLAB/Simulink を用いた設計手法には，

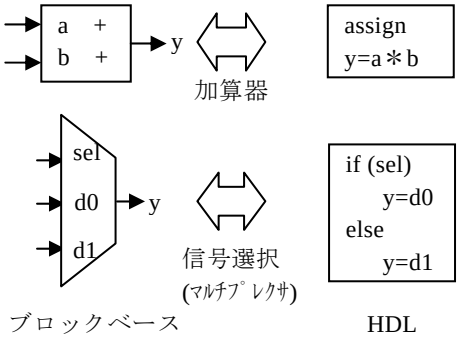


図 1 ハードウェブロック

```
% sobel coef
    sobel_x = [ -1   0   1 ;
                -2   0   2 ;
                -1   0   1 ];

    sobel_y = [ -1   -2   -1 ;
                 0   0   0 ;
                 1   2   1 ];
```

図 2 M 言語記述の一例

ブロックベース設計手法や M 言語ベース設計手法などがある．本稿でのブロックベース設計とは，図 1 のように，ハードウェアを構成するブロック（加算・乗算器，マルチプレクサからフィルタなどの大規模機能まで）を組み合わせてハードウェア回路を実現することである．この設計手法により，回路構成などが C 言語やハードウェア言語（HDL）よりも容易に理解しやすくなる．また，図 2 に画像処理アルゴリズム[2]の一つであるソーベルフィルタに用いられる，縦線検出と横線検出の 2 つのオペレータを M 言語にて記述した例を示す．一般的な特徴として，M 言語ベース設計とは，このような行列演算においては，数式表記に近い記述にて演算を可能とする．これにより，アルゴリズム開発における，演算処理過程などが容易に理解しやすくなる．これら 2 つの設計手法からは，回路構成の理解の容易さのみならず，そのままシミュレーション実行や HDL の自動生成なども可能となる．これにより，MATLAB/Simulink を用いたハードウェア設計は，従来の設計手法でのアルゴリズムレベル開発から，ハードウェア設計，実装および検証作業での，多くの人手によるコーディングを省くことができる．よって，この設計フローにおいて，アルゴリズムレベルの段階から処理過程を通して全体構造が可視化（見える化）できれば，設計 TAT の短縮および設計品質の向上に大きく貢献できる．また，組込みシステムのハードウェア及びソフトウェアがブロック間での接続構造や連携が明確になれば，これらの協調設計及び検証にも効率的である．そこで，今回はアルゴリズムレベル開発からハードウェア設計まで一貫して自動化になる，この設計手法による効率的なフィルタのハードウェア設計事例を示す．また，この新しい設計手法により，ハードウェア回路の設計工数や回路検証に一定の効果があったことを報告する．

本稿では，MATLAB/Simulink を用いた設計手法である，M 言語ベース設計手法を用いた．もう一つのブロックベース設計手法については，別の機会に紹介する．今回のアルゴリズム開発には，設計期間短縮と設計品質向上のために，ハードウェアのアルゴリズムレベルの構想及び開発段階からの設計生産性の向上が求められている．その対策の一つとして，図 3 に示す M 言語ベース設計手法を，ハードウェア設計への導入が検討されている．その取組みとして，ハードウェア設計の初期段階で検討するアルゴリズムである数学演算や信号処理（線形代数演算，複素演算など）[3]を M 言語で記述することが推進されている．それにより，その M 言語を入力とする高抽象度レベル合成技術を適用する設計手法がハードウェアを設計する上で非常に効率的である．

そこで，この M 言語ベース設計手法を効率的に適用するために，アルゴリズムレベル開発の段階でハードウェア化される信号処理へ M 言語ベース設計手法を適用して，アルゴリズムレベル開発段階の M 言語からハードウェア設計のための HDL までの設計フローの自動化を実現した．これにより，アルゴリズムの全体構造を可視化すると共に，不要変数の削除，ハードウェアのトレードオフ解析が容易となる効率的なハードウェア設計手法を採用した．本稿では，この設計フローをハードウェア設計に適用した事例を報告する．

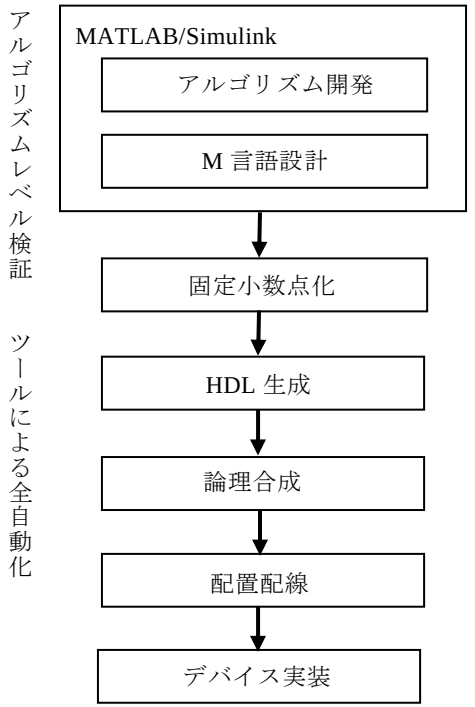


図 3　M 言語ベース設計フロー

Vol.2010-EMB-19 No.1
2010/12/6

2. M 言語ベース設計と C 言語ベース設計

組込みシステムにおけるハードウェア設計の設計工数短縮の解決手法として，種々の手法が提案されている[4]. 最も汎用的に提案されている手法として C 言語ベース設計手法がある[5][6][7][8][9]．これは，C ベース言語（C/C++，SystemC，SpecC など）から HDL へ動作合成ツールにより自動的に変換する手法である．しかし，C ベース言語は元来ソフトウェア向きであり，これらの動作合成ツールへの適合は，多くの労力を要する．さらに，システムレベル設計においては，C ベース言語は演算に関するデータ構造や演算子を実装してないので，種々の関数群を必要に応じて作成する必要があり，設計から実装までかなりの工数とコストを要する．例えば，画像処理の場合，各種フォーマットの画像ファイルに対して，それを設計対象に入力する場合は，画像データのストリームを作成するまでのプログラムを作成しなければならない. しかし，M 言語では，組込み関数として便利な I/O 関数があらかじめ用意されている．具体的には，画像やイメージファイルは，通常 BMP，JPEG，TIFF などの様々なファイル形式で存在するが[10]，MATLAB 上での組込み関数を使えば，簡単に画素データを配列にて読込むことができる．以下に，簡単に C 言語ベース設計に比べて，M 言語ベース設計の主な利点を挙げる．

- テストベンチの構築が容易
- 固定小数点化が容易
- デザイン制約への対応がスピーディ

つまり，M 言語ベース設計では，機能検証としてのシミュレーション環境が充実している．そこで，このような M 言語ベース設計を活用すれば，アルゴリズム開発からハードウェア設計までの一連の流れを短時間で精度よくシミュレーションできる．さらに，この M 言語から HDL へ確実かつ自動的に変換できれば，アルゴリズムレベルからハードウェアへの実装まで, 高抽象度レベルでのパフォーマンスを維持したまま, 高機能／高品質のハードウェアが設計可能となる．

3. M 言語での画像処理アルゴリズム開発

ここでは，M 言語ベース設計手法についての概略を説明する．具体的なモチーフとして，M 言語を用いて，画像処理の一つであるエッジ検出処理のアルゴリズムを開発した事例を示す．一般的に，画像のある特徴を知りたいときに，フィルタ処理を施して特徴を取り出すことが必要となる. ここでは, 特徴抽出の重要なものの一つである，エッジ検出について説明する．このアルゴリズム開発では，画像の縦線と横線のエッジを検出するソーベルフィルタを用いる[11]．通常，ソーベルフィルタのアルゴリズムには，表 1 のような縦線検出と横線検出の 2 つの係数行列が用いられる．その処理結果を図 4 に示す．この画像を見てわかるように，縦や横の線のみ，はっきりと検出

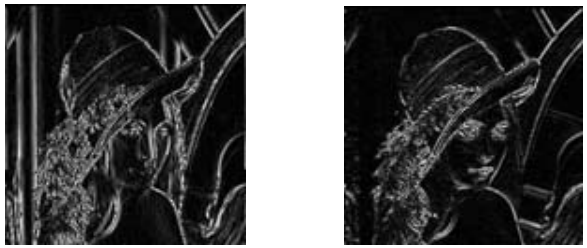
表 1　ソーベルフィルタの係数行列

(a) 縦方向係数

−1	0	1
−2	0	2
−1	0	1

(b) 横方向係数

−1	−2	−1
0	0	0
1	2	1



(a) 縦方向の抽出例　　(b) 横方向の抽出例

図 4　ソーベルフィルタの処理 1

図 5　ソーベルフィルタの処理 2

```
function [K] = edge_detector(I)

%% Parameter and coeff setting
% size
[nRows, nCols] = size(I);

% sobel coef
      sobel_x = [ -1    0    1 ;
                  -2    0    2 ;
                  -1    0    1 ];



      sobel_y = [ -1 -2    -1 ;
                   0    0    0 ;
                   1    2    1 ];



%% Edge Detection
% Initialize variables
K = zeros(nRows, nCols);



% Adding margin rows and colomns
Img_temp = [zeros(nRows,1) double(I) zeros(nRows,1)];
Img_marg = [zeros(1,nCols+2); Img_temp; zeros(1,nCols+2)];

% Main operation
for    iRow = 1:nRows
    for    iCol = 1:nCols

       ghs = 0;
       gvs = 0;
           └
```

```
    % Sobel filtering
       for    jRow = iRow:iRow+2
           for    jCol = iCol:iCol+2
              ghs = ghs + sobel_x(jRow-iRow+1,jCol-iCol+1)
                    * Img_marg(jRow,jCol);
              gvs = gvs + sobel_y(jRows-iRow+1,jCol-iCol+1)
                    * Img_marg(jRow,jCol);
           end
       end
       g = sqrt(ghs^2 + gvs^2);
    end

%Perform Thresholding
    if g >150
       K(iRow, iCol) = 255;
    else
       K(iRow, iCol) = 0;
    end
  end
end└
```

図 6　M 言語によるソーベルフィルタのアルゴリズム記述例

されている．それから，縦方向に検出されたエッジ画像の濃度値を$f_x(x, y)$，横方向に検出されたエッジ画像の濃度値を$f_y(x, y)$ とすると，この両方の検出をかけた画像の濃度値$f'(x, y)$ は，次式で計算できる．

$$f'(x, y) = (f_x(x, y)^2 + f_y(x, y)^2)^{1/2} \quad (1)$$

そして，この式（1）のアルゴリズムを計算して，その検出結果が図 5 となる．このように，ソーベルフィルタを用いることにより，エッジ検出ができるようになる．

この計算処理過程における，ソーベルフィルタのアルゴリズムについて M 言語を用いて開発した記述例を図 6 に示す．この M 言語のプログラムを見てわかるように，sobel 関数を行列式にてそのまま定義できる．また，画像データの配列を表す（I）などは，組込み関数で簡単に読込むことも可能である．

現在，ハードウェアの詳細設計には Verilog-HDL，VHDL などハードウェア記述言

語，ソフトウェア開発では C/C++などのプログラミング言語，これらを統合するシステム設計ではドキュメントによる仕様書，あるいはシステム記述言語によるハードウェブロックが用いられている．このように三者が相異なる言語を用いることは，システム設計の効率化を阻害する一つの要因となっている．

よって，M 言語ベース設計手法を用いることにより，抽象度の高いアルゴリズム開発から詳細仕様を決めてハードウェア実装モデルを作り上げていくまで，一貫して自動化することにより，設計生産性が向上することが容易に考えられる．また，M 言語により高抽象度のリファレンスモデルを作成して，シミュレーションをおこなうことにより，シミュレーション速度やモデリング効率が向上する．このように，M 言語から，HDL コードを自動生成することができれば，HDL を手作業にて記述せずに LSI 及び FPGA 設計などのハードウェア設計が単 TAT で可能となる．また，アルゴリズムや HDL のシミュレーションも容易にでき，機能検証の効率も向上する．よって，組込みシステムの効率的な検証，協調シミュレーション環境が実現可能となる．

4. 画像フィルタ設計事例

ここでは，前節で述べた画像処理アルゴリズムの 1 つであるソーベルフィルタの設計おいて，M 言語ベース設計を適用した事例を報告する．今回は，このアルゴリズムのハードウェア設計のターゲットデバイスとして FPGA を使用した．まずは，図 6 で示した M 言語記述を MATLAB 上でシミュレーションを実行する．その過程で，アルゴリズムの妥当性を検証して，仕様書通りの結果が導き出されれば，次は，ハードウェア化へのフェーズへ入っていく．そこで，完成された M 言語をハードウェア化への制約，例えば，固定小数点化，入出力ビット数，動作周波数などを与えて，動作合成処理を実行する．これにより，アルゴリズムレベルの M 言語から HDL が自動生成される．その回路性能を表 2 に参考として示す．ここでは，最重要な制約は動作周波数であり，これを 100MHz としている．これらを満たしており，希望通りの回路が生成されたことが理解できる．

次に，これらの設計工数を評価する．図 7 では，ソーベルフィルタ設計の仕様検討（アルゴリズム開発を含む）は共に M 言語を用いたが，その後，M 言語ベース設計環境で設計自動化した M 言語ベース設計工数と，ハンドコーディング（手設計）によって HDL を設計した手設計工数との比較をおこなった．M 言語ベース設計手法では，アルゴリズムレベルの M 言語をから, ハードウェア設計を考慮したシミュレーションを実行するため，浮動小数点処理から固定小数点処理への変換作業や，M 言語をから HDL のコーディング作業が自動化となる．また，HDL 生成と共に HDL 記述のテストベンチも自動的に生成されるので，シミュレーション（Simulation）検証時間も短縮された．さらに，仕様検討が終了すると同時に，ハードウェア／ソフトウェア設計が並

表 2　回路性能結果

レイテンシ	加算器	ロジック段数	動作周波数	面積
8	15	9	101MHz	4455Gate

図 7　設計工数の比較

列に検証できる．これにより，設計のイタレーション及びリスピンが減り，全体的な設計工数も短縮される．今回の設計事例では，設計工数が 84(h)時間から 54(h)時間に短縮しており，全体の約 36%工数削減が実現できた．これには，動作合成技術の取り扱いや操作習得工数や HDL シミュレータツールとのインターフェースの設定工数などを考慮していないため, 初回のトライでは 5～10 時間の余分に要する可能性はあり, 若干誤差が出ることも想定される．しかし，この M 言語ベース設計を継続することにより, 操作環境の再利用性も高まり, さらに工数削減効果を生み出せる可能性もある．

5. 通信フィルタ設計事例

次に，別の具体的なモチーフとして，多くの組込みシステムに適用されているデジタルフィルタを選択した．これも，前節と同様にハードウェア設計のターゲットデバイスとして FPGA を使用し，M 言語ベース設計手法を評価した．これらフィルタのアルゴリズム開発を M 言語用いて設計し，その後，このアルゴリズムを浮動小数点レベルにて検証した．アルゴリズムの演算及び動作に問題がなければ，ハードウェア化を目的とするため，固定小数点化への自動変換を実行する．固定小数点処理では整数部分に用いるビット数と小数部分に用いるビット数をあらかじめ固定して表現するため，

Vol.2010-EMB-19 No.1
2010/12/6

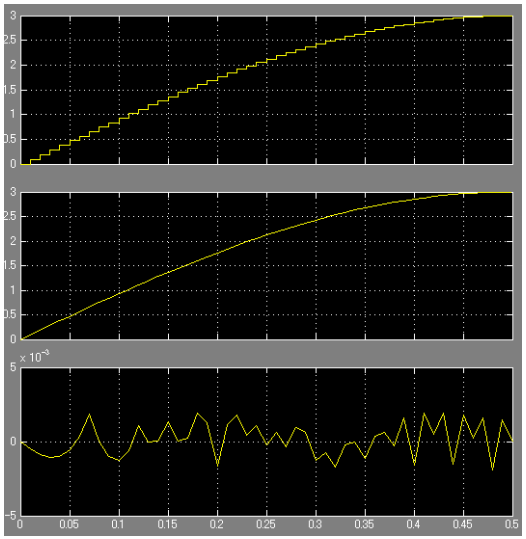


図 8　浮動小数点と固定小数点の比較

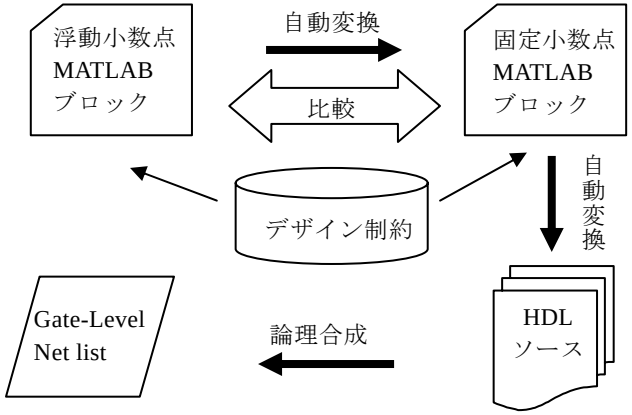


図 9　デジタルフィルタ設計フロー

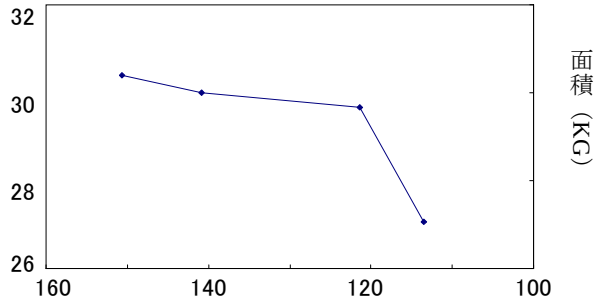


図 10　トレードオフ解析結果

同じく実数を近似表現する方式で小数点の位置が可変である浮動小数点処理に対しては，どうしても量子化による誤差が発生する．この誤差の程度を検証して，ハードウェア設計をする必要がある．これらの確認も MATLAB 上では簡単かつスピーディに確認できる．ここでは，具体的に，固定小数点処理と浮動小数点処理の結果を表示して，量子化誤差を表した結果を図 8 に示す．図 8 では，上段に固定小数点処理のシミュレーション結果を示し，中段に浮動小数点処理のシミュレーション結果を示した．そして，下段には，これら 2 つの結果の誤差を表示するようにした．この図 8 から，2 つの量子化誤差は, 1000 分の 3 以内の範囲に収まっていることが理解できる. さらに, MATLAB 上ではこれらの 3 つのシミュレーション結果を同一画面に同時に表示できるため，簡単に比較できることも検証が容易な点で大きな特徴である．

そこで，動画像処理やカーナビゲーションなどによく用いられるカルマンフィルタを設計した事例を説明する．まず，カルマンフィルタのアルゴリズム開発に M 言語を用いて，ハードウェア設計までのフローを図 9 に示す．この図を見てわかるよう，アルゴリズム開発段階，浮動小数点 M 言語モデルを作成してしまえば，ハードウェア実装のゲートレベルまで自動的に設計フローが流れることがわかる．

一般的に，デジタルフィルタの設計において，2 つの主要な制約は，これらの速度（動作周波数）とコスト（デバイス面積）である．つまり，高速でかつ低コストの成果物が要求されている．しかし，この 2 つの要素はトレードオフの関係にあるため，多くのパラメータを振って検証する必要があり，設計のイタレーションが多数回発生する．しかし，これらの一連の設計フローに対して，設計自動化が可能であれば，デ

ザイン制約を変更するだけで，簡単かつ短時間に多くの組合せのトレードオフ解析ができてしまう．この設計事例の解析結果を図 10 に示す．この図より，設定制約が動作周波数 100MHz 以上で面積が 30KG 以下であるので，120MHz 付近が最適解であることが理解できる．このように 120MHz 付近が最適解であることが判明した．このように，M 言語ベース設計手法を用いることにより，トレードオフ解析が，アルゴリズムレベルの M 言語を変更することなく，スピーディかつ正確に実行可能となる．

6. FIR フィルタ設計及び検証事例

次に，デジタルフィルタとして最も一般的な FIR フィルタについて，M 言語ベース設計を用いて設計及び検証をおこなった事例を示す．このフィルタは，携帯電話や受信機などに広く活用されている．今回は，16 次 FIR 型ローパスフィルタの設計をターゲットとして設計仕様書を作成した．主なフィルタの仕様である，サンプリング周波数は 44kHz とし，カットオフ周波数は 2kHz とする．その他，フィルタの係数である，応答タイプ，周波数仕様，振幅仕様などは MATLAB 上で簡単に設定できる．今回の M 言語ベース設計手法の環境で設計すると，色々なパラメータをユーザ変数とすれば，変数を自由に変更することができ，フィルタ全体の設計仕様を変更することが可能となる．ここでは，フィルタのハードウェア化を実行するため，その変数の中で重要な変数の 1 つである入力ビット数に着目した．先ほど述べたように，入力ビット数を構成する，Number of bits，Binary point を変数化しておくことによって，データ長や小数点の位置を簡単に変更することが可能となる．これらの設定形式を簡易的に図 11 に示した．フィルタのアルゴリズム開発レベルでは，入力信号は浮動小数点で処理されているが，ハードウェア化を最終目的とするには，固定小数点化処理は必須である．このため，入力信号を何ビットで表現するか，ビット数の設定が必要である．ここで，入力ビット数を増やせば，信号の精度は高まるが，回路規模が増大（コスト増大）する．逆に，入力ビット数を少なくすると，信号の精度は劣るが，回路規模は小さくなる．前節でも述べたトレードオフ解析については，ここでは入力ビット数を要素として選択した．ここで，入力ビット数を変数化させて，回路性能が変化する結果を表 3 に示す．設計仕様段階では，入力ビット数を 12 ビットか 16 ビットで検討しており，それらの判断を表 3 によって結論付けることにする．表 3 に示すように，これら 2 つのファクターで動作周波数も面積も大きな変化がなかったので，精度のなるべくよい 16 ビットを選択した．ここでの入力信号ビット数 16 ビットとは，データ長（Number of bits）が 16 ビットで小数点の位置（Binary Point）を 10 ビットとしたことを示す．これらトレードオフ解析から，今回の M 言語ベース設計手法の適用により，解析結果の早期入手と設計仕様の高精度検証が実現でき，設計工数の改善とデザイン検証の精度向上が得られたことがわかった．

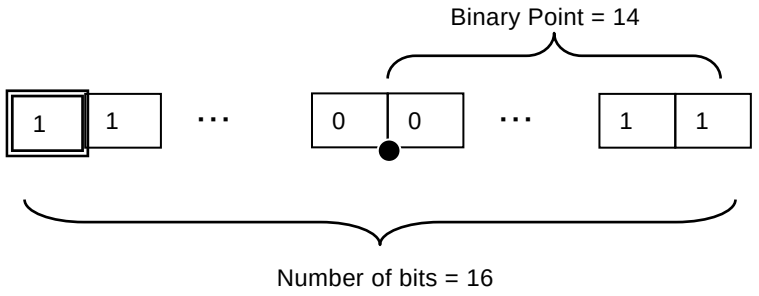


図 11　入力ビット数設定

表 3　入力ビット数によるトレードオフ解析

ビット数	8	12	16	20
周波数(MHz)	142	136	133	110
面積(Gate)	923	1170	1215	1890
Slice使用数(個)	67	83	103	144

7. まとめ

今回の M 言語ベース設計手法を用いることにより，アルゴリズム検証からハードウェアのシステム記述，詳細設計まで一貫してほぼ自動化することによって，システムレベル設計が高品質かつスピーディに実行できることがわかった．今まで，このような取組みは様々な方向からアプローチされてきた．現在，その代表的な手法が，C 言語ベース設計手法（動作合成手法）であり，この設計手法を用いることにより，C ソースコードから HDL への書き換え作業が不要となり，また，シミュレーションに要する時間も短縮され設計工数が短縮される．しかし，C 言語ベース設計環境を構築することは，動作合成ツールに過度に依存した環境構築や設計資産の再利用性が課題となっている．また，ベースとなる C 言語のコーディングが複雑になり，逆に工数が増大するケースもある．

これらの課題を解決するため，シミュレーション環境の構築が容易な M 言語ベース

設計手法を試みた．この手法が普及することにより，これまで熟練したソフトウェア技術者及びハードウェア技術者にしかできなかった C 言語設計及び HDL 設計という足かせを組込みシステム設計から排除し，アルゴリズムレベルで両技術者が同じシミュレーション上で検証及びデバッグをおこなうことが可能になる．これにより，システム設計及びアルゴリズム開発者の構想が正確かつスピーディにハードウェア実装ブロック作成に反映できるようになる．

また，アルゴリズム開発からハードウェア開発をシームレスに連携するための重要な要素の 1 つである，浮動小数点表現から固定小数点表現に変更するフェーズは，アルゴリズム設計からハードウェア化するにあたり重要な作業である．この作業に対して，今回は重点的に M 言語ベース設計手法の適用を試みた．この作業は，C 言語ベース設計においても重要な要素であるが，今のところ，M 言語ベース設計のように完全自動化には至っていない．

今回の結果は，FPGA デバイスへ特化されて最適化されている面もあるので，あらゆるハードウェア設計において，同じ結果は出ないのかもしれないが，着実に設計期間が短縮されることは理解できた．また，動作合成などのツールに過度に依存することなく，M 言語をブロック化してデータベースに蓄積することも可能となった．また，これらのアルゴリズムが可視化（見える化）されることにより，設計資産の再利用にも貢献できると考える．

8. 今後の課題

現状では，まだ全てハードウェア設計へ M 言語が適用できるわけではなく，例えば，高度な関数や，複雑な制御回路などの対応ができていない．また，生成された HDL コードの等価性検証及びフォーマル検証（プロパティ検証）などが十分に実施できる環境が整っていない．今後は，M 言語モジュールと HDL モジュール間において接続関係を設定するだけで，協調シミュレーション及び協調検証が実現できれば，検証時間短縮にもさらに貢献できると考える．これらの課題についても，今後さらなる研究を進めていく予定である．

参考文献

1) Gerard De Hann, and Erwin B. Bellers: Deinterlacing –An Overview, Proc. of IEEE, Vol.86, No.9, pp.1839-1857 (1998).
2) Rafael C. Gonzalez and Richard E. Woods: Digital Imaging Processing, 2nd edition, Prentice Hall (2001).
3) Jhon, Woods, W.: Multidimensional Signal, Image and Video Processing and Cording, Academic Press (2006).
4) 松田昭信: システム LSI 設計における CAD シミュレーション技術，信学技報, Vol.101, No.473, pp.37-42 (2001).
5) Ghosh, A., Kunkel, J. and Liao, S.: Hardware Synthesis from C/C++, Din Proc. of DATE’99, pp.387-389 (1999).
6) Arnout, G.: C for System Level Design, in Proc. of DATE’99, pp.384-386 (1999).
7) Gajski, D., Zhu, J., Domer, R., Gerstlauer, A. and Zhao, S.: Spec C, Specification Language and Design Methodology, Kluwer Academic Publishers.
8) Yamada, A., Nishida, K., Sakurai, R., Kay, A., Nomura, T. and Kambe, T.: Hardware synthesis with the Bach system, in Proc. of IEEE ISCAS’99, Vol.VI, pp.366-369 (1999).
9) Wakabayashi, K.: C-based Synthesis Experience with a Behavior Synthesize, “Cyber”, in Proc. of DATE’99, pp.390-393 (1999).
10) Bahadir K Gunturk et al.: Demosaicking, Color Filter Array Interpolation, IEEE Signal Proc. Magazine, Vol.44 (2005).
11) Vaidyanathan, P.P.: Multirate Systems and Filter Banks, Prentice Hall (1993).