International Journal of Computing and Digital Systems
-- An International Journal

M-based Filter Design for Communication and Imaging Systems

Akitoshi Matsuda[1] and Shinichi Baba[2]

[1]Dept. of Automotive Science, Kyushu University, Fukuoka 814-0001, Japan
matsuda_aki@slrc.kyushu-u.ac.jp
[2]Kyushu Embedded Forum, Fukuoka 814-0001, Japan
shinichi.baba@nifty.com

Abstract: The MATLAB M language provides constructs for the simple representation of functions, and has been used extensively for algorithm development and verification in communication and imaging systems design. The M language processes entire arrays simultaneously, and it is expected that design based on M can be implemented faster and with the same or higher quality as compared with design based on other languages. This paper presents a design methodology using M-based design flow for signal processing filters. The results of this design methodology demonstrate the high efficiency of the M-based design flow as compared with conventional functional modeling and manual coding using a hardware description language.

Keywords: M language; algorithm development; verification; filter design; signal processing.

I. INTRODUCTION

In the design of current large-scale and highly functional embedded systems, enhancement of design quality often entails an increase in the time and cost of developing the hardware of embedded systems. Design methods based on MATLAB and Simulink focus on improving the efficiency of developing increasingly complex hardware as well as algorithm design in specification-level design [1]. Here, a design method is proposed based on the MATLAB M language. In this paper, model-based design (Fig. 1) represents hardware models (multipliers, adders, multiplexer, etc.) as combinations of hardware circuitry. The adoption of this representation facilitates the understanding of hardware configurations and architectures in comparison with representation based on the C language or a hardware description language (HDL). Fig. 2 shows coefficient matrices used in a Sobel filter, which is commonly used for edge detection in image processing. In the example, two operators (one for horizontal and one for vertical line detection) are represented in the M language [2].

In general, these matrix operations provide a description similarly to mathematical formulas in M-based design, which also facilitates the understanding of the processes in algorithm development. In addition, these two design methods support automatic simulation and generation of HDL code. Thus, their adoption in hardware development is expected to reduce the need for manual coding as compared with traditional methods in design, implementation, and verification at the algorithmic level. A "visualization" feature can also be added to the entire development process, which is also expected to reduce turn

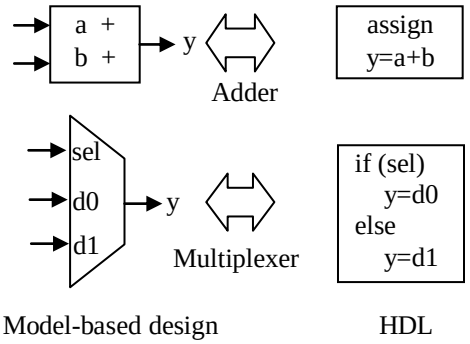


Figure 1. Hardware model.

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

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

Figure 2. Example of matrix representation using M.

around time (TAT) and to improve the overall design quality.

Our paper is structured as follows: Section 2 describes the background to M-based design. Section 3 gives an overview of design methodology in both M-based design and C-based design. Section 4 presents an outline of the image processing algorithm used in this study. Section 5 describes a set of experiments that we conducted on diverse real-world applications on the filter, and furthermore presents the design methodology for the implementation and simulation of M-based design. After that, we discuss the results in Section 6 and state our conclusions in Section 7. Lastly, Section 8 discusses future work.

II. BACKGROUND

In this paper, we introduce M-based design methods implemented in MATLAB and Simulink. Model-based design methods implemented in MATLAB and Simulink will be described elsewhere. Increasing the productivity of hardware implementation from the stage of algorithm design and exploration should help to improve TAT, as well as the overall quality of the entire embedded system. From the results of several trials, the M language has been recommended for algorithm design in earlier stages of hardware development, which includes mathematical operations for signal processing (linear operations, complex arithmetic operations, etc.) [3]. Behavioral synthesis with the M language is expected to be advantageous for hardware design since M is highly efficient for developing hardware design techniques that apply synthesis at a high level of abstraction and language input. This efficient M-based design method was applied in this work with the aim to implement signal processing algorithms in hardware. While an M-based design method was utilized in design development, the HDL code for hardware implementation was generated with a language conversion tool in order to automate the design flow. An efficient hardware design method was used that facilitates the trade-off analysis of hardware design in order to visualize the overall structure and to remove unnecessary variables. In this paper, we report a case study on the application this design method to hardware design flow.

III. DESIGN BASED ON M AND C BASED LANGUAGES

Various methods have been proposed for reducing the time necessary for designing embedded system hardware. Schliebusch et al.'s technique [4] has been proposed as the most versatile in the case of design methods based on C-based languages (such as C/C++, SystemC, and SpecC) [5,6,7]. Techniques for automatic conversion from C-based languages into HDL by means of behavioral synthesis tools have been developed [8]. However, since C-based languages are originally software-oriented, the utilization of these behavioral synthesis tools for this task poses considerable difficulties. Furthermore, in system-level design, C-based languages do not implement the data structure and operators necessary for the target operation, which further increases the cost and effort associated with design implementation since these operations must be created separately in accordance with the required functionality. For example, in image processing of files in various formats, a program must be written to create a stream of image data. For this purpose, input/output functions are embedded into the M language. Specifically, image files are typically stored in various formats such as BMP, JPEG, and TIFF, and pixel data can be read as an array by using the built-in functions in MATLAB [9]. The following items represent several major advantages of the M language over C-based languages in regard to hardware design:

- Simplified test bench generation.
- Straightforward creation of fixed-point models.
- Quick response to design constraints.

In other words, the M language provides a rich simulation environment for functional verification. Therefore, if the advantages of the M language are utilized in the design process, the flow from algorithm development to hardware design can be simulated rapidly and accurately. Moreover, the automatic conversion of M source code into valid HDL code is expected to enable the design of high-performance and high-quality hardware in a seamless process encompassing all steps from algorithm development to hardware implementation, while maintaining a high level of abstraction.

IV. ALGORITHM DEVELOPMENT

In this section, the image processing algorithm used in this study is described. In image processing, there are a variety of purposes that require detecting a boundary (edge) of an area in the image. An edge can be detected by using the differential operator for the change in pixel brightness, because this change is large at the region boundary. Another possibility is using a difference operation instead of a differential operation in processing the digitized image. Here, a coefficient combination for performing the differential operation is referred to as a differential filter. Differential filters can be classified as first-derivative filters (Sobel filter, Prewitt filter, etc.) and second-derivative filters (Laplacian filter, etc.).

A linear filter is generally expressed in an input image $f(x,y)$ and filter function $h(x,y)$ with a convolution integral as follow:

$$g(x,y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x-x,y-y)\,f(x,y)\,dxdy \tag{1}$$

In an actual computer, a digital calculation is performed using a digital image. Therefore, equation (1) is expressed in discrete form as follows:

$$g(i,j)=\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} h(i-k,j-l)\,f(k,l)\,. \tag{2}$$

Here, M and N are the number of pixels along the x- and y-directions of each image. Moreover, a neighboring region in the filter is a rectangular region around the target image element.

$$g(i,j)=\sum_{k=i-m_1}^{i+m_2}\sum_{l=j-n_1}^{j+n_2} h(i-k,j-l)\,f(k,l)\,. \tag{3}$$

Furthermore, we define *m*, *n*, and *h*' as follows:

$$m = k - i;\ n = l - j;\ h'(i, j) = h(-i, -j), \tag{4}$$

and

$$g(i, j) = \sum_{m=-m_1}^{m_2} \sum_{n=-n_1}^{n_2} h'(m, n) f(i + m, j + n). \tag{5}$$

This section describes a differential filter. We calculate the differential in the image with respect to *x* and *y*:

$$\frac{\partial f}{\partial x} = f(i+1, j) - f(i, j) \tag{6}$$

and

$$\frac{\partial f}{\partial y} = f(i, j+1) - f(i, j). \tag{7}$$

In the first-derivative filter, we then set up a differential whose magnitude is independent of direction:

$$\nabla f = \left\{ \left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right\}^{1/2}. \tag{8}$$

However, this calculation is nonlinear. The output filter is therefore a nonlinear filter [10]. In the second-derivative filter, the Laplacian of the second-order differential equation is shown below:

$$\nabla^2 f = \frac{\partial^2}{\partial x^2} f + \frac{\partial^2}{\partial y^2} f. \tag{9}$$

We present an overview of a design method based on the MATLAB M language. This particular section presents the development of an image processing algorithm for edge detection with the aid of the M language. In general, when investigating image features, it is necessary to perform feature extraction and filtering. Accordingly, edge detection, an important feature extraction process, is described here. This algorithm uses Sobel edge detection of horizontal and vertical lines in images [11]. Typically, the Sobel filter uses coefficient matrices (shown in Table 1) to detect horizontal and vertical lines Fig. 3 shows the original image, while Fig. 4 shows the results of the edge detection process [12]. It is apparent that only vertical and horizontal lines are clearly detected in the respective steps of this algorithm. Combining the horizontal and vertical gradients (f_x(x,y) and f_y(x,y) respectively), the edge detector *f* '(x, y) can be calculated as follows:

$$f'(x, y) = (f_x(x, y)^2 + f_y(x, y)^2)^{1/2}. \tag{10}$$

Equation (10) is calculated for each pixel, the resulting image shows sharp contours representing edges in the image. Thus, edge detection is possible by using a Sobel filter. In this calculation process, an example of an implementation of the Sobel filter algorithm in M is shown in Fig. 5, where the coefficient matrices used in the filter can be defined directly inside the function. The image data can be represented in the form of an array, such as (I) in the example, by using built-in file input functions.

Currently, the Verilog HDL and VHDL languages are used for detailed design of hardware, the C and C++ programming languages are used for software development, and system description languages or blocks are used for system design in order to integrate the specifications in a manner that satisfies the specifications. Thus, one factor that obstructs the efficient design of embedded systems is the use of different languages by the parties involved, namely, the hardware designer, the software designer, and the system designer.

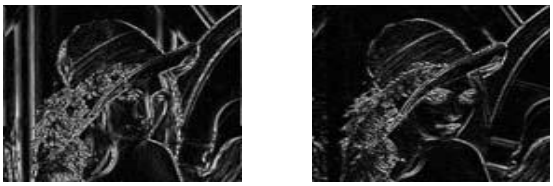
In this regard, a model can be created automatically in order to determine the detailed specifications for hardware implementation from a high-abstraction algorithm developed by using the M-based design method, which can improve design productivity. In addition, simulation performance and modeling efficiency can also be improved when a reference model is created at a high abstraction level with the M language and a simulation is conducted with that model.

TABLE I. COEFFICIENT MATRICES USED IN SOBEL FILTERING

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

Figure 3. Original image.



Vertical feature extraction Horizontal feature extraction

Figure 4. Vertical and horizontal feature extraction with Sobel filtering.

Figure 5. Combined results of vertical and horizontal feature extraction.

Therefore, if HDL code can be generated automatically from the M language, it would be possible to design a large-scale integration (LSI) and field-programmable gate array (FPGA) in short TAT and without the need for manual HDL coding. In addition, the developed algorithms and the HDL code can be easily simulated, which improves the efficiency of functional verification. As a result, these measures would allow for the implementation of efficient verification and co-simulation environments for embedded systems.

V. EXPERIMENTAL RESULTS

A. Case study of image processing filter design

In this section, we report a case study where the M language was used in the design of the Sobel image processing algorithm described in the previous section. In this case study, an FPGA was used as the target device for designing a hard-ware implementation of this algorithm. First, the description in M shown in Fig. 6 and 7, was simulated in MATLAB in order to validate the algorithm, where the verification was followed by the hardware development phase if the results agreed with the specifications. The constraints of the hardware implementation, such as fixed-point operations, the number of input and output bits and the performance frequency, were given, and behavioral synthesis was conducted. HDL was automatically generated from the M source code at the algorithmic level. Table 2 shows the circuit performance for reference. The most important constraint in this study was the operating frequency of the FPGA, which was set to 100 MHz. The circuit was considered to be generated as desired when this constraint was met.

```
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 columns
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
```

Figure 6. Example of algorithm development in M.

```
clear all;
clc;
close all;

% Stimulus
          I = imread('original.jpg');

% Edge Detector (HW part)
          K = edge_detector ( ...
            I(:,:,3) ...
             ) ;
% Display result
          O(:,:,1) = uint8(K);
          O(:,:,2) = uint8(K);
          O(:,:,3) = uint8(K);

figure('Position',[100, 100, 1200, 500]);

          subplot(1,2,1)
          image(I)
          subplot(1,2,2)
          image(O)
```

Figure 7. Example of script in M.

TABLE II. PERFORMANCE RESULTS

Latency	Adder	Logic level	Frequency	Area
8	15	9	101MHz	4455 Gate

Next, the design time was evaluated. The M language was used to examine the Sobel filter design specifications (including algorithm development). Thereafter, the time required for automatic generation of HDL code in M-based design was compared with the time necessary for manual HDL coding. In the M-based design method, which was based on the algorithmic level of abstraction, a simulation considering the hardware design was conducted. Thus, the process of converting floating-point operations into fixed-point operations

TABLE III. COEFFICIENT MATRICES USED IN PREWITT FILTER

-1	0	1
-1	0	1
-1	0	1

1	1	1
0	0	0
-1	-1	-1

TABLE IV. COMPARISON BETWEEN SOBEL AND PREWITT FILTERS

Item	Sobel	Prewitt
Max freq. (MHz)	100.9	102.9
Area (KG)	7.74	7.49
LUT (pieces)	393	374
Levels of logic	9	7
Run time (s)	438	441

(i.e., from M to HDL code) was automatic. In addition, the test benches of the HDL description were also generated automatically and simultaneously with the generation of HDL code, and the time required for simulation verification was reduced. Note that hardware and software design can be validated in parallel with the design requirements after the specifications are in place. Doing so can reduce the number of design iterations and re-spins and thus reduce the overall development time. Moreover, the reusability of the operating environment increased by continuing the M-based design process, which presents further possibilities for reducing the development time [13].

Next, a Prewitt filter was designed by using M-based design. This design was easily accomplished by simply changing the values of the matrix elements of a Sobel filter. Specifically, some values of the matrix elements in Table 1 are changed to those in Table 3. Table 4 compares these two filters. The performance (maximum operating frequency) and area (K-gate number) of these two filters were nearly the same. The run time in the table indicates the implementation time from register-transfer level generation using the M language. In short, once a Sobel filter is designed using an M-based language, a Prewitt filter can be designed within approximately 500 s. This remarkable feature is also effective for design reuse in M-based design flow.

B. Case study of comunication filter design

We also conducted a case study involving the design of a digital filter used extensively in various embedded systems. As in the previous section, the target device of the design case study was an FPGA. In this case, the filter algorithm was developed in M, after which it was verified at the level of floating-point operations. When the behavior of the algorithm

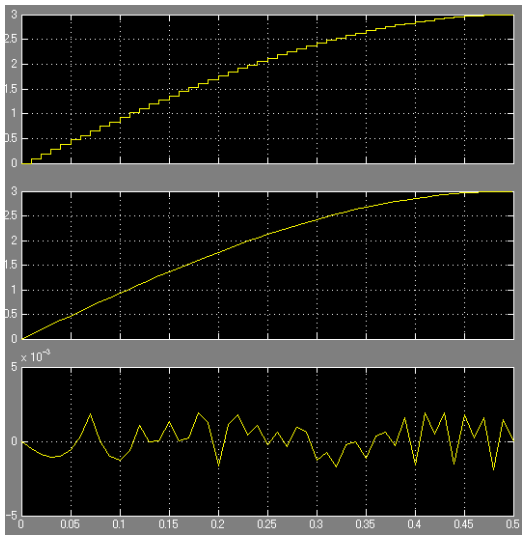


Figure 8. Comparison of fixed-point results (top) and floating-point results (middle), and the resulting quantization error (bottom).

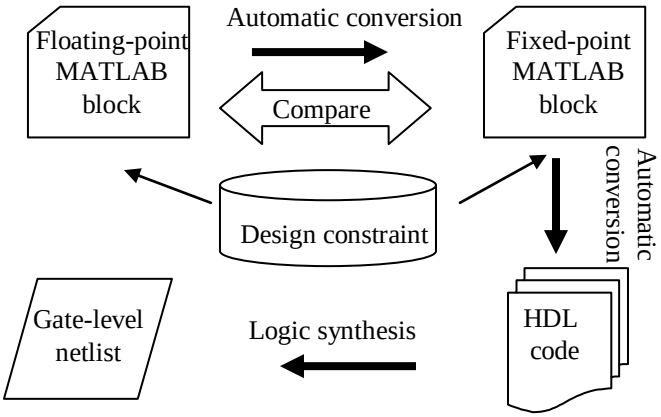


Figure 9. Digital filter design flow.

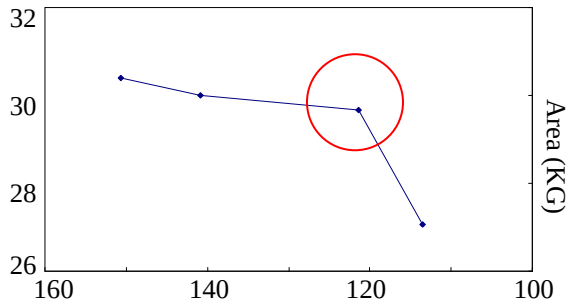


Figure 10. Trade-off analysis results.

was satisfactory, an automatic conversion into fixed-point operation was performed in order to obtain the respective hardware design, where the fixed-point operations were described as fixed integer and decimal parts of a bit sequence.

Quantization error arises when approximating real numbers with a variable decimal point, for example, when comparing the results of floating-point and fixed-point calculations. The magnitude of this error must be examined in hardware design, and MATLAB can be used for rapid and straightforward computation of this parameter. Fig. 8 shows the results for the quantization error associated with fixed-point and floating point operations, where the fixed-point simulation results are shown in the top panel, the floating-point simulation results are shown in the middle panel, and the quantization error is shown in the bottom panel. Clearly, the quantization error is within the interval of ±3/1000. Additionally, MATLAB can display and compare three simulation results simultaneously on the same screen, which is convenient when verifying the results.

Here, we describe a case study of designing a Kalman filter, which is often used in car navigation systems as well as in video processing. First, the Kalman filter algorithm was developed in M. Fig. 9 shows the hardware design flow, where it is clear that once developed, the floating-point model in M at the algorithm development phase automatically proceeds to the hardware implementation phase in the form of a gate-level netlist.

In general, the two major constraints in this case are speed (frequency) and cost (device area) for digital filter design, in other words, the desired results are high speed and low cost. However, those two constraints are in a trade-off relationship, and they must be investigated by changing a number of parameters and performing several design iterations with each set of parameters. This design flow can be automated in order to promptly obtain trade-off analysis results in a simple manner by examining various combinations of constraints. Fig. 10 shows the analysis results for this case study, where the design constraints were a frequency over 100 MHz and an area smaller than 30 KG, and it is clear that the optimal result is attained at a frequency of around 120 MHz Thus, the M-based design method can be used for performing faster and easier trade-off analysis without any changes to the M-based description at the algorithmic level.

C. Case study of FIR filter design and verification

Next, we introduce a case study where M-based design is used in the verification of a finite impulse response (FIR) filter, which is one of the most commonly utilized digital filter types in mobile phones, receivers, and other devices. This time, the target of the design is a 16-dimensional FIR low-pass filter with given specifications. The main specifications are a sampling frequency of 44 kHz and a cut-off frequency of 2 kHz. Other filter factors, such as response type, frequency specification and amplitude, can be easily set in MATLAB.

In the M-based design environment, if a parameter is defined as a user variable, then it is considered that the parameter can be changed freely, and therefore the design

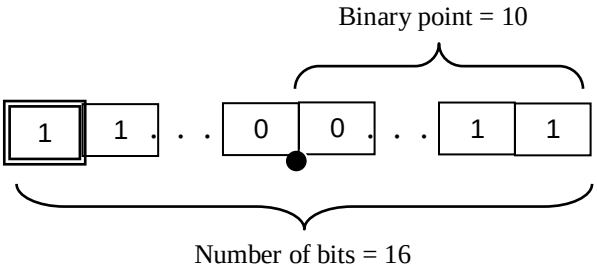


Figure 11. Setting the number of input bits.

TABLE V. TRADE-OFF ANALYSIS BASED ON THE NUMBER OF INPUT BITS

Number of bits	**8**	**12**	**16**	**20**
Freq. (MHz)	142	136	133	110
Area (Gates)	923	1170	1215	1890
Slices (pieces)	67	83	103	144

specifications can be modified. Here, the process involves designing the filter hardware and focusing on the number of input bits, which is one of the most important variables. As described earlier, the input consists of a number of bits and a binary point. If these parameters are provided as data with variable length and decimal point position, they can be changed. In this regard, Fig. 11 shows an outline of this configuration. At the filter algorithm development phase, input signals are processed by floating-point operations, while fixed-point operations are needed in the case of hardware design. Therefore, the number of input signal bits must be fixed in the latter. Although the signal accuracy becomes higher as the number of input bits increases, the size of the designed circuit increases, and vice versa. As described in the previous sections, trade-off analysis is performed with respect to the number of input bits. Table 3 shows how the circuit performance varies based on the number of input bits.

At the design specification phase, the number of input bits was set to 12 or 16, and the corresponding results are shown in Table 3. As clear from the table, the effect of changing the number of input bits on the operation frequency and the area was negligible, and since the result with 16 bits indicates higher accuracy, the number of bits was set to 16. In this case, the data length (number of bits) was 16 bits, and the position of the decimal point (binary point) was set to 10 bits. Based on the trade-off analysis performed by applying the M-based design method, it can be concluded that the analysis results can be obtained promptly, thus reducing the amount of time required for development, and the design specifications can be verified with high accuracy [14].

VI. DISCUSSION

We have demonstrated the possibility for automating all processes from algorithm verification to system description and detailed design of hardware in a consistent manner by using a design method based on the M language. As a result, high-quality system-level design can be accomplished in a

comparatively shorter period of time than with conventional tools. The realization of such functionality has been approached from various directions in the past, and currently the most commonly used design techniques are based on C-based languages (high-level synthesis techniques). This design approach eliminates the need for rewriting C source code into HDL code and shortens both the development time and the simulation time.

When a design environment based on a C-based language is constructed, two problems occur that strongly depend on the high-level synthesis tool and the possibility for reusing properties from current and previous designs. Using a C-based language for the original design produces increasingly complex code, which often entails an increase in development time. To resolve these problems, we adopted a design technique based on the M language, which facilitates the construction of simulation environments.

The adoption of this method is expected to alleviate the difficulties associated with design based on C-based languages, which is implemented only by skilled software engineers, and design based on manual HDL coding, which is implemented only by skilled hardware engineers for embedded system design. This method allows both software engineers and hardware engineers to perform verification and debugging at the algorithmic level. It also allows for specifications to be promptly and accurately implemented as hardware modules by system designers and algorithm designers.

On the other hand, the conversion of floating-point operations into fixed-point operations is one of the important issues which must be resolved in order to ensure the seamless integration of all steps from algorithm development to hardware development. Therefore, in this work we intentionally adopted techniques based on the M language. However, the support for automation for this language is not at a satisfactory level, whereas design based on C-based languages is 100% automated.

Since the results of these trials include application-specific integrated circuit (ASIC) optimization, the same results or trends might not be observed when designing different types of hardware. However, it is clear that TAT can be consistently reduced. It is possible that without strong dependence on high-level synthesis tools, M-based design can be regarded as a process of combining block modules from a database. We believe that such algorithm modules can accelerate and improve the reuse of properties from current and previous designs while providing the possibility for visualization of the design flow.

VII. CONCLUSION

Simulation environments for hardware design can be built in a straightforward manner by using M-based design, and simulations can be readily conducted in such environments. The M language was also found to improve the reusability of intellectual property from current and previous hardware designs. This design method supports automatic conversion from floating-point to fixed-point operations, which is highly useful in hardware design.

In addition, the design method can be used to analyze trade-off tendencies in circuit performance by taking into account the input signal and the operating frequency. Therefore, M-based design can increase the productivity of embedded systems designers. In the design of complex embedded systems, this design method might prove highly effective for shortening the development period and reducing the incidence of serious flaws.

VIII. FUTURE WORK

Currently, the adoption of the M language for all hardware development is not possible since advanced functions, complex control logic, a complete environment for fulfilling equivalency checking, and formal property checking of automatically generated HDL code are as yet unavailable. We believe that when such functionality is implemented, hardware development time can be dramatically reduced and co-simulation and co-verification can be performed by assigning equivalence relations between M-based modules and HDL modules. We plan to continue our investigations in this direction by using M-based design.

REFERENCES

[1] W.E. Leonard and W.S. Levine, "Using MATLAB to Analyze and Design Control Systems," Benjamin-Cummings Publishing, 1995.

[2] R.C. Gonzalez, R.E. Woods, and S.L. Eddins, "Digital Imaging Processing Using MATLAB," Prentice Hall Press, 2007.

[3] J.W. Woods, "Multidimensional Signal, Image, and Video Processing and Coding," 2nd edition, Academic Press 2011.

[4] O. Schliebusch, H. Meyr, and R. Leupers, "Optimized ASIP Synthesis from Architecture Description Language Models," Springer, pp.1-21, 2007.

[5] P. Coussy, G. Gajski, A. Takach, and M. Meredith, "An Introduction to High-Level Synthesis", Special issue on High-Level Synthesis, IEEE Design and Test of Computers, Vol.26, No.4, 2009.

[6] G. Martin and G. Smith, "High-Level Synthesis: Past, Present, and Future," IEEE Design & Test of Computers, Vol.26, No.4, pp.18-25, 2009.

[7] Z. Zhang, Y. Fan, W. Jiang, G. Han, C. Yang, and J. Cong, "AutoPilot: A Platform-Based ESL Synthesis System," in High-Level Synthesis: From Algorithm to Digital Circuit, Ed. P. Coussy, A. Morawiec., 2008.

[8] K. Wakabayashi, "C-based synthesis experiences with a behavior synthesizer, "Cyber"," in Proc. of DATE'99, pp.390-393, 1999.

[9] B.K. Gunturk, J. Glotzbach, Y. Altunbasak, R.W. Schafer, and R.M. Mersereau, "Demosaicking: color filter array interpolation," IEEE Signal Processing Magazine, Vol.22, pp.44-54, 2005.

[10] J. Astola and P. Kuosmanen, "Fundamentals of Nonlinear Digital Filtering," Boca Raton, FL: CRC, 1997.

[11] P.P. Vaidyanathan, "Multirate Systems and Filter Banks," Prentice Hall 1993.

[12] Image web site: http://users.ecs.soton.ac.uk/msn/book/new_demo/

[13] A. Matsuda and S. Baba, "An Automated Design Flow for Image Processing Filter in Embedded Systems," in Proc. of the 9th IEEE International Conference on ASIC (ASICON2011), Vol.1, pp.768-771, 2011.

[14] A. Matsuda and S. Baba, "Case Study of Filter Design in Embedded Systems," in Proc. of the 2011 IEEE Student Conference on Research and Development (SCOReD), pp.271-276, 2011.