Application on the Hardware/Software Co-simulator; Implementation of Multi-stage, Multi-rate 2-D filter

Yukiko Takanishi
Faculty of System Design
Tokyo Metropolitan University
Hino, Asahigaoka,
191-0065, Japan
takanishi-yukiko@sd.tmu.ac.jp

Yuichi Nakamura
System IP Core Research
NEC Corporation
Kawasaki, Nakahara-Ku, Shimonumabe,
211-8666, Japan
yuichi@az.jp.nec.com

Takao Nishitani
Faculty of System Design
Tokyo Metropolitan University
Hino, Asahigaoka,
191-0065, Japan
nishitani@tmu.ac.jp

Abstract— A functional expansion of a hardware-software co-simulator, using "Simulink" on PC and an FPGA emulator board, is realized for the purpose of real-time HDTV signal processing. In the proposed co-simulator, the emulation is carried out by using a set of raster scanning data processing, instead of the frame-based processing which is suitable for processing "Simulink" block. In addition, a visual verification approach in terms of processing delay between Simulink blocks is introduced for adjusting the connection of these blocks within the emulator.

I. Introduction

In recent years, a number of consumer electronic products related to video signal processing is increasing and also their life-time is decreasing rapidly. Therefore, many kinds of SoC (System-on-Chip) have to be developed within a limited development period. In order to shorten the development cycle including time-periods to introduce new functions as efficient IP-cores, a hardware-software co-design approach is useful for designers, because this approach can simultaneously optimize both system level architecture and LSI circuit design. Otherwise, the iterative approach of architecture optimization followed by circuit optimization is required and needs a long development period.

There are several hardware-software co-simulators now available. For example, the "EDA Simulator Link" [1] is a co-simulator, which provides a verification interface between "Simulink" [2] and a HDL simulator. However it includes hardware simulation by software, and therefore, needs a lot of time for the simulation. A rapid prototype system [3] is another example, but it cannot realize partial emulation and is very expensive.

Our developed system [4] provides the interface between "Simulink" and a FPGA emulator board, where Simulink on PC and an actual emulation for S-Function (a block in a Simulink description) by the FPGA are used. As the FPGA emulation can run in real-time, simulation time becomes short and timing problems can be clearly evaluated. However, our original approach is not suitable for HDTV signal processing, because HDTV frame memories are required in the FPGA for Simulink interface, but the FPGA is not suitable for implementing large capacity RAMs. The introduction of external frame memories on a FPGA board may be acceptable, but the input and output data for circuits in the FPGA is something different from the actual circuits. As a result, the developed hardware-software co-simulator cannot be easily introduced to HDTV signal processing which becomes popular in mobile terminals like tablet computers.

This paper describes the HDTV signal processing on a FPGA emulator board by modifying the interface function between them. The approach employed is to divide one frame data in the buffer into small blocks containing several raster scanning line data for interfacing a S-Function block in a Simulink description and the board. When the corresponding S-Function becomes active, the S-Function iteratively sends line data to the FPGA and fetches the processed data from the FPGA, until the whole output buffer for a frame picture becomes full. Therefore, this simulator modification needs the system level architecture re-optimization based on the hardware side requirement in the co-simulation. The architecture optimization should be a line-based one, instead of a frame-based one.

The other functional expansion is to improve verification function, when two circuits for adjacent S-Functions are verified and these circuits are combined. As the processing is based on block-based one in every verified circuit, different processing cycle delay may cause some trouble, if the adjacent blocks have some feed-forward/feedback connections. The proposed approach makes such processing cycle delay visible by using a monitoring S-Function and some modification on an input image.

The paper first reviews the reported co-simulator in section 2. The proposed system is described in section 3. In section 4, the system re-optimization is carried out by using an example. Consideration on the IP-Core circuit implementation of the example is described in section 5. Section 6 shows some impact from a system level architecture modification to the hardware side implementation.

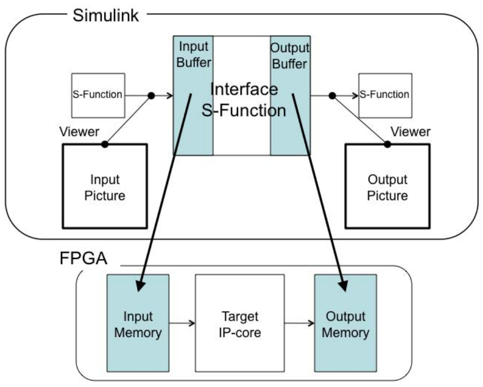

Fig. 1. The reported co-simulator structure

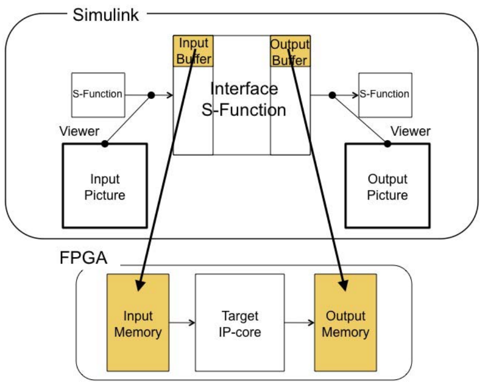

Fig. 2. The proposed co-simulator structure

II. The reported hardware-software co-simulator

The hardware-software co-simulator using "Simulink" on PC and a FPGA emulator board [4] has been developed so as to realize seamless connection between them. "Simulink" is widely used in the development of a signal processing system. It acts as a state machine controller for block diagrams composed of "S-Functions." The system level functions in all S-Functions can be written in C-language and LSI circuits are written in Verilog-HDL in the FPGA. Both S-Functions on PC and LSI circuits on the FPGA are programmable for modifications. Therefore, it is easy to find problem and to improve system performance by using this tool. Figure 1 shows the reported co-simulator block structure and the following procedure is employed:

1. The S-Function to be implemented by hardware circuits should be replaced with an interface S-Function which sends and receives data from and to the FPGA board.
2. All data in the input buffer of the interface S-Function has to be prepared for activating by the Simulink controller.
3. The input data in this S-Function are transferred to the FPGA and stored into the input memory within the FPGA.
4. When the input memory becomes full, the FPGA automatically starts processing of the data stored in the input memory.
5. All processed data is stored into the output memory within the FPGA.
6. When the output memory becomes full, the FPGA sends a notification signal to the interface S-Function.
7. The interface S-Function starts to fetch the data in the FPGA output memory and stores them into the output buffer in the interface S-Function.
8. When the output buffer in the interface S-Function becomes full, the Simulink controller terminates this S-Function and activates the following S-Function.

By using the above structure for co-simulator, the following merits are obtained.

1. The realized seamless connection eliminates a special test bench for verifying the hardware circuits.
2. High speed processing in the FPGA emulation has a great advantage over software simulation of the RTL codes.
3. It is easy to find bugs in a system by using a standard monitoring S-Function of "Display" in "Simulink. The signal brunch in a Simulink description can be easily observed by putting Display before and after the interface S-Function which is shown in Fig. 1.
4. The monitoring blocks also supports to check the processing accuracy and dynamic range of signals in the hardware emulation, because some shortages of these parameters result in noisy pictures.
5. It is possible to take a step-by-step SoC development when the SoC functions are written by Simulink description. When two verified circuits for the corresponding S-Functions in the above Simulink description are finalized, they can be merged into one functional circuit of an expanded S-Function in the Simulink description.

III. The proposed co-simulator

A. *Suitable for HDTV signal processing*

Many merits are obtained from the reported co-simulator. However, the reported approach is not suitable for HDTV signal processing, because HDTV frame memories are required within the FPGA. Recent largest FPGA chip such as Xilinx Virtex-6 is not enough for realizing one HDTV input and output frame memories for RGB components.

In order to overcome the memory shortage in the FPGA, a modified approach on the interface function between the PC and the FPGA emulation board is proposed for the HDTV signal processing. The approach employed here is to divide one frame data in the S-Function buffers into small blocks, shown in Fig. 2. The proposed structure in the FPGA is realized to reduce the input and output memory capacity.

The role of the interface S-Function is modified to execute the iterative operations, which are composed of 4 operations: data transfer of a block within the input buffer in the S-Function, waiting for the notification from the FPGA, fetching the processed data from the FPGA and storing these data into the output buffer in the S-Function. When the output buffer in the S-Function becomes full, the Simulink controller automatically activates the following S-Function. The seamless connection between "Simulink" and the FPGA emulator board is maintained as well as the reported co-simulator. Note that the HDTV frame memory is required only in the S-Function in PC side. As a result, almost the same merits of the original co-simulator described in section 2 are still obtained, because the interface S-Function works just like that in the reported simulator in terms of the relationship to other S-Functions.

One appropriate method to divide frame data in the buffer into small blocks is to divide a picture into blocks composed of raster scan lines, because input pictures from a camera are normally fed in the raster scan order (from left to right and from top to bottom). How many lines consist of one block depends on what kind of algorithm is applied to, but it is important to be consistent in the same structure among all S-Functions in order to connect the developed LSI circuits for the corresponding S-Functions. In case of an employed block which is composed of a single line, this approach can reduce the required memory in the FPGA to about 1/1000 capacity memory, compared with those used in the reported approach for HDTV processing.

Further reduction of required internal memory capacity in the FPGA is possible to change the employed algorithm from a frame-based consideration to a line-based consideration. For example, there is a case that the interim results must be stored into a temporal memory. It is useful for the algorithm implementation to remove these memories. Such kind of considerations contributes a small die size and low cost SoC implementation. In addition, the reduction of memory capacity is not a limited solution to FPGA implementation.

B. *Introduction of a visual verification function*

One big problem to employ the line-based algorithm in a case is that the circuits become to generate processing cycle delay within a line more easily than the frame-based one, although the frame-based one may also generate a processing cycle delay less than one line. Such a processing cycle delay becomes a heavy problem when two adjacent circuits corresponding to the S-Functions are verified by the proposed co-simulator and have some interactions

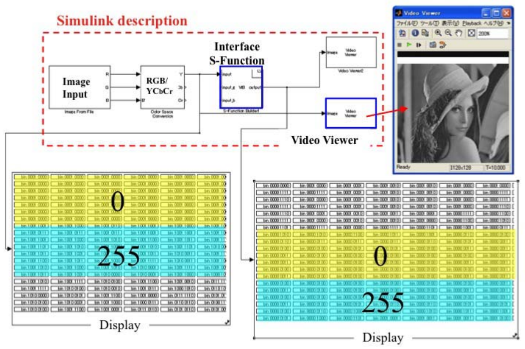

Fig. 3. The interface structure

such as a feed-back/feed-forward connection between the two. These processing cycle differences are hard to check by the viewer, shown in Figs. 1 and 2, because every S-Function output is synchronized with the beginning of the frame. The processing cycle delay cannot be observed.

A visual verification approach is newly established in the proposed system by using S-Functions provided by "Simulink". This approach makes use of a zero pixel sequence followed by a sequence with maximum value pixels, both of which replace actual pixel values in the input picture at the beginning area of the frame. These sequences may have a few lines and the sequence acts as a marker. A sequence, which is a plenty of zeroes in the line direction, resets the hardware and the maximum value sequence indicates huge changes based on zero reset hardware. The example is shown in Fig. 3, where one of the monitoring S-Functions "Video Viewer" shows the "Lena" picture with black and white bands at the top area. Two "Display blocks" are used before and after the S-Function to be verified. The Display block in the input side shows the marker area and the other Display block in the output side shows the marker area shifted by the processing. This is because the output memory in the FPGA records all of the data appeared at every clock period in this monitoring mode. In this figure, these Display blocks employ pixel-based monitoring mode. In addition, pixel values can be observed at "Video Viewer." The actual processing delay can be evaluated by the final pixel position at the marker area. Also, a marker sequence is composed of 8 zero lines. Note that the S-Function of RGB to YCbCr conversion in the standard Simulink library uses the ITU-R BT.601 rule which asks that the conversion result is always larger than 16 in 8 bit representation. Therefore, the pixel values in the marker area never become zero, but 3 in our down-sampling filter.

Consider about a 2-D FIR filtering function by this approach. From the vertical position shift from the top line, the pixel-based processing cycles can be observed, when the filter moves from the left to the right. The image processing by Simulink in this direction causes the movement from the top to the bottom in the display. The insertion of one word register by a system clock causes one line shift of the black band from the top line. When the line direction

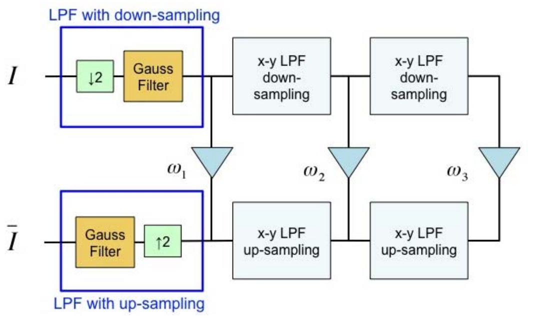

Fig. 4. The multi-stage and multi-rate MSR filter

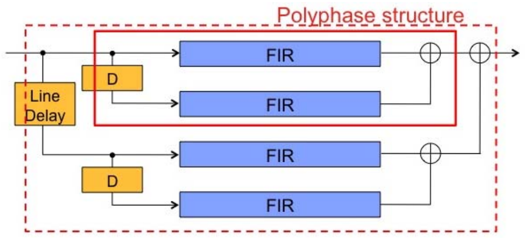

Fig. 5. The 2-D polyphase filter

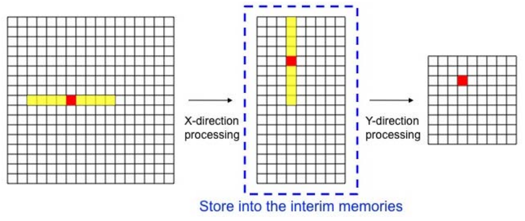

Fig. 6. The processing using the X-Y separable filter

delay is expected, black and white vertical bands should be added. In order to find the shift lines, the filter coefficient should be set to 1/MxM where the M-th order filter is employed. Then, the output position can be found by searching the line position of 255/M value increase.

IV. The system efficiency on the image processing

A. A picture enhancement system

A MSR (Multi-Scale Retinex) picture enhancement system [5] is evaluated by the proposed system. It is an expanded system of SSR (Single-Scale Retinex), based on the human retina and cortex activities [6]. The SSR calculation first extracts color contrast signals in human measure by the following equation.

$$R_i = \log I_i(x,y) - \log \overline{I}_i(x,y) \tag{1}$$

where I_i is the image distribution in the i-th color spectral band, $\overline{I_i}$ is the estimated illumination component, and R_i is the output of SSR. $\overline{I_i}$ can be replaced by a spatially averaged luminance of a pixel at (x,y) by a 2-D Gaussian filter. The final output is obtained by clipping the valid

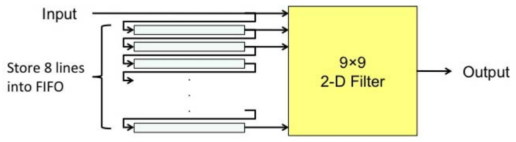

Fig. 7. The processing using the direct 2-D filter

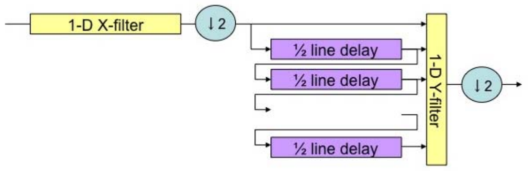

Fig. 8. The line-based X-Y filter strucure

pixel values of the R_i and by normalizing their levels from 0 to 255. The MSR algorithm is defined by

$$R_{M_i}(x,y) = \sum_{i=1}^{N} \omega_n R_{n_i}(x,y) \tag{2}$$

where N is the number of scale, in the case of Fig. 4, $N=3$. R_{n_i} is the SSR output of the n-th scale. ω_n is the weighting coefficient of the n-th scale, where the sum of all ω_n equals 1, and R_{M_i} is the MSR output.

B. Multi-stage and Multi-rate signal processing for MSR

In order to determine the estimation of the illumination component, the original enhancement system [5] employs iterative X-Y separable Gaussian filters with down-sampling and up sampling instead of a 2-D filter. The multi-stage and multi-rate signal processing can reduce the region to be filtered. The introduction of the X-Y separable filters is due to the lower computational amount than the corresponding direct 2-D filters. In addition, all multi-rate filters with sub-sampling are designed by a pair of down and up sampling with polyphase structures [7] shown in Fig. 5 for the down-sampling filter implementation. The processing in Fig. 5 is carried out by the line direction polyphase structure followed by the vertical scan direction. The polyphase structure [8] allows a low computational cost by removing cast-away samples in the down-sampling filters. This approach reduces the total calculation amount by 1/4. The optimization mentioned above is architecture level one in terms of the reduction of arithmetic operations.

However, the original X-Y separable filter [4] cannot be employed, even if the amount of the arithmetic operations is much smaller than that of the direct 2-D filter for the following reasons. In the X-Y separable filtering approach shown in Fig. 6, the interim results of the X-direction 1-D filtering is required to store into a temporal memory. This is because the Y-direction processing starts after all X-direction processing is completed. In case of this enhancement algorithm, 2 HDTV frame memories in total are required for the interim results. In addition, every

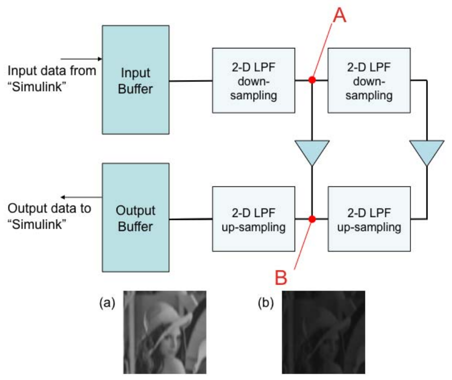

Fig. 9. The experimental visual verification tool

filter output data has to be saved in the memories until the up-sampling process asks for these results in the multi-scale processing.

On the other hand, a direct 2-D filter structure [9] is depicted in Fig. 7, where line delays and a part of the 2-D filtering operations is connected. This approach seems to be smarter than the X-Y separable approach, even if the amount of multiplications and additions becomes huge.

The structure leads to the other X-Y separable filter implementation, shown in Fig. 8. A line processing is carried out by only one 1-D FIR filter, and the X-direction length is halved due to the down-sampling effect. In order to realize the Y-direction filter, these output samples should be stored in a set of line memories, shown in Fig. 8. The required operation in the Y-direction filter calculation becomes only multiply-and-add.

As a result, the most suitable implementation for consumer products is to employ the line-based X-Y separable filter with the structure shown in Fig. 8.

C. *The experimental results*

The developed co-simulator is composed of a PC, having AMD Athlon. 64 X2 Dual processor at 2.00 GHz and the FPGA board having a FPGA chip of Xilinx Virtex-4 XC4VLX200. Thanks to the proposed line-based I/O buffers in the FPGA chip, the resident portion requires only 43 4-input LUTs, 40 Slice Flip-Flops and 2 BRAMs, all of which result in 1% usage in the FPGA resources, as shown in Table 1. Two thousand BRAMs required in the original co-simulator are replaced with only 2 BRAMs.

Table 2 shows the complexity differences among 3 kinds of filters on 3-scale processing of HDTV picture: the X-Y separable filter with the interim memories; the X-Y separable filter employed the line-based approach; and the direct 2-D filter. In case of both kinds of X-Y separable filter, 30 DSP48 modules as multipliers are employed. The Direct 2-D filter employs 51 DSP48 modules in the FPGA which contains 18x18 bit multipliers, 36 bit adders, registers and so on. The 2-D filter needs more multipliers than X-Y separable one as expected. Note that 12 of 51 DSP48 modules are used only for 30 bit adders, which calculate a sum of multiplied results between a set of pixel values and filter co-efficient values. The number of DSP48 used as multipliers in the X-Y separable filter implementation and that of the direct 2-D filter becomes 9 differences. On the other hand, there is no difference among 3 kinds of filters in the number of 18 kbit block RAM modules. However, the implementation of the HDTV signal processing is impossible for the reported X-Y separable filter because of the need of the interim memories. The required memories for this purpose are significantly increased for HDTV processing. In case of using the line-based X-Y filter structure and direct 2-D filter structure, no more memory capacity is required. Note that the line-based X-Y separable filter requires half-length delay lines while the direct 2-D filter requires full-length delay lines. However, the same number of delay lines is required, which uses 18 kbit block RAM modules. It is possible to reduce a number of the memory modules in order to modify multiple delay lines to a single word delay line with wide word. In such a case, the X-Y separable filter implementation has an advantage over the direct 2-D filter in terms of RAM requirements in addition to the number of multipliers. As described above, the line-based structure has the advantage over other structures.

In Fig. 9, an experimental visual verification tool is demonstrated, where monitors in Figs. 9(a) and 9(b) shows some timing problems in the actual use. In the proposed co-simulator, processing cycle delay at point A and point B, corresponding to the Figs. 9(a) and 9(b) can be observed in the following way. Three filters located from the input port to the third filter and the first filter is separately designed in the FPGA. These two filtering FPGA can work without any problem, because there are no feed-back/feed-forward connections. However, the precise cycle delay adjustment between two observation points should be required for the addition of 2-scale signals, because this addition makes a kind of feed-forward processing.

Based on the above observation, some processing cycle

TABLE I
LOGIC UTILIZATION

Logic Utilization	Used	Utilization
4 input LUTs	43	1%
Slice Flip Flops	40	1%
BRAM	2	1%

TABLE II
THE COMPLEXITY DEFFERENCE ON HDTV PICTURE PROCESSING

	DSP48s	BRAMs
X-Y separable filter		
- Original	30	-
- Line-based	30	32
Direct 2-D filter	51	32

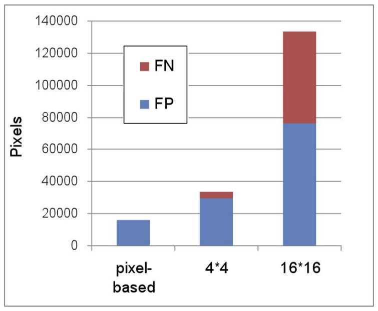

Fig. 10. The evaluation

delay has to be put on the path from point A to the adder.

V. Impact from the algorithm modification

The line-based input/output method decreases a hardware amount, but it is still large for consumer products in some cases. Indeed, the motivation of the co-simulator development is based on a single chip implementation on the foreground segmentation [10] for gait recognition or extremely low bit rate picture coding. For these applications, a clear foreground separation performance is required, even if a people wearing a dark T-shirt is walking through the dark area. For such a case, the picture enhancement function is required as a pre-processing of the foreground segmentation function [11]. Therefore, the picture enhancement approach described in section 4 should be light enough for the pre-processing.

Foreground segmentation [10] uses 4x4 blocks as the minimum segmentation accuracy. Therefore, it is possible to reduce the operation by reducing picture size of HDTV. The approach employed here introduces a block-based MSR. The following processing is carried out. The average of the divided 4x4 block of the HDTV frame are first calculated and a thumbnail picture is formed by such average pixel value. Then, the picture enhancement algorithm is applied to the thumbnail picture. The required hardware resource of the picture enhancement algorithm is reduced to 1/16 depending on the above processing.

Figure 10 shows the results of the foreground segmentation employed the MSR picture enhancement as a pre-processing. Figure 11 shows the evaluation by the segmented results employed block-based enhancement and pixel-based one by comparing a number of False Positive (FP) pixels with that of False Negative (FN) pixels. The proposed 4x4 block-based approach shows better than 16x16 block-based approach, but shows slightly worse than pixel-based one. Note that there is no more than 1% difference between these processing, because 1920x1080 HDTV picture is large. As a result, the 4x4 block-based method has the advantage of processing efficiency.

VI. Conclusion

The functional expansion of the reported hardware-software co-simulator [4] is proposed for real-time HDTV signal processing. This expansion enables HDTV signal processing and also introduces the visual cycle delay measurement approach which is an important function for realizing a feed-back/feed-forward situation between two functional circuits so as to adjust the two signals in the same phase. An example of the MSR picture enhancement system is used to show the impact between architecture optimization and circuit optimization by our hardware-software co-simulator.

Acknowledgements

The authors thank Mr. Yuichi Muraki for his help in the computer simulation on a new idea based on the low computational picture enhancement algorithm.

References

[1] The MathWorks, "EDA Simulator Link" http://www.mathworks.com/products/eda-simulator/

[2] The Math Works, "Simulink" http://www.mathworks.com/products/simulink/

[3] Cadence, "Rapid Prototyping Platform" http://www.cadence.com/products/sd/rapid_prototyping/pages/default.aspx

[4] Yukiko Takanishi, et al., "Hardware/software co-simulator for ASIC DSP chips," Proc. of MWSCAS 2010, pp. 809-812, Aug. 2010.

[5] Takeshi Okuno, et al., "Efficient Image enhancement Algorithm Using Multi-Rate Image Processing," IEICE Trans. Fundamentals, Vol. E-93A, No. 5, May. 2010.

[6] Z.Rahman, et al., "Properties and performance of a center/surround retinex," IEEE Trans. Image Process., Vol. 6, pp. 451-462, 1997.

[7] Marco Winzker, et al., "VLSI Chip Set for 2D HDTV Subband Filtering With On-Chip Line Memories," IEEE Journal Of Solid-state Circuits, Vol. 28, No. 12, Dec. 1993.

[8] Sanjit K.Mitra, *Digital Signal Processing*, McGraw-Hill Higher Education, 2001.

[9] K. Parhi, *VLSI Digital Signal Processing systems -design and Implementation*, A Wiley-International Publication, 1999.

[10] Hiroaki Tezuka, et al., "Multiresolutional Gaussian Mixture Model for Precise and Stable Foreground Segmentation in Transform Domain," IEICE Trans. Fundamentals, Vol. E92-A, No. 3, pp. 772-778, March. 2009.

[11] Yuta Akasaka, Yukiko Takanishi, et al., "High Precision Contour Detection on Computationally Efficient Silhouette Extraction," Proc. of ITC-CSCC 2010, pp. 288-291, July. 2010.