Design of an Image Interpolator for Low Computation Complexity

Young-Hyun Jun*, Jong-Ho Yun*, Jin-Sung Park, and Myung-Ryul Choi***

Abstract: In this paper, we propose an image interpolator for low computational complexity. The proposed image interpolator supports the image scaling using a modified cubic convolution interpolation between the input and output resolutions for a full screen display. In order to reduce the computational complexity, we use the difference in value of the adjacent pixels for selecting interpolation methods and linear function of the cubic convolution. The proposed image interpolator is compared with the conventional one for the computational complexity and image quality. The proposed image interpolator has been designed and verified by Verilog HDL(Hardware Description Language). It has been synthesized using the Xilinx VirtexE FPGA, and implemented using an FPGA-based prototype board.

Keywords: *Interpolation, Interpolator, Cubic convolution , Linear function*

1. Introduction

There are various images and video formats which are depending on the target applications. When the resolution of images and videos is pre-encoded and stored, the characteristics of the target devices are not considered. An efficient way to overcome this problem is to use an interpolation. It is required for resolution conversion to adapt to the characteristics of a particular displays device.

Standard approaches of interpolation fit the original discrete data with a continuous model and resample the interpolation function on a new sampling point.

Popular methods of interpolation by convolution include nearest neighbor interpolation [1], bilinear interpolation [2], and cubic convolution interpolation [3]. The nearest neighbor interpolation is to assign the pixel closest to the newly generated address as the output pixel. It is a simple method of interpolation and is very expensive to implement but visual blockiness, also known as the jaggies, can be seen in the output [1]. The most widely used method is the bilinear interpolation which is regarded as the linear function. The newly generated pixel is a weighted sum of the four nearest pixels. Cubic convolution interpolation sharpens an image but requires more computational complexity.

This paper proposes an image interpolator for low computational complexity. We use the linear function of the cubic convolution and the difference in pixel value for selecting interpolation methods. In section 2, we describe the related work briefly. We describe the proposed method in section 3. In section 4 and 5, we show the simulation results and synthesis results. The conclusion is made in section 6.

Manuscript received October 13, 2006; accepted December 19, 2006.
This paper was supported by Brain Korea 21(BK21).
Corresponding Author: Myung-Ryul Choi
* Dept. of EECI, Hanyang University, Ansan, Korea
(impword@asic.hanyang.ac.kr, sfw1179@asic.hanyang.ac.kr, choimy@asic.hanyang.ac.kr)
** CEN Co.,Ltd, Ansan, Korea(pjs@asic.hanyang.ac.kr)

2. Related Work

2.1 Nearest Neighbor Interpolation

Nearest Neighbor Interpolation is a simple method of interpolation. It simply selects the values of the generated pixels, and does not consider the values of other neighboring pixels at all. This interpolation is very expensive to implement, but has visual blockiness [1].

2.2 Bilinear Interpolation

Bilinear interpolation determines the newly generated pixel from the weighted average of the four closest pixels to the specified input coordinates. The four closest pixels are NW(North West), NE(North East), SW (South West), and SE(South East). Bilinear method uses three linear interpolations. The first linear interpolation evaluates the first interpolated value(A) from the values at NW and NE. In the same way, linear interpolation at the second interpolated value(B) evaluates from the values at SW and SE. The new pixel value(C) is then linearly interpolated from the two values previously obtained. It is represented by Figure 1 and Equation (1).

$$\begin{aligned} A &= NW + X_x(NE - NW) \\ B &= SW + X_x(SE - SW) \\ C &= B + Y_y(A - B) \end{aligned} \tag{1}$$

Here, X_x is the Horizontal distance, and Y_y is the vertical distance between the newly generated pixel and neighbor pixel.

Bilinear interpolation yields a smoother image than nearest neighbor interpolation. Because of the three linear interpolations per pixel, bilinear interpolation requires significantly more computations than nearest neighbor interpolation does.

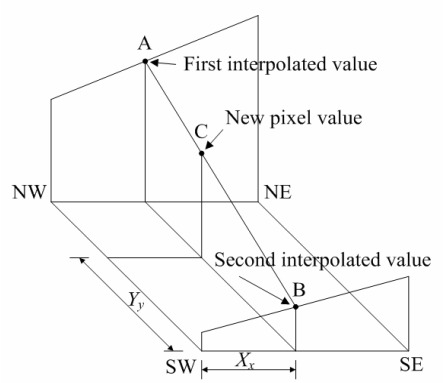


Fig. 1. Physical meaning of the bilinear interpolation

2.3 Cubic Convolution Interpolation

Nearest neighbor interpolation uses one pixel to generate a new pixel. Bilinear interpolation requires the four nearest pixels of input. Cubic convolution interpolation requires sixteen-nearest pixels to generate the output pixel.

Cubic convolution function has negative values, which brings up two important points. The first is that this function will sharpen more than nearest neighbor interpolation or bilinear interpolation. The second is that it is possible to output a negative number. The output will need to be clipped so as not to output negative pixel values.

The 1-dimensional cubic convolution function is defined as Equation (2).

$$f(x)=\begin{cases}(a+2)|x|^3-(a+3)|x|^2+1 & 0\le|x|<1\\ a|x|^3-5a|x|^2+8a|x|-4a & 1\le|x|<2\\ 0 & 2\le|x|\end{cases} \quad (2)$$

Figure 2 shows the cubic convolution function for a = -0.5, -1, and -2.

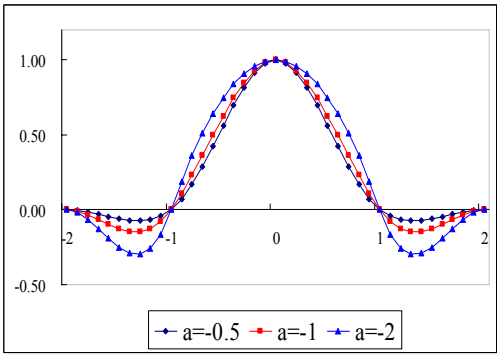


Fig. 2. Cubic convolution function

It has the performing of interpolations in two dimensions such as horizontal and vertical dimensions. Figure 3 has two dimensions interpolation of the Cubic convolution interpolation.

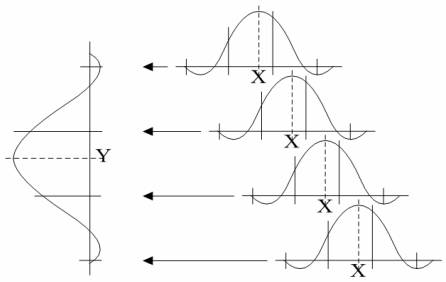


Fig. 3. 2-dimensional interpolation of the cubic convolution interpolation

3. Proposed Interpolation

In this paper, we use the linear function of the cubic convolution and the difference in value of adjacent pixels for selecting interpolation methods.

3.1 Proposed linear function

Cubic convolution interpolation has a three order function for the weighting value of pixels. In order to reduce the computation complexity, we use the linear function of the cubic convolution function to calculate the weighting value of a pixel. Figure 4 shows the proposed linear function.

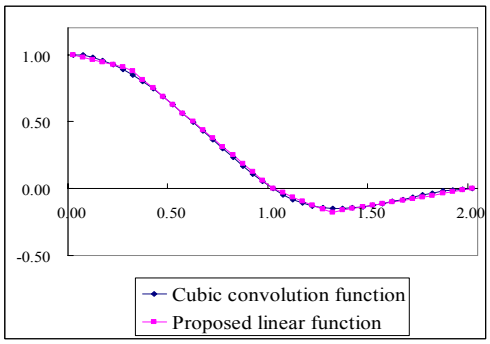


Fig. 4. Proposed linear function

We use Equation (3) to reduce the computation of the weighting value, where one multiplier and one adder are employed in one dimension. In order to use the barrel shift, the coefficient of the function compounds the power of 2.

$$f(x)=\begin{cases}-0.375|x|+1 & 0\le|x|<0.25\\ -1.25|x|+1.25 & 0.25\le|x|<1\\ -0.625|x|+0.625 & 1\le|x|<1.25\\ 0.25|x|-0.5 & 1.25\le|x|<2\end{cases} \quad (3)$$

Table 1 shows the number of operations of the comparative algorithms for the operation per one pixel. The number of operations is the number of multiplier and adder per one pixel.

Table 1. Comparison of operations per one pixel

Factor / Algorithm	Weighing factor	Neighbor pixel operations	Memory operations
Bilinear	Mul : 0 Add : 2	Mul : 4 Add : 3	Read : 4 Write : 1
Cubic	Mul : 10 Add : 15	Mul : 16 Add : 15	Read : 16 Write : 1
Proposed	Mul : 5 Add : 5	Mul : 8 Add : 7	Read : 8 Write : 1

The hardware structure of the linear function is shown in Figure 5. We employ adders and barrel shifts (BSs) instead of multipliers.

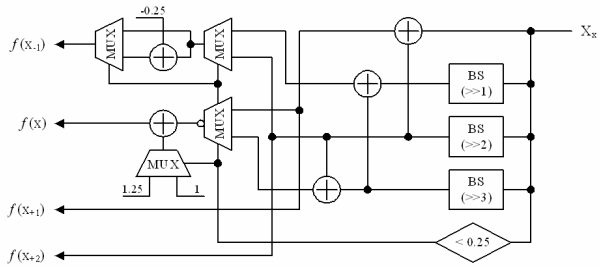


Fig. 5. Hardware structure of proposed linear function

3.2 Selecting interpolation methods

We reduce the number of a pixel to generate a new pixel using image analysis. Neighbor pixel values of the destination point are close. A difference of the neighbor pixel values is represented by Equation (4).

$$\begin{aligned} Diff(x_k) = & abs\left|a(x_k) - a(x_{k+1})\right| \\ & + abs\left|a(x_k) - a(x_{k-1})\right| / 2 \\ & + abs\left|a(x_{k+2}) - a(x_{k+1})\right| / 2 \end{aligned} \tag{4}$$

The cubic method uses sixteen numbers of neighbor pixels, but the bilinear method uses four. We apply two interpolation methods to decrease an unnecessary computation.

One interpolation method is selected by the threshold value using a neighbor pixel difference value. Figure 5 is the RMSE(Root Mean Square Error) of the up-after-down scale and down-after-up scale. We determine a threshold value as 30.

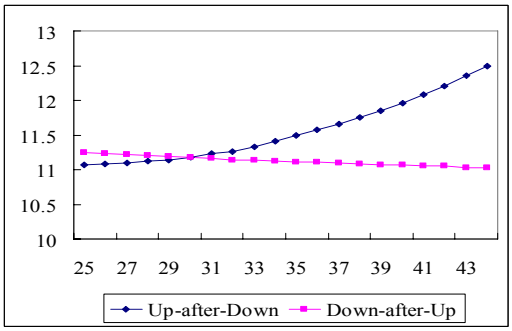


Fig. 6. RMSE of up-after-down scale and down-after-up scale

We determine the interpolation algorithm by the threshold value, as used in Equation (5).

$$f(x_k) = \begin{cases} bilinear(x_k) & diff(x_k) < threshold \\ linear\,cubic(x_k) & otherwise. \end{cases} \tag{5}$$

Figure 7 shows the processing procedure of the proposed method.

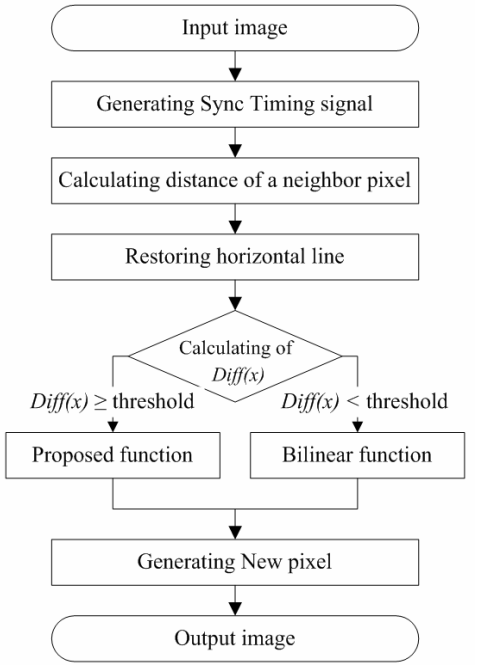


Fig. 7. The processing procedure of the proposed method

As shown in Figure 8, a zoneplate image is used to compare a frequency response of the proposed method with that of the conventional interpolation methods.

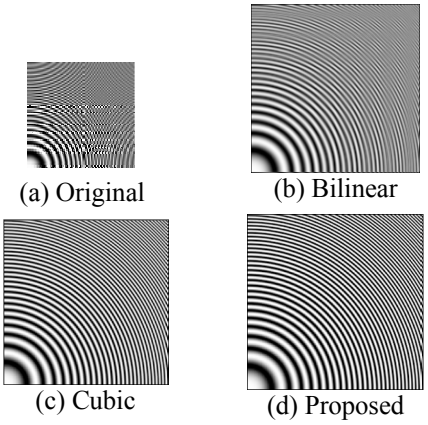


Fig. 8. Zoneplate image up(1.6 times) scale

Figure 9 is the architecture of the vertical interpolation. It consists of the sync time generator, the line memory, the distance calculation, the difference calculation, the weight generator, the new pixel generator, and the clamping.

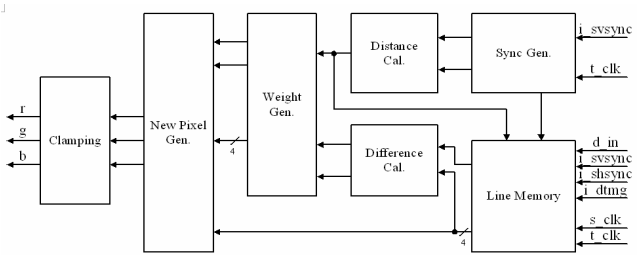


Fig. 9. The architecture of the vertical interpolation

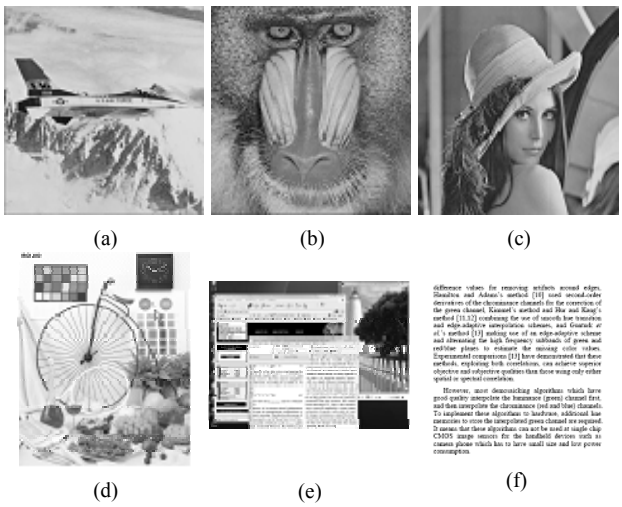
4. Simulation Results

We have compared the proposed interpolation method to the conventional interpolation methods in terms of image quality using RMSE. RMSE is represented by Equation (6).

$$RMSE = \sqrt{\frac{\sum_{\text{Image}} \left(g(x,y) - g'(x,y) \right)^2}{N}}, \quad (6)$$

where g( ) is the original image, g'( ) is the interpolated image, and N is the total number of image.

We used the six images shown in Figure 10.



(a) (b) (c)

(d) (e) (f)

(a) F16, (b) Baboon, (c) Lena,
(d) Bike, (e) Background, (f) Text

Fig. 10. Six images for RMSE comparison

The image quality of the comparative interpolations is shown in Table 2 and Table 3.

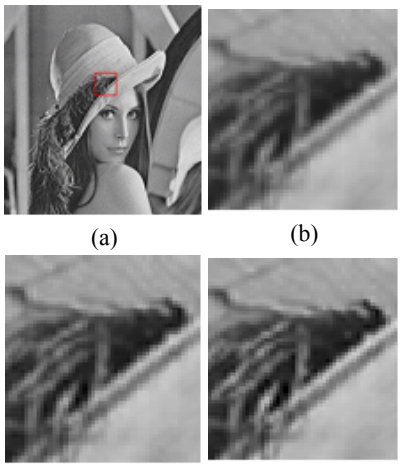
Table 2. Comparison of Up(1.6 times)-after-Down(1/1.6 times) RMSE

Test image \ Algorithm	Bilinear	Cubic	Proposed
F16	2.16	1.16	2.05
Baboon	6.73	3.57	4.85
Lena	2.10	1.14	1.95
Bike	7.18	3.60	3.57
Background	9.71	5.81	5.61
Text	21.91	14.86	9.35

Table 3. Comparison of Down(1/1.6 times)-after-Up(1.6 times) RMSE

Test image \ Algorithm	Bilinear	Cubic	Proposed
F16	5.05	4.03	4.12
Baboon	14.87	13.92	14.41
Lena	4.68	3.98	4.06
Bike	16.02	14.30	12.81
Background	22.54	20.70	20.14
Text	49.33	48.46	47.87

Figure 11, 12, and 13 show the Up-scaling(1.6 times) of the Lena, Bike, and Text image. The Up-scaling of images causes the blurring at the edge of images. The proposed method results more clear image than that obtained using conventional ones at the edge of images.

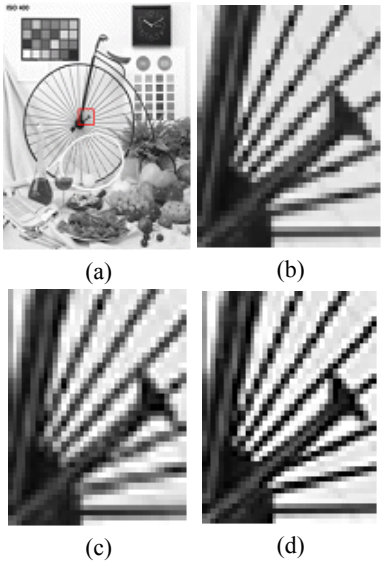


(a) (b)

(c) (d)

(a) Original image, (b) Bilinear interpolation, (c) Cubic convolution interpolation, (d) Proposed interpolation

Fig. 11. Up(1.6 times) scaling of Lena image

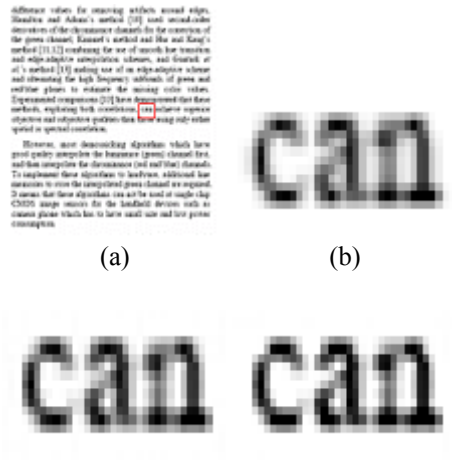


(a) (b)

(c) (d)

(a) Original image, (b) Bilinear interpolation, (c) Cubic convolution interpolation, (d) Proposed interpolation

Fig. 12. Up(1.6 times) scaling of Bike image



(a) (b)

(c) (d)

(a) Original image, (b) Bilinear interpolation, (c) Cubic convolution interpolation, (d) Proposed interpolation,

Fig. 13. Up(1.6 times) scaling of Text image

5. Synthesis results

We have designed the proposed method by using a verilog HDL and verified it through functional simulation. It has been synthesized by Synplify pro 7.3. Figure 14 shows the Block diagram of the proposed method.

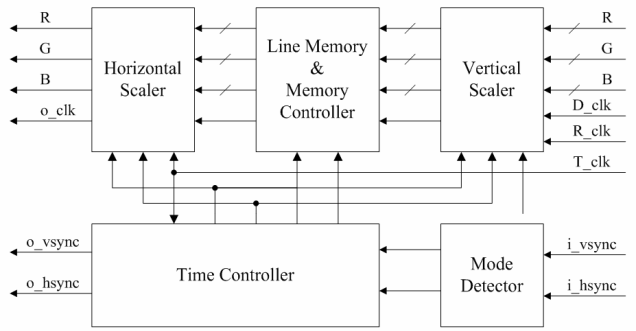


Fig. 14. Block diagram of the proposed method

Figure 15 shows the synthesis result of the proposed method using Synplify pro. Table 4 shows the number of gates by the block each.

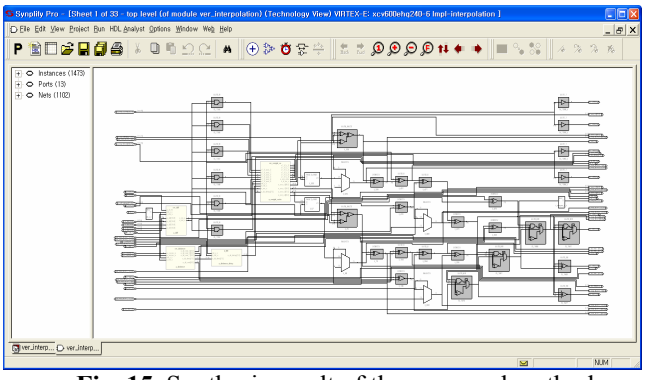


Fig. 15. Synthesis result of the proposed method

Table 4. The number of gates by the block each

Block	Gates
Vertical Scaler	15,226
Horizontal Scaler	14,991
Mode Detector	550
Timing Controller	624
Line Memory & Memory Controller	660.899

The proposed method was implemented in the hardware using an FPGA. Figure 16 shows the prototype board.

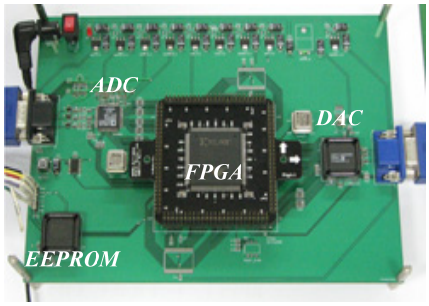


Fig. 16. The FPGA-based prototype board

6. Conclusion

In this paper, we proposed an image interpolator which had less computation complexity than conventional interpolation. In order to reduce the computation complexity, we used the linear function of the cubic convolution function and the difference in pixel value for selecting interpolation methods. Simulation results show that the proposed method provides less computation complexity than one of the conventional interpolation. The proposed method might be suitable to be applied to display systems which have the different resolution of image sources and digital display devices.

Reference

[1] R. Crane, A Simplified Approach to Image Processing, Prentice-Hall, 1997.

[2] R. C. Gonzalez and R. E. Woods, Digital Image Processing, Prentice-Hall, 1992.

[3] R. G. Keys, “Cubic Convolution Interpolation for Digital Image Processing,” IEEE Trans. ASSP, Vol. ASSP-29, pp. 1153-1160, 1981.

[4] Y. Bai and H. Zhuang, “On the Comparison of Bilinear, Cubic Spline, and Fuzzy Interpolation Techniques for Robotic Position Measurements,” IEEE Trans. Inst. Meas., Vol. 54, No. 6, pp. 2281-2288, 2005.

[5] S. S. Rifman, “Digital Rectiflcation of ERTS Multispectral Imagery,” Pmc. Symposium Signifiant Results Obtained from ERTS-1 (NASA SP-327), I, Sec. B, pp. 1131-1142, 1973.

[6] R. Bernstein, “Digital Image Processing of Earth Observation Sensor Data,” IBA4 J. Res. Dewel., MI. 20, pp. 40-57, 1976.
[7] C. H. Kim, S. M. Seong, J. A. Lee, L. S. Kim, “Winscale-: An Image-Scaling Algorithm Using an Area Pixel Model,” IEEE Trans. Circuit and System, Vol. 13, No. 6, pp. 549-553, 2003.
[8] H. C. Kim, B. H. Kwon, M. R. Choi, “ An Image Interpolation with Image Improvement for LCD Controller,” IEEE Trans. Cons. Elect, Vol. 47,No 2, pp. 263-271, 2001.
[9] Y. H. Jun, J. H. Yun, and M. R. Choi, "Modified Cubic Convolution Interpolation for Low Computational Complexity," IMID/IDMC 2006, pp. 1293-1296, 2006.

Young-Hyun Jun
He received the B.S. degree in electronic engineering from Kyungnam University in 2003. He is currently pursuing the M.S. degree at the Hanyang University. His research interests include SoC/ASIC design, image processor, and flat panel displays.

Jong-Ho Yun
He received the B.S. degree in control and instrumentation engineering from Hanyang University in 1999 and the M.S. degree in control and instrumentation engineering from Hanyang University in 2003. He is currently pursuing the Ph.D. degree at the Hanyang University. His research interests include SoC/ASIC design, image processors, and flat panel displays.

Jin-Sung Park
He received the B.S. degree in control and instrumentation engineering from Hanyang University in 1995 and the M.S. and Ph. D. degrees in control and instrumentation engineering from Hanyang University in 1997 and 2005, respectively. From 2000 to 2002, he was with the Mani Network Co. where he was a Software Development Team Manager. From 2003 to 2004, he was with the Nautilus Hyosung Inc. where he was a Solution Team Manager. In 2005, he joined the CEN Co.,Ltd., where he is now a CTO. His current research interests include SoC/ASIC design, Smart Card and RFID security.

Myung-Ryul Choi
He received the B.S. degree in electronic engineering from Hanyang University in 1983 and the M.S. and Ph. D. degrees in electrical engineering from Michigan State University in 1985 and 1991, respectively. From 1991 to 1992, he was with the Korea Institute of Industrial Technology where he was an assistant professor. In 1992, he joined the Department of Electrical Engineering and Computer Science at Hanyang University, where he is now a professor. His current research interests include SoC/ASIC design, 2D/3D FPD controller, and Smart Card/RFID applications.