A Low-Cost and High-Throughput FPGA Implementation of the Retinex Algorithm for Real-Time Video Enhancement

Jin Woo Park, *Student Member, IEEE*, Hyokeun Lee, *Student Member, IEEE*, Boyeal Kim, *Student Member, IEEE*, Dong-Goo Kang, Seung Oh Jin, Hyun Kim, *Member, IEEE*, and Hyuk-Jae Lee, *Member, IEEE*

***Abstract*— For video applications in a special environment such as medical imaging, space exploration, and underwater exploration, the video captured by an image sensor is often deteriorated because of low lighting conditions. Therefore, it is necessary to enhance the part of the image that is too dark to distinguish details while maintaining the remaining part with the same brightness. The retinex algorithm is widely used to restore naturalness of a video, especially exhibiting outstanding performance in the enhancement of a dark area. However, it demands large computational complexity because of its intricate structure, such as the Gaussian filter and exponentiation operations, and consequently, it is difficult to process in real time. This article presents a low-cost and high-throughput design of the retinex video enhancement algorithm. The hardware (HW) design is implemented using a field-programmable gate array (FPGA), and it supports a throughput of 60 frames/s for a 1920 × 1080 image with negligible latency. The proposed FPGA design minimizes HW resources while maintaining the quality and the performance by using a small line buffer instead of a frame buffer, by applying the concept of approximate computing for the complex Gaussian filter, and by designing a new and nontrivial exponentiation operation. The proposed design makes it possible to significantly reduce HW resources (up to 79.22% of total resources) compared to existing systems and is compatible with commercialized devices through the standard HDMI/DVI video ports.**

***Index Terms*— Approximate computing, field-programmable gate array (FPGA) implementation, low-cost implementation, real-time implementation, retinex algorithm, video enhancement.**

Manuscript received January 4, 2019; revised April 26, 2019, May 31, 2019, and July 8, 2019; accepted August 10, 2019. Date of publication September 11, 2019; date of current version December 27, 2019. This work was supported in part by the Korea Electrotechnology Research Institute (KERI) Primary Research Program through the National Research Council of Science and Technology (NST) funded by the Ministry of Science, ICT and Future Planning (MSIP) under Grant 17-12-N0103-10 and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) under Grant 2019R1F1A1057530. *(Corresponding author: Hyun Kim.)*

J. W. Park, H. Lee, B. Kim, and H.-J. Lee are with the Inter-University Semiconductor Research Center, Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, South Korea (e-mail: jwpark@capp.snu.ac.kr; hklee@capp.snu.ac.kr; bykim@capp.snu.ac.kr; hyuk_jae_lee@capp.snu.ac.kr).

D.-G. Kang and S. O. Jin are with the RSS Center, Korea Electrotechnology Research Institute, Ansan-si 15588, South Korea (e-mail: dgkang@keri.re.kr; sojin@keri.re.kr).

H. Kim is with the Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul 01811, South Korea, and also with the Research Center for Electrical and Information Technology, Seoul National University of Science and Technology, Seoul 01811, South Korea (e-mail: hyunkim@seoultech.ac.kr).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2019.2936260

I. Introduction

As DIGITAL images are widely used in many applications, image sensors with their processing techniques have been actively developed [1]–[3]. For applications such as space exploration [4], medical imaging [5], and underwater exploration [6], of which the capturing environment is often invisible directly for humans, the digital image captured by an image sensor is important to obtain the information about the environment. In these applications, it is often the case that the image quality is degraded because of a poor lighting condition. Therefore, quality enhancement is essential prior to extracting the information necessary for a target application. For a digital image obtained in a poor lighting condition, the difference between bright and dark areas is very large, which makes it difficult to distinguish details, especially in dark areas [7]. The increase of overall brightness makes dark areas look better, but it results in the saturation in bright areas thereby making the details in bright areas disappear. Therefore, these images require enhancement of details only in dark areas.

Medical imaging is a typical application in which the image quality deteriorates due to a poor lighting condition. When a camera sensor or a camera lens is inserted into a human body (for example, during endoscopy or laparoscopy), certain areas of an image may be very dark because of limited lighting. Especially when the inside structure of an organ is long and cylindrical, a low lighting condition makes the image very difficult to distinguish details. Therefore, an effective image preprocessing technique is essential to partially enhance the brightness of dark areas while maintaining the same brightness of the remaining areas.

Extensive efforts have been made to overcome the low illumination problem, and as a result, a number of algorithms have been proposed to brighten dark areas or enhance contrast such as gamma correction [8]–[11], histogram equalization [12]–[15], and tone mapping [16]. These algorithms often suffer from over-enhancement if the gray level of an image is concentrated at a specific intensity [17]. Moreover, the use of global information may result in intensity saturation such that enhancement results are inconsistent depending on image details. To overcome these problems, advanced algorithms with increased complexity, such as adaptive histogram equalization [18], [19], adaptive contrast enhancement [20], [21], and retinex algorithm [22]–[25], have been proposed. Adaptive

algorithms use local pixel data together with global information, which is very effective in enhancing dark areas based on different image details. However, these algorithms demand high computational cost, making them difficult to use for applications that require real-time operation. In particular, the retinex algorithm is considered highly valuable for medical imaging [17], [26], thanks to its excellent enhancement in dark areas without a degradation of image details. However, its computational cost is very high to process in real time.

One of the solutions to speed up highly complex algorithms is the implementation of a dedicated hardware (HW) accelerator using a field-programmable gate array (FPGA) [27]–[30]. Although these HW accelerators achieve real-time operation, they use large HW resources, thereby demanding an expensive FPGA device. They require large HW resources because of the following three reasons. First, the input image is stored in a memory. Li *et al.* [27], Marsi and Ramponi [29], and Tsutsui *et al.* [30] use external DRAMs on an FPGA to store image frames, whereas Ustukov *et al.* [28] use internal BRAMs instead. If an external DRAM is used, additional overhead is incurred for memory access and an additional DRAM controller is required, which increases the HW cost. If the image is stored in internal BRAMs, the image resolution is limited to the BRAM size and other HW modules are also limited due to the limited BRAM. Second, the previous HW implementation employs image enhancement based on multiscale retinex (MSR) which requires large HW resources to utilize the results from multiple Gaussian filters. Third, the previous implementation does not take advantage of the tradeoff relationship between performance and HW resources, and they lack efficiency in terms of HW design. Li [31] has tried to solve these problems by implementing a low-cost and high-speed HW with dynamic range compression. However, because the base algorithm used in [31] is simpler than the retinex algorithm and the size of the Gaussian filter utilized in [31] is very small, the techniques in [31] are not appropriate to be adopted in the retinex algorithm.

To address these problems of the previous implementations, this article presents a low-complexity and real-time HW design of the retinex algorithm proposed by Shin *et al.* [32] for efficient naturalness restoration. The main contributions of the proposed design are as follows. First, the proposed HW reduces excessive memory access and large memory latency by employing stream data control. Second, the concept of approximate computing [33] is applied to the proposed HW design by trading off HW resources with performance while maintaining subjective video quality. Particularly in the Gaussian filter module, which is the most expensive module, an optimal sigma value is determined and the filter size is minimized accordingly. Third, external memory access is further reduced by using only a line buffer to store the pixel values used by the Gaussian filter module. Fourth, the multiplications in convolution operations are approximated and then implemented only with shifters and adders [34]. Finally, a new method is proposed to efficiently implement the exponentiation operation that demands an important part of the total HW cost. Exponentiation is very challenging to implement in HW, thereby incurring large HW overhead which is significantly reduced by the proposed implementation. The proposed HW design reduces HW resources significantly (up to 79.22% of total resources compared to those of the previous works) while supporting the throughput of 60 frames/s for a 1920×1080 resolution (FHD) image.

The proposed HW design is implemented using an FPGA, and it is verified with real-time demonstration using popular medical devices, namely, a Karl Storz sensor for the input device and a Sony medical display for the output device. The standard HDMI/DVI ports are used as the input and output interfaces so that the proposed FPGA implementation can be connected to other medical devices as well.

The rest of this article is organized as follows. Section II introduces the theoretical basis of the proposed work, the retinex theory, and a few algorithms based on the retinex theory including efficient naturalness restoration. Section III describes the characteristics and novelties of the proposed HW design which is implemented in the register transfer level (RTL). Section IV discusses the results including the latency, video quality, and resource utilization. Section V concludes this article.

II. Retinex Theory and Algorithms

A. Concept of the Retinex Algorithm

The retinex theory proposed by Land and McCann [35] mathematically shows that both reflectance and illumination components enter the human eye when recognizing light; however, the brain perceives the color information mainly by the reflectance component. Therefore, even if the color and brightness of different scenes are equally received by the eye, the brain perceives the scene differently depending on the composition of the reflectance and illumination components. The image recognized by the human eye is expressed by the product of the illumination and reflectance components, and the reflectance component contains only the color information of the object regardless of the intensity of the light. Therefore, if the reflectance component can be extracted from the input image, the dark area of an unidentifiable image can be effectively restored through color restoration.

Based on the retinex theory, many researchers including Jobson *et al.* [36], Terzopoulos [37], and Kimmel *et al.* [38] propose algorithms for the color restoration of the human optic nerve to apply for computer images. By using a specific convolution filter, the illumination channel of the input image is obtained and then the illumination channel is separated from the original pixel to exclude the reflectance channel. The reflectance channel accurately represents the color information of each part in the input image. There are various methods to estimate the illumination channel such as the inverse square function proposed by Land [39], the exponential absolute value proposed by Moore *et al.* [40], and the Gaussian filter. Jobson *et al.* [41] suggest that using Gaussian filters is more efficient than using other filters because a Gaussian filter achieves good performance for wide dynamic ranges in various spatial regions.

B. Retinex Algorithm Using Gaussian Filters

The retinex algorithm using Gaussian filters, which shows effective enhancement in dark areas, can be expressed as follows. First, the input image $I(x, y)$ is expressed by the product of the illumination channel $L(x, y)$ and the reflectance channel $R(x, y)$

$$I(x, y) = L(x, y) \times R(x, y). \tag{1}$$

To obtain the illumination channel, the Gaussian filter kernel $F(x, y)$ is obtained using (2) and the input image and the kernel are convolved ($\otimes$) to obtain the illumination channel, as shown in (3)

$$F(x, y) = Ke^{-(x^2+y^2)/\sigma^2} \tag{2}$$

$$L(x, y) = F(x, y) \otimes I(x, y) \tag{3}$$

where K represents the normalization factor which makes the sum of all $F(x, y)$ become 1 and σ represents the standard deviation of the kernel which decides the shape of the filter.

To obtain the reflectance channel, the illumination channel is separated from the input image in a pixel-wise manner and this operation is expressed by the following logarithmic form:

$$R(x, y) = \log I(x, y) - \log(F(x, y) \otimes I(x, y)). \tag{4}$$

As described in this section, using a single-scale Gaussian filter to estimate the illumination channel and to calculate the reflectance channel is called the single-scale retinex (SSR) algorithm.

C. Multiscale Retinex Algorithm

The Gaussian filter can be used to efficiently estimate the illumination channel; however, using one Gaussian filter does not reflect the delicate nature of the human eye. Especially when an image has both high-frequency and low-frequency regions, the SSR cannot achieve good quality. To reduce this problem, Jobson *et al.* [36] propose the MSR algorithm which uses several Gaussian filters with different scales.

The MSR algorithm uses several Gaussian filters of various scales by adjusting the σ_n value and creates several kernels $F_n(x, y)$, as shown in the following equation:

$$F_n(x, y) = Ke^{-\frac{x^2+y^2}{\sigma_n^2}}. \tag{5}$$

For each kernel, the reflectance channels $R_n(x, y)$ are created by using (6) and the weighted sum is obtained by using the weight w_n to create the MSR reflectance channel, R_{MSR}

$$R_n(x, y) = \log I(x, y) - \log(F_n(x, y) \times I(x, y)) \tag{6}$$

$$R_{\mathrm{MSR}} = \sum_{n=1}^{N} w_n R_n. \tag{7}$$

The weight value w_n is determined such that the sum from w_1 to w_n becomes 1. Jobson *et al.* [36] proposes that using three different σ_n and determining each w_n to 1/3 generates the best result.

By using the MSR, the retinex algorithm can be applied to various frequency domains to effectively estimate the delicate illumination channels; however, the output reflectance channel is too extreme and unnatural to use. In addition, the MSR algorithm requires large computational complexity because of the utilization of several Gaussian filters.

D. Efficient Naturalness Restoration

To overcome these disadvantages of the MSR algorithm, Wang *et al.* [42] propose a naturalness preserved enhancement algorithm. Wang *et al.* [42] enhance the image using the retinex algorithm and gain naturalness by adapting the bright-pass filter to restrict the reflectance channel. Shin *et al.* [32] propose an efficient naturalness restoration algorithm, which has much less computational cost compared to [42]. By using the maximum value of an RGB channel called the intensity channel, the illumination channel can be obtained by performing single-scale Gaussian filtering and it is modified to obtain the reflectance channel by dividing pixel-wise from the intensity channel. Furthermore, the illumination channel is modified again and is recomposed with the reflectance channel to enhance the dark areas of the image by natural color restoration. It should be noted that the previous research by Gao *et al.* [43] shows performance comparison between different naturalness preserving retinex algorithms, and the results show that Shin *et al.*'s algorithm [32] has achieved lower computational cost even compared to the most recent works especially when the image resolution is large.

Efficient naturalness restoration can be described as follows. First, the $L(x, y)$ channel is calculated with the maximum value of the RGB channel from the input image $I^c(x, y)$ as shown in (8). Then, the $L(x, y)$ channel is filtered using Gaussian kernel $G(x, y)$ through convolution, which creates the $F(x, y)$ channel as shown in (9)

$$L(x, y) = \max(I^c(x, y)) \tag{8}$$

$$F(x, y) = L(x, y) \otimes G(x, y). \tag{9}$$

Then, the weighting map channel $w(x, y)$ is generated as follows:

$$D(x, y) = |L(x, y) - F(x, y)| \tag{10}$$

$$w(x, y) = D(x, y) \otimes G_w(x, y). \tag{11}$$

Next, the modified illumination channel $F_m(x, y)$ is generated by (12). Subsequently, the original RGB channels are each divided by the $F_m(x, y)$ channel to create the reflectance channel, as shown in (13)

$$F_m(x, y) = w(x, y) \times L(x, y) + (1 - w(x, y)) \times F(x, y) \tag{12}$$

$$R^c(x, y) = I^c(x, y)/F_m(x, y). \tag{13}$$

To gain naturalness, adaptive gamma correction is applied to the $F_m(x, y)$ channel and creates an $F_{\mathrm{enh}}(x, y)$ channel, as shown in (14). $\gamma(x, y)$ and α are derived by (15) and (16), respectively, where m is the average value of the $L(x, y)$ channel

$$F_{\mathrm{enh}}(x, y) = F_m(x, y)^{\gamma(x,y)} \tag{14}$$

$$\gamma(x, y) = (F_m(x, y) + \alpha)/(1 + \alpha) \tag{15}$$

$$\alpha = 1 + m. \tag{16}$$

The original efficient naturalness enhancement algorithm further modifies $F_{\mathrm{enh}}(x, y)$ via contrast enhancement using data distribution; however, this requires calculation and the updation of the histogram. This contrast enhancement method

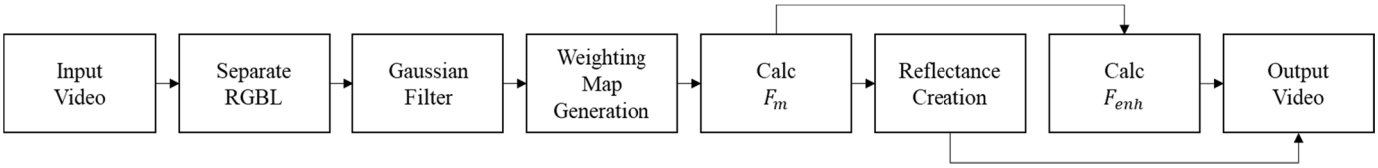


Fig. 1. Block diagram of the proposed architecture.

is suitable for a single image, but when adapted in a video stream, it creates a blinking effect where the dynamic range of the frame varies between frames. In addition, in our proposed system, calculating and updating the histogram causes frame unit latency. Therefore, we do not adapt the contrast enhancement method and instead use the $F_{\text{enh}}(x, y)$ channel to calculate the final result by using the following equation:

$$F^c_{\text{final}}(x, y) = F_{\text{enh}}(x, y) \times R^c(x, y). \tag{17}$$

The main advantage of the efficient naturalness restoration algorithm is excellent color restoration, improved naturalness in color representation, and low computational complexity compared to the MSR algorithm. In this article, the most efficient naturalness restoration algorithm proposed by Shin *et al.* [32] is implemented in HW targeting an FPGA device by applying novel optimization techniques for complexity reduction and fast execution while taking advantage of the properties of the target algorithm. A detailed description of the implementation is presented in Section III.

III. RTL Implementation and Optimization of an Image Enhancement Algorithm for an FPGA

A. Overview of the Proposed HW Design for Image Enhancement HW

This section presents the HW implementation of the efficient naturalness restoration algorithm proposed by Shin *et al.* [32] verified with an FPGA to operate in real time. Real-time FPGA implementation of the target algorithm is challenging because it involves complex modules, such as the Gaussian filter and exponentiation operation. In addition, the target algorithm includes some variables that must be calculated with all pixels in the entire frame, which is difficult to implement without using a frame memory. Fig. 1 describes the block diagram of the image enhancement HW design, which efficiently overcomes these challenges. The proposed HW design is implemented with seven main modules. Each module is implemented to process the image enhancement algorithm in real time by reducing the overall latency. In particular, the data are processed as a stream between each module from the input stage to use only the minimum line buffer without using the frame buffer. In addition, the variables that require the entire frame data are calculated from the previous frame to refrain from generating frame-level latency. Furthermore, from the input stage of the video, each HW module is implemented to achieve a high-throughput processing one pixel per cycle. The whole system is implemented in the Verilog HW description language and Xilinx IPs, and the stream data are processed using the AXI4 stream bus [44].

TABLE I
Fixed Point Precision for Each Module's Output Channel

Module	Integral Bit	Fractional Bit	Q-format
Gaussian filter	0	13	Q0.13
Weighting map	0	13	Q0.13
Calculate F_m	1	13	Q1.13
RGB creation	5	13	Q5.13
Calculate F_{enh}	0	8	Q0.8

The role of each module is described as follows. The separate RGBL module separates the intensity channel from the input image. The intensity channel is created by taking the maximum value of the RGB channel. The Gaussian filter module convolves the intensity channel with the predefined Gaussian kernel to create the F channel, which is used for illumination estimation. To calculate the convolution between the kernel and the large pixel window with a low cost, a delicate decision process that chooses the kernel size is used, and a small line buffer instead of the frame buffer is utilized in this module. The weighting map generation module calculates the weight values that are used in the calculate F_m module, which modifies the F channel to calculate the modified illumination channel (i.e., F_m channel). The RGB creation module divides the F_m channel from the RGB channels of the original image in a pixel-wise manner to create the reflectance channels R_R, G_R, and B_R, respectively. The calculate F_{enh} module performs adaptive gamma correction with operations including exponentiation by using F_m and the average of intensity channels. Finally, the F_{enh} channel and reflectance channels, R_R, G_R, and B_R are multiplied to create the output channel $I_{\text{enh}}(x, y)$. Calculating F_{enh} channel requires a very complex operation, namely, exponentiation. A novel method of implementing the exponentiation is introduced with mathematical developments and optimization of the HW resources.

A software (SW) algorithm using double-precision floating point is implemented in RTL with fixed-point precision to reduce HW resources. The input image is represented by 8 bits for each RGB channel, and each module is implemented in the fixed-point precision with 13 fractional bits to minimize the error. The fixed-point precision of each channel is described in a Q-format in Table I.

Fig. 2 describes the output results of each module depicted in the block diagram. Thanks to the effect of the retinex algorithm, the reflectance channel $R(x, y)$ dramatically boosts up the color detail of the invisible dark areas in the input image. However, the result of $R(x, y)$ is boosted too much

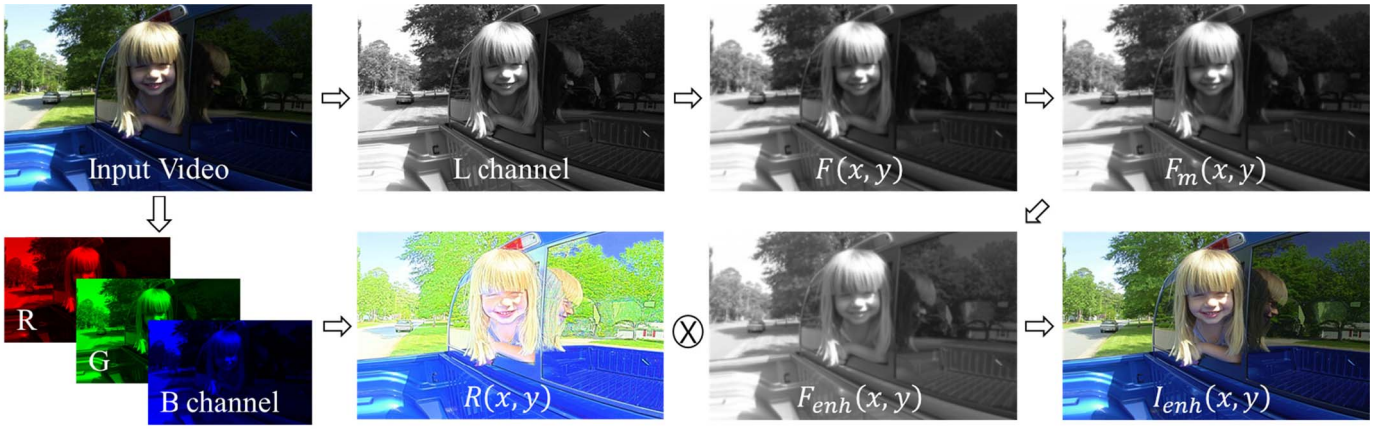


Fig. 2. Block diagram depicting output image of each module.

to lose color naturalness. Therefore, the result of $R(x, y)$ is multiplied with the $F_{\text{enh}}(x, y)$ channel to create the final output channel $I_{\text{enh}}(x, y)$.

Sections III-B–III-D describe the efficient input data processing method, a low-complexity implementation of the Gaussian filter module with an approximation technique, and efficient implementation of a nontrivial exponentiation operation.

B. Input Data Processing Without a Frame Buffer

For processing the input data, only a small line buffer is used instead of a large frame buffer. Therefore, the whole system does not use either large internal block memory, or additional external memory. The entire module transmits and receives image data by a simple data transfer method, AXI4-stream bus, and the latency is minimized by designing the modules to start sending outputs as soon as the required number of pixels are given as the input data.

For the implementation of the modules that require a certain amount of pixel data before the operation, the size of the line buffer is minimized. For example, the Gaussian filter module uses a pixel window of size $N \times N$, which means that this module requires a slightly larger line buffer than N to calculate convolution as soon as the last pixel of the window is received. In this article, a 29×29 window is used for the Gaussian filter module, and consequently, 32 lines are buffered considering the input and output latencies. The input pixels are received in stream as raster scan order, and some of the input pixel data as well as the filtered pixel data are used on the later module. Therefore, 32 lines of the input data are stored in the line buffer to be used in the Gaussian filter module as well as the later module that uses the filtered data and the input data.

If the internal block memory is used as the frame buffer, a single frame requires 49 Mb of memory for an FHD resolution image. If the frame buffer uses a large resource from the limited internal block memory, other modules cannot utilize the internal block memory sufficiently, and the frame buffer also consumes significant power. If the frame buffer uses the external memory, an extra module is required to control the data transfer from the external memory, which leads to additional HW resource utilization. Moreover, if the line buffer is used instead of a frame buffer, the internal block memory resource can be saved considerably because the required number of pixels to be stored is the number of required lines multiplied by the image width. In addition, it can prevent the use of additional HW resources because the external memory is not required.

C. Approximation in the Gaussian Filter Module

The Gaussian filter module in the target algorithm plays an important role in image enhancement. The σ value affects the performance of the whole system, and the target algorithm achieves better performance if σ has a larger value. However, a larger σ value requires a larger filter size, which leads to utilizing more HW resources, such as BRAMs and DSPs. Furthermore, it increases the line buffer size to store the number of pixels, which results in the increase of the latency between input and output.

1) Decision of the Parameters and the Optimum Filter Size: The Gaussian kernel is created using $\sigma = 10$ to enhance the dark area of the image clearly. Fig. 3 shows the results of the target algorithm according to the σ value of the Gaussian filter. As the σ value increases, the image contrast is enhanced. However, its performance is low when enhancing dark areas. The proposed design aims to improve the details of invisible dark areas on an image; thus, excessively high σ value is unnecessary. As shown in Fig. 3, the dark part reflected on the window improves the most when the σ value is 10 or 20, and consequently, σ is set to 10 because a larger σ requires more HW resources. As mentioned previously, the kernel coefficient value is set to 13-bit fixed point so that the filter size for all the Gaussian coefficients having nonzero values is 53×53. (Note that in the case of $\sigma = 20$, the filter size is 120×120.) However, the coefficient value, which is far from the center, is very small; thus, applying the concept of approximation computing by implementing with a smaller filter with similar shape does not affect the subjective quality. A smaller filter size results in fewer line buffers to use less internal block memory, thereby greatly reducing the HW resources. Therefore, the 29×29 filter size is used in consideration of

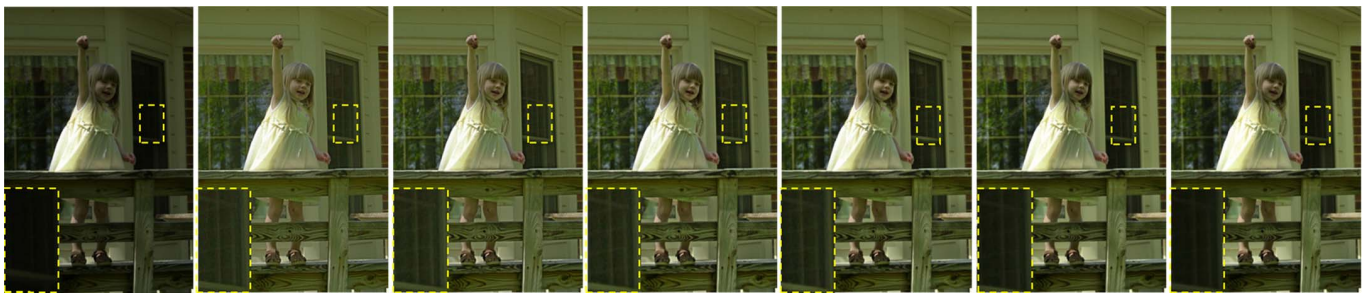


Fig. 3. Retinex result of different Gaussian filter shapes decided by the sigma value. (a) Original image. (b) $\sigma = 2.5$. (c) $\sigma = 5$. (d) $\sigma = 10$. (e) $\sigma = 20$. (f) $\sigma = 40$. (g) $\sigma = 80$.

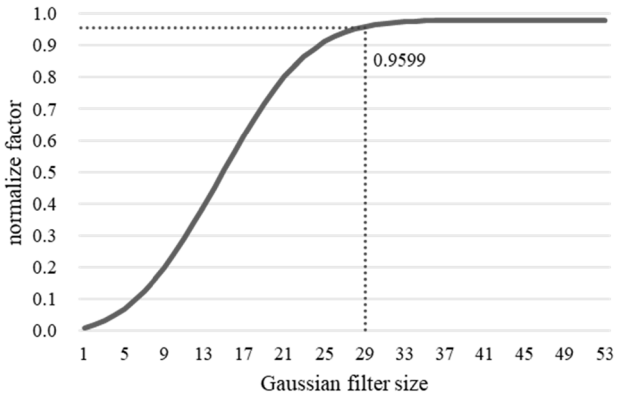


Fig. 4. Normalization factor value of different Gaussian filter sizes.

the tradeoff between performance and HW resource (see the B1 results in Section IV). The Gaussian coefficient values are normalized to make the sum equal to 1 by the normalization factor, which is the sum of the coefficients of the small filter. If the normalization factor has a larger value, the output of the Gaussian filter has a larger error. Fig. 4 shows the value of the normalization factor according to the filter size when $\sigma = 10$. When the filter size is 29×29, the normalization factor is 0.9599, which does not create a noticeable error, and the size is small enough to effectively reduce HW resources. If a smaller filter size is chosen, the gradient of the normalization factor increases exponentially, and the error exhibits the same trend. Fig. 5 depicts the shapes of the approximated kernel and the original kernel. It is observed that there is almost no difference between the two kernels. However, if the size of the filter is smaller than 29×29, the shape difference becomes noticeable and the error generated in the Gaussian filter module becomes significant.

2) Approximations on Convolution: The HW resources used in the Gaussian filter module are reduced by replacing the fixed-point multiplication operation with shifter and adder operations. Instead of multiplying the 13-bit fixed point of Gaussian coefficients, each coefficient is modified to use at most three shifters and two adders. The ith Gaussian coefficient f_i is defined as in (18), and the multiplication between f_i and pixel p is described as (18)

$$f_i = 2^{a_i} \pm 2^{b_i} \pm 2^{c_i} \tag{18}$$

$$p * f_i = p \ll a_i \pm p \ll b_i \pm p \ll c_i. \tag{19}$$

TABLE II
APPROXIMATED GAUSSIAN FILTER KERNEL OF SIGMA 10

Real Value	13-BIT FIXED POINT VALUE	Approximated Kernel Value	Approximated Kernel Notation
0.0082794	68	63	$2^6 - 2^0$
0.0108457	89	88	$2^6 + 2^4 + 2^3$
0.0139262	114	112	$2^7 - 2^4$
0.0175275	144	144	$2^7 + 2^4$
0.0216233	177	176	$2^7 + 2^5 + 2^4$
0.0261479	214	216	$2^8 - 2^5 - 2^3$
0.0309933	254	254	$2^8 - 2^1$
0.0360091	295	292	$2^8 + 2^5 + 2^2$
0.0410082	336	336	$2^8 + 2^6 + 2^4$
0.0457765	375	376	$2^8 + 2^7 - 2^3$
0.0500875	410	416	$2^9 - 2^6 - 2^5$
0.0537192	440	440	$2^9 - 2^6 - 2^3$
0.0564735	463	464	$2^9 - 2^6 + 2^4$
0.0581934	477	478	$2^9 - 2^5 - 2^1$
0.0587782	482	482	$2^9 - 2^5 + 2^1$

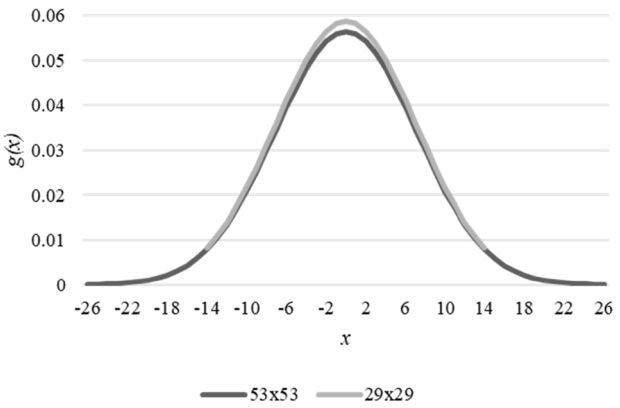


Fig. 5. Gaussian kernel shapes of full-size filter and approximated filter.

The 13-bit fixed-point coefficient of the modified kernel and the equation notation of f_i from number 1–15 are presented in Table II. The coefficient values of number 16–29 are symmetrical with the previous coefficients. The difference between these two values is negligible; thus, by using shifters and adders, Gaussian filter multiplication is implemented with reduced HW resources.

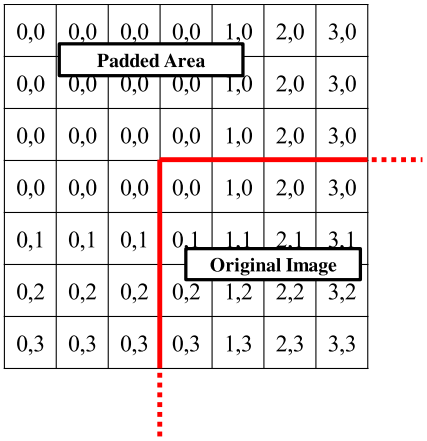


Fig. 6. Padding edge pixels to form a Gaussian filter window. Coordinates of pixels are marked on each pixel.

Finally, to implement a Gaussian filter of size 29×29, a line buffer of size 1920×32 is utilized. Instead of using a 2-D filter, where 29×29 operations between a pixel and a coefficient value are required for convolution, two 1-D filters are used vertically and horizontally to reduce the number of operations to 29×1 plus a small number of additional adders. For filtering the edge of the image, the pixel window contains an inexistent area of the image. To overcome this problem, a method of constructing a Gaussian window by padding the edge area is shown in Fig. 6. As shown in Fig. 6, the pixel value of this part is used by copying the edge part of the image when applying 1-D filtering to the pixel at the required position, so that additional memory and latency are not generated.

D. Implementation of Exponentiation Operation

The module for calculating F_{enh} computes the adaptive gamma correction on the F_m channel using the average value of the intensity channel to create the F_{enh} channel. Calculating the F_{enh} channel for each pixel is expressed in (14). To calculate $F_{\text{enh}}(x, y)$, an exponentiation operation is required. Both the base and the exponent are variables in this operation, and consequently, this operation is nontrivial and very complicated to be implemented in RTL. In this article, the exponentiation HW module with minimal HW resources is optimized considering the bit width required for the image enhancement algorithm.

The proposed exponentiation HW module is implemented by decomposing the operation with base 2 logarithm and exponential operation. By substituting the base and the exponent as x and y, respectively, the exponentiation equation to calculate $F_{\text{enh}}(x, y)$ in (14) is written with base 2 logarithm and exponential as follows:

$$\text{Base}^{\text{Exponent}} = 2^{\text{Exponent} \times \log_2 \text{Base}}. \tag{20}$$

The exponent part of (20) is composed by a multiplication of Exponent and $\log_2$ Base. To calculate this exponent, the multiplication and logarithm operations are required. The exponent part is expressed as the sum of the integral part (I_{exp}) and the fraction part (F_{exp}) as follows:

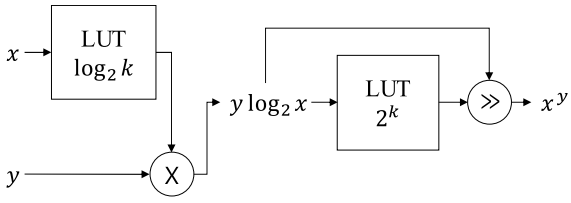


Fig. 7. Block diagram of the proposed exponentiation function.

$$\text{Exponent} \times \log_2 \text{Base} = I_{\text{exp}} + F_{\text{exp}}. \tag{21}$$

By expressing the exponent as the sum of these two parts, exponentiation in (20) is expressed as follows:

$$2^{\text{Exponent} \times \log_2 \text{Base}} = 2^{I_{\text{exp}} + F_{\text{exp}}} = 2^{I_{\text{exp}}} \times 2^{F_{\text{exp}}}. \tag{22}$$

The base 2 exponential of the integral part is replaced with shift operation; thus, the entire operation is expressed as follows:

$$2^{I_{\text{exp}}} \times 2^{F_{\text{exp}}} = 2^{F_{\text{exp}}} \gg |I_{\text{exp}}|. \tag{23}$$

It should be noted that the input variable Base of the exponentiation used in this article always has a range smaller than 1; thus, the absolute value of I_{exp} can be used for the right-shift operation, as shown in (22). By decomposing the exponentiation operation, it is obtained by multiplying, adding, shifting, base 2 logarithm and exponential operations. Multiplying, adding, and shifting can be performed in less than one cycle in RTL; however, the base 2 logarithm and exponential operations still require complex implementation. Two LUTs are used to perform the logarithm and exponential operations because both the range of the input value and the output bit width is fixed in the efficient naturalness restoration algorithm. Decomposed exponentiation operation using LUTs, a multiplier, and a shifter are depicted as an RTL block diagram in Fig. 7.

The sizes of the two LUTs decide the HW resource utilized in this module. Each LUT has its own input and output bit width, and the size of each LUT is decided by the bit width. The size of the LUT is expressed as follows:

$$\text{Size(LUT)} = 2^{n_{\text{bits,in}}} \times n_{\text{bits,out}}. \tag{24}$$

As shown in (24), the output bit width $n_{\text{bits,out}}$ affects the LUT size linearly while the input bit width $n_{\text{bits,in}}$ affects it exponentially. Therefore, the appropriate selection of the input bit width for each LUT is critical. In this article, bit width is chosen by analyzing the error caused by selecting the fixed-point bit width.

As we have implemented the operation to function with a specific purpose of image enhancement, the range of the possible input and output values is set. The range of the base is $0 \leq F_m(x, y) \leq 1$, where 0 indicates that the input image is entirely black, and 1 indicates that it is entirely white. The range of the exponent is $(1/2) \leq \gamma(x, y) \leq 1$, where 1/2 indicates that the input image is entirely black and 1 indicates that it is entirely white. Based on these ranges,

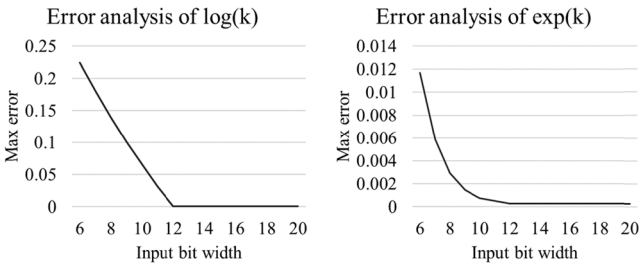


Fig. 8. Error analysis of the two LUTs according to the input bit width.

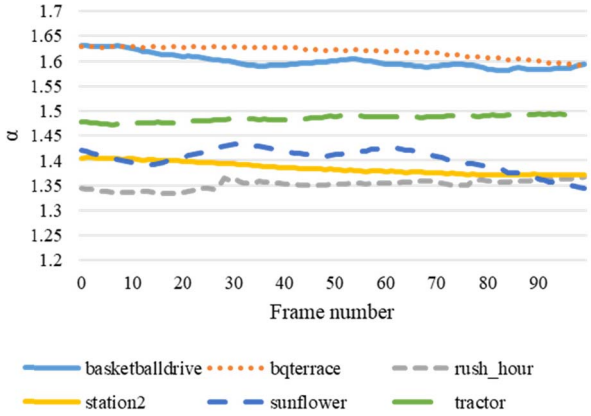


Fig. 9. Value of α on consecutive frames on a 60 frames/s video.

experiments have been conducted to analyze the error for all possible calculation of $\text{Base}^{\text{Exponent}}$ and the results are depicted on Fig. 8.

The max error is calculated from the difference between the implemented exponentiation function and the MATLAB double-precision power function. As shown in the graph in Fig. 8, both $\text{LUT}_{\log 2}$ and $\text{LUT}_{\exp 2}$ have the minimum error when the input bit width is 12 bits. It is noteworthy that the max error value remains constant even if the bit width is larger. The max error is $1/2^{12}$, which is the unit difference of the 12-bit fixed point used as the output of the overall exponentiation. This implies that a function with max error $1/2^{12}$ will produce either the most or the second most accurate calculation result when rounded to the nearest 12-bit precision. Therefore, we have created LUTs with input bit widths of 12 bits. The size of each LUTs is 49 kb.

This HW module of exponentiation operation is implemented efficiently with minimal HW resource while hardly affecting the performance. Therefore, the HW module for image enhancement, such as the Gaussian filter, can afford to utilize sufficient resources to utilize BRAMs and DSPs. In addition, the decomposition of the complex operation to a few simple operations saves the number of cycles required to perform the operation, and consequently, the proposed design with a low latency is suitable for real-time image processing.

E. HDMI/DVI Compatibility With Negligible Latency

Many commercial video devices transmit and receive video data through the HDMI or DVI port for input and output; thus, the proposed FPGA system is designed to support the HDMI/DVI interface. For the input, the Digilent FMC-HDMI expansion card is used, and for the output, the built-in HDMI output port of ZC706 evaluation board is used. It is worth mentioning that DVI port can be used with the HDMI port with a simple gender. The video data received from the HDMI port are transferred to the image enhancement module using the AXI4-stream bus without holding the data in any form of a frame buffer and are transmitted to the output HDMI port using the same bus protocol. All the data are transmitted at the rate of one pixel per cycle, which is the same as the video stream data transfer protocol.

For this compatibility to be a great advantage, it is important to eliminate any latency in the communication between the devices and the image enhancement HW module. As described in Section III-A, a use of a line buffer instead of a frame buffer makes it possible to implement the proposed image enhancement HW module with negligible latency, and consequently, it interacts smoothly with commercial devices. It should be noted that the proposed system creates a latency of only a half the line buffer size, which results in a very small latency.

Furthermore, an additional implementation technique to achieve negligible latency is applied to the F_{enh} channel. The F_{enh} channel requires a variable α, which is calculated using the average of the channel values of the entire frame. However, the derivation of α using the current frame creates at least a single frame delay. Therefore, the average of the intensity channel is calculated from the previous frame to prevent a delay of an entire frame. Experimental results show that the derivation of α using the previous frame does not create a noticeable error because the target video operates at 60 frames/s and therefore two consecutive frames have very little difference in their average of intensity. Fig. 9 shows the value of α for first 100 frames of several video sequences. As the frame changes, the value of α changes with a relatively large difference; however, the average difference between two consecutive frames is only 0.001272, which is 0.08% of the average α. The largest difference between two consecutive frames is 0.01687, and this is only 1.14% of α. This shows that in most cases, sudden change of α is unexpected, and it is reasonable to use the α calculated from the previous frame.

However, despite the rarity of the case, the system can suffer from the sudden change of the frame which leads to inappropriate usage of α. Even when the α is misused, the retinex algorithm functions normally and restores color information successfully, but the overall intensity is affected. However, when the scene is dramatically changed, it is likely to be in the process of moving the region of interest for the observer, and the importance of the observing this scene would be very low. Because the importance of observing the dramatically changing scene is very low, and this case is rare according to Fig. 9, the α calculated from the previous frame can be used for low latency design in the proposed HW implementation without performance degradation.

In conclusion, the proposed HW module operates with negligible latency and supports the HDMI/DVI port; thus, it can be used with various commercial devices in which image enhancement is required.

(a) (b) (c)

Fig. 10. Images with strong sunlight. (a) Original image. (b) Full-size filter. (c) Approximated filter.

TABLE III
LATENCY COMPARISON BETWEEN FRAME BUFFER AND LINE BUFFER 1920 × 1080 60 FRAMES/S VIDEO AT 148.5 MHz, 1 PIXEL PER CYCLE

Architecture	Cycles to Process	Latency (ms)
Frame buffer*	2073600	13.96
Line buffer – 53×53 filter*	51840	0.349
Line buffer – 29×29 filter*	28800	0.194
Proposed system**	35732	0.241

*Values are based on computations.
**Value is based on FPGA measurements.

IV. RESULTS AND DISCUSSION

A. Real-Time System With Negligible Latency

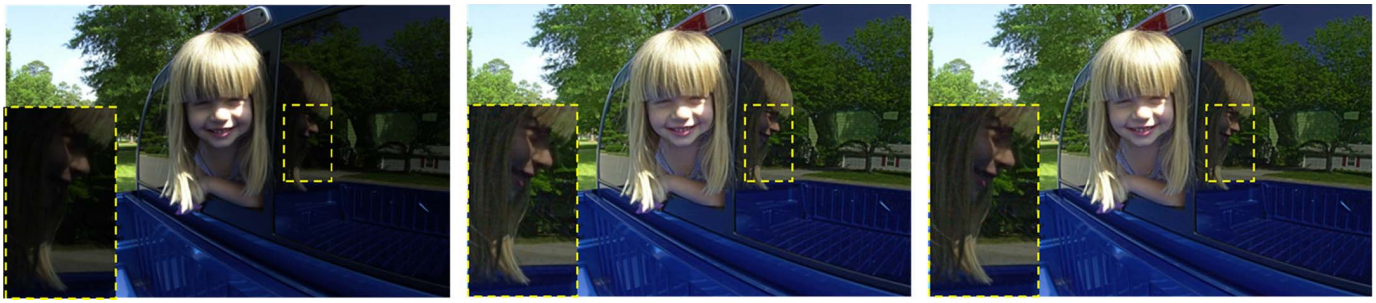
The proposed HW module processes the input of FHD resolution in real-time (i.e., 60 frames/s) and creates the output sequence with the same resolution and frame rate. The implemented module operates with the AXI4-stream bus, which transmits the video stream at the rate of one pixel per cycle; thus, a minimum clock frequency of $1920 \times 1080 \times 60 \approx 125$ MHz is required to satisfy FHD 60 frames/s throughput. In general, the video processing module supported by Xilinx uses 148.5-MHz clock frequency to process an FHD 60 frames/s video [45]; thus, the proposed system is also designed to operate with 148.5-MHz clock frequency. The increased clock frequency improves the throughput, but it also causes large power consumption. Therefore, a proper clock frequency is set to obtain the desired throughput while avoiding unnecessary power consumption.

As mentioned above, it is important to minimize the latency between the input and output videos because of the characteristics of the image enhancement system. Therefore, the latency is minimized through the proposed design method that does not use the frame buffer. Table III shows the comparison of the number of required cycles and latency between input and output videos according to the data processing method. When using a frame buffer, a delay of one frame or more is inevitable, resulting in a latency of at least 13.96 ms based on the FHD 60 frames/s video at 148.5 MHz. If double or triple frame buffers are used, the latency is even increased. On the other hand, the use of a line buffer minimizes the latency generated while processing the pixel data.

Among various modules used in this design, the Gaussian filter has the most complicated structure and generates the most latency. The Gaussian filter module with a 53×53 filter generates a latency of at least 0.349 ms even if the line buffer is used. This means that it is possible to create a filter window and process the Gaussian filter with a very low latency of 2.5% compared to that produced using a frame buffer. The idea of approximated computing is additionally applied to the design of the Gaussian filter to construct a 29×29 window, and consequently, the latency of the module is further decreased to 0.194 ms (57.3% reduction as compared to that using a 53×53 filter). The overall latency including the Gaussian filter module and all the other modules is measured as 0.241 ms, which is only 1.7% of that produced using a frame buffer. Therefore, it is possible to eliminate the latency between the input and output in commercial display devices satisfying FHD 60 frames/s even at 148.5 MHz. The value of 0.241 ms is only 2.67% of the smallest input lag of commercial display devices (9 ms) [46]; thus, the latency is unnoticeable even when the input and the output videos are displayed side by side.

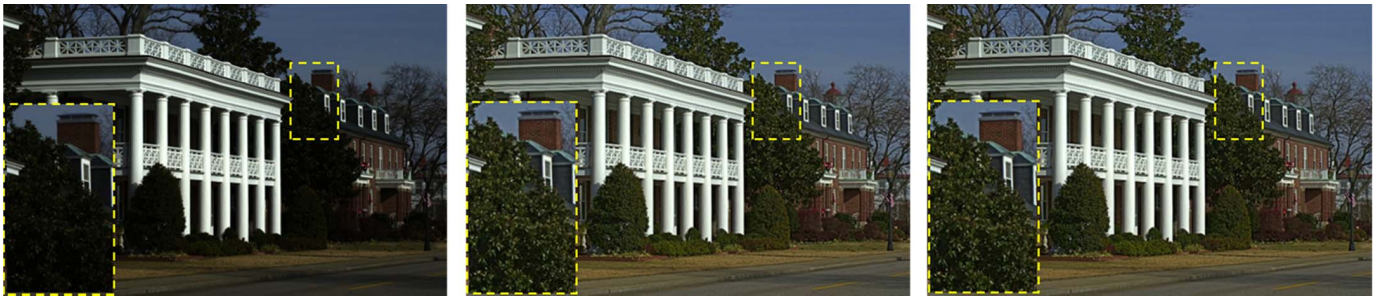
B. Evaluation and Comparison of Image Quality

To show that the approximated computing does not degrade the subjective quality, Figs. 10–12 depict the results of three cases (i.e., original image, enhanced image with a full-size filter, and enhanced image with an approximated filter) with three test images provided at NASA retinex image processing webpage [47], and Fig. 13 depicts the results of three cases with the images inside the artificially generated internal human abdominal captured by a laparoscopy camera. As shown in these figures, a clear difference is observed between the original image [i.e., (a)] and the full Gaussian [i.e., (b)]/approximated Gaussian [i.e., (c)] images because the retinex algorithm improves the dark area of the image. Moreover, difference is unnoticeable between the results of the full-size Gaussian filter and the results of the approximated 29×29 Gaussian filter. Even in the enlarged part of each image, the results of both the full and approximated filters show the same effect on improving the dark part and the details are maintained in both images. In Fig. 10, the edges of the tire and the details of the boy's head are improved and, in Fig. 11, the restoration of the details of the girl's face and the invisible leaves in the background are retained in the approximated filter. In Fig. 12, the detail of the leaves and



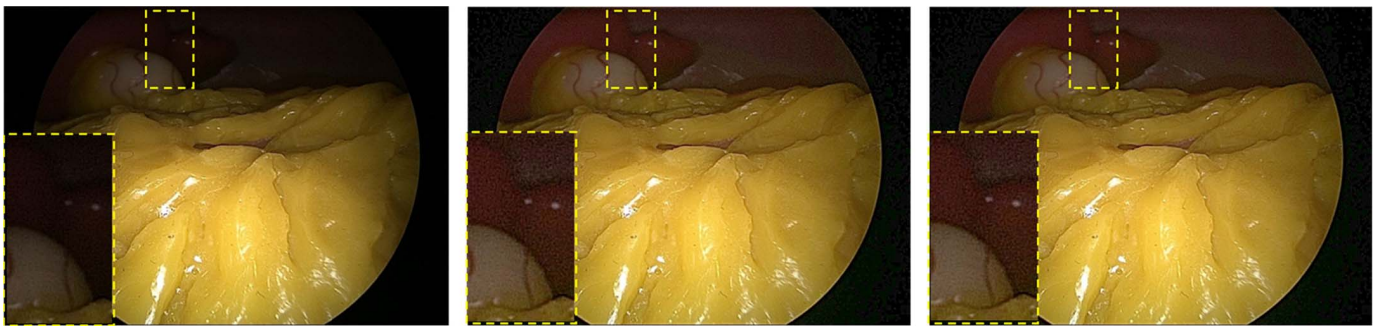
(a) (b) (c)

Fig. 11. Images with dark reflection area on a tinted window. (a) Original image. (b) Full-size filter. (c) Approximated filter.



(a) (b) (c)

Fig. 12. Images with loss created by shadows. (a) Original image. (b) Full-size filter. (c) Approximated filter.



(a) (b) (c)

Fig. 13. Images of a phantom taken with a laparoscopic camera. (a) Original image. (b) Full-size filter. (c) Approximated filter.

TABLE IV
AVERAGE OF ABSOLUTE DIFFERENCE BETWEEN FULL-SIZE FILTER AND APPROXIMATED GAUSSIAN FILTER

Figure	Boy	Building	Girl	Phantom	Average
Average of Absolute Difference	0.4143	0.4082	0.3622	0.1366	0.3303

the color restoration of the building wall is also maintained. In Fig. 13, the effect of improving the details of the artificial organ is not different between the two filters. This result shows that the effect of improving the dark part is reliably maintained even if approximation computing is applied to the Gaussian filter.

Table IV presents the average absolute difference per pixel between the results of the target algorithm using the full-size Gaussian filter and the result of approximation computing for the same four images. The results show that there is less than 0.42-pixel difference in all images and the average difference is approximately 0.33 per pixel. The average difference of 0.33 per pixel is 0.1% error rate, which is smaller than the unit difference of 8-bit precision pixel. This result shows that the approximate computing with a small-sized Gaussian filter does not make a significant difference in performance.

C. Evaluation of FPGA Implementation

The proposed design for image enhancement is implemented using the Xilinx ZC706 evaluation board. All image processing modules are implemented using only FPGA resources without access to an external memory. The effect of resource saving by applying the approximate computing to the Gaussian filter (i.e., using a reduced-size filter and replacing the multiplier with shifters and adders) is shown

TABLE V
RESOURCE COMPARISON OF APPROXIMATED GAUSSIAN FILTER

	53×53 Filter	29×29 Filter	Decreased %
Slice LUTs	36498	12998	35.61
Slice registers	10416	6085	58.42
Memory (RAMB36)	140	60	42.86
DSP (DSP48E)	37	0	-

in Table V. When approximation computing is applied, only 35.61% of the slice LUT, 58.42% of the slice register, and 42.86% of the memory are used compared to those used with a full filter design, and the DSPs are not used at all. Nevertheless, the subjective quality degradation is negligible as shown in Figs. 10–13, and summarized in Table IV.

Table VI represents the FPGA implementation resource utilization. The Gaussian filter module uses the most resources in the entire system. Therefore, the approximate computing of the Gaussian filter module effectively reduces the overall HW resources. In addition, it is possible for the proposed HW module to be used as a preprocessing system in other complex image processing modules to be implemented in a single FPGA board because the utilization ratio of the proposed HW module in the entire FPGA board is very low because of the low-cost implementation.

D. Comparison of FPGA Implementation

Table VII shows the comparison with the previous FPGA implementations of the retinex algorithm using Gaussian filters, developed by Li *et al.* [27], Ustukov *et al.* [28] and Marsi and Ramponi [29]. The target FPGAs and the target resolution of the previous implementations differ from those of the proposed implementation; thus, a reasonable metric is required to compare the HW with regards to different specifications.

To generate a reasonable comparison metric, each resource of different types, namely, LUT, register, block memory, and external memory@comm is normalized into memory bits according to the user guide of each FPGA fabric [48], [49]. In [48], the work is implemented on Xilinx Virtex-4, and each LUT and register used can be substituted with 16-bit memory. The work in [49] and the proposed system are implemented on Virtex-7 and Zynq7000, respectively, and these seven-series FPGA's LUT and registers can be used as 32-bit memory. In Marsi's [29] work, the system is implemented on Cyclone III. According to an article analyzing the difference between the two FPGA fabrics [50], one unit of the logic element used in Cyclone III is 1.3 times larger than one unit of LUT used in Virtex-4, so the resource utilization is converted into the estimated amount of LUTs of Virtex-4. This method of normalizing HW resources between different FPGA fabric usages for comparison has been adopted by Choi [51].

The eighth row of Table VII shows the normalized HW resources in megabits. As shown in Table VII, even without considering the throughput of the entire system, the proposed system utilized the least amount of resources because all three systems from the previous works utilized the frame buffer while the proposed system used the line buffer instead. The proposed system requires only 89.37% of the resources for [27], 20.78% of that for [28], and 41.51% of that for [29], while also exhibiting an improved throughput.

The throughput of the system is considered and the normalized HW resources per 1000 pixels (i.e., resource per throughput) are calculated in the ninth row of Table VII. The throughput can be calculated by considering the resolution and the frame rate. While considering the throughput, the proposed system utilizes only 5.96%, 3.08%, and 6.92% of the resources required for [27]–[29], respectively.

To demonstrate that the proposed system achieved low-cost, a comparison with HW implementations of local tone mapping algorithms proposed in [52] and [53] is shown in Table VIII. Because the complexity and the performance of the algorithms differ, a direct comparison of the resource usage may not be simple. However, it is comparable in some fashion, as these different algorithms function to enhance low dynamic range images to high dynamic range images. It should be noted that the algorithm used for tone mapping is much less complex than the retinex algorithm used in the proposed system [54].

Two studies using tone mapping and HDR method use a smaller size Gaussian filter; a 4×4 filter is used by Licciardo *et al.* [52] and a 5×5 filter is used by Ambalathankandy *et al.* [53]. Because the filter size plays a crucial role in determining the resource size, the comparison of the resource usage is based on the estimation of the HW resource, assuming the use of a 5×5 Gaussian filter instead of the 29×29 for the proposed system. The resources used for the proposed system are estimated by substituting the utilization of Gaussian filter of Table VI with the resources of the weighting map generation module, because a 5×5 sized Gaussian filter is used in this module.

The comparison between these works is also based on the normalization methodology stated earlier. The resource unit of Virtex-6 is similar to that of the seven series, and each LUT and register can be considered as 32-bit memory [55]. The resource usage of [53] is converted using the same method used to convert [29].

As shown in Table VIII, the proposed system utilized 94.51% of the resources used in [52] and 120% of the resources used in [53]. The resource usage is similar, or slightly larger, compared to other implementations. However, because the complexity of retinex is at least 168% greater than that of the local tone mapping algorithms as found in [54], we can assume that the proposed system successfully implemented the more complex algorithm with relatively low-cost HW resources.

E. HDMI/DVI Compatibility With Consumer Devices

The proposed FPGA implementation is tested by connecting it to the computer HDMI output, DVI output, and general display devices at FHD 60 frames/s. In addition, the proposed HW module is tested by connecting it to Karl Storz's Image 1 Hub image processor and Sony LMD-2760MD medical display. Karl Storz is a leading manufacturer of endoscopic and

TABLE VI
RESOURCE UTILIZATION OF SELECTED MODULES OF THE PROPOSED SYSTEM

	Slice LUT		Slice Register		Block Memory (RAMB36)		DSP	
	Utilization	%	Utilization	%	Utilization	%	Utilization	%
Gaussian filter	12853	5.88	6254	1.43	60	11.01	0	0
Weighting map generation	3565	1.63	5094	1.17	24	4.40	6	0.67
Calculate F_m	285	0.13	1079	0.25	0	0	2	0.22
RGB creation	1893	0.87	4263	0.98	0	0	0	0
Calculate F_{enh}	1684	0.77	1420	0.32	0	0	5	0.55
Full system	24176	11.06	22787	5.21	93	17.06	13	1.44

TABLE VII
COMPARISON RESULTS OF HARDWARE PERFORMANCE

	Li [27]	Ustukov [28]	Marsi [29]	Proposed
FPGA	Xilinx Virtex-4	Xilinx Virtex-7	Cyclone III	Xilinx Zynq7000
Resolution	720×576 Gray	640×480 RGB	720×576 RGB	1920×1080 RGB
Frame rate	60 fps	60fps	50 fps	60 fps
LUT	18186	20043	21608*	24176
Register	12724	420331	-	22787
SRAM	1616 Kb	9252 Kb	1179 Kb	3348 Kb
External memory	3317 Kb	0	9953Kb**	0
Normalized HW resource (Mbits)	5.43	23.34	11.69	4.85
Normalized HW / Throughput (Kbits / 1000 pixels)	39.26	75.99	33.81	2.34

* converted to LUT based on Virtex-4, where Cyclone III utilizes Logic Element instead of LUT and Register.

** resource usage is estimated because the paper stated the use of frame memory, but the utilization information is omitted.

TABLE VIII
COMPARISON RESULTS OF HARDWARE PERFORMANCE OF SIMILAR ALGORITHM IMPLEMENTATION

	Licciardo [52]	Ambalathankandy [53]	Proposed 5×5
Algorithm	Inverse tone mapping	Tone mapping with halo-reducing	Retinex
FPGA	Xilinx Virtex-6	Cyclone III	Xilinx Zynq7000
Resolution	1920×1080 RGB	1024×768 RGB	1920×1080 RGB
Frame rate	60 fps	126 fps	60 fps
LUT	51300	122186*	14888
Register	9200	-	21627
SRAM	1460 Kb	87 Kb	2052 Kb
Normalized HW resource (Mbits)	3.40	2.04	3.22
Normalized HW / Throughput (Kbits / 1000 pixels)	1.64	1.30	1.55

* converted to LUT based on Virtex-4, where Cyclone III utilizes Logic Element instead of LUT and Register.

laparoscopic devices and is used by many medical researchers and hospitals [56], [57]. The Sony medical display is a widely used medical device because of its high-quality display capability. Therefore, verification with these devices validates the compatibility and commercialization possibility of the proposed HW module.[1]

V. CONCLUSION

In this article, the algorithm used for improving the dark part of an image is implemented using an FPGA with a low-cost design to operate in real-time while maintaining the image quality. For a low-cost HW design, the efficient naturalness restoration algorithm is selected instead of MSR; the frame buffer is removed and complex modules, such as the Gaussian filter or exponentiation, are approximated and optimized considering the tradeoff between performance and HW resources. As a result, the proposed HW module implemented using an FPGA operates at 60 frames/s in real time with the FHD resolution even at a low operating clock frequency of 148.5 MHz, and the latency between the input and the output is unnoticeable. The proposed system saves up to 79.22% of HW resources compared to the existing retinex HW design without a noticeable subjective quality degradation. Furthermore, it can be used in various systems because of its high compatibility with commercialized video devices thanks to the support of the HDMI/DVI port. The

[1]The real-time demo of the proposed retinex HW module connected to consumer devices can be found at http://capp.snu.ac.kr/?p=research#Demos.

proposed design is expected to be widely used in real-time video processing where the improvement in dark areas is required.

REFERENCES

[1] W. Niblack, *An Introduction to Digital Image Processing*. London, U.K.: Prentice-Hall, 1986.

[2] D. Yang, A. El Gamal, B. Fowler, and H. Tian, "A 640×512 CMOS image sensor with ultrawide dynamic range floating-point pixel-level ADC," *IEEE J. Solid-State Circuits*, vol. 34, no. 12, pp. 1821–1834, Dec. 1999.

[3] T. Fukushima, Y. Kobayashi, K. Hirasawa, T. Bandoh, M. Ejiri, and H. Kuwahara, "An image signal processor," in *IEEE Int. Solid-State Conf. Circuits (ISSCC) Dig. Tech. Papers*, Feb. 1983, pp. 258–259.

[4] R. Molina, A. K. Katsaggelos, and J. Mateos, "Bayesian and regularization methods for hyperparameter estimation in image restoration," *IEEE Trans. Image Process.*, vol. 8, no. 2, pp. 231–246, Feb. 1999.

[5] P. K. Saha, R. Strand, and G. Borgefors, "Digital topology and geometry in medical imaging: A survey," *IEEE Trans. Med. Imag.*, vol. 34, no. 9, pp. 1940–1964, Sep. 2015.

[6] Y.-T. Peng and P. C. Cosman, "Underwater image restoration based on image blurriness and light absorption," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 1579–1594, Apr. 2017.

[7] M. Kim, D. Park, D. K. Han, and H. Ko, "A novel approach for denoising and enhancement of extremely low-light video," *IEEE Trans. Consum. Electron.*, vol. 61, no. 1, pp. 72–80, 2015.

[8] Y.-C. Chang and J. F. Reid, "RGB calibration for color image analysis in machine vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 5, no. 10, pp. 1414–1422, Oct. 1996.

[9] N. Moroney, "Local color correction using non-linear masking," in *Proc. 8th Color Imag. Conf.*, Nov. 2000, pp. 108–111.

[10] X. Guan, S. Jian, P. Hongda, Z. Zhiguo, and G. Haibin, "An image enhancement method based on gamma correction," in *Proc. IEEE Int. Symp. Comput. Intell. Design*, Dec. 2009, pp. 60–63.

[11] H. Farid, "Blind inverse gamma correction," *IEEE Trans. Image Process.*, vol. 10, no. 10, pp. 1428–1433, Oct. 2001.

[12] Y.-T. Kim, "Contrast enhancement using brightness preserving bi-histogram equalization," *IEEE Trans. Consum. Electron.*, vol. 43, no. 1, pp. 1–8, Feb. 1997.

[13] Y. Wang, Q. Chen, and B. Zhang, "Image enhancement based on equal area dualistic sub-image histogram equalization method," *IEEE Trans. Consum. Electron.*, vol. 45, no. 1, pp. 68–75, Feb. 1999.

[14] T. Arici, S. Dikbas, and Y. Altunbasak, "A histogram modification framework and its application for image contrast enhancement," *IEEE Trans. Image Process.*, vol. 18, no. 9, pp. 1921–1935, Sep. 2009.

[15] S. Battio, A. Castorina, and M. Mancuso, "High dynamic range imaging for digital still camera: An overview," *J. Electron. Imag.*, vol. 12, no. 3, pp. 459–469, Jul. 2003.

[16] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, "Photographic tone reproduction for digital images," in *Proc. SIGGRAPH Annu. Conf. Comput. Graph.*, Jul. 2002, pp. 267–276.

[17] S. Setty, N. K. Srinath, and M. C. Hanumantharaju, "Development of multiscale retinex algorithm for medical image enhancement based on multi-rate sampling," in *Proc. IEEE Conf. Signal Process., Image Process. Pattern. Recognit.*, Feb. 2013, pp. 145–150.

[18] J.-Y. Kim, L.-S. Kim, and S.-H. Hwang, "An advanced contrast enhancement using partially overlapped sub-block histogram equalization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 4, pp. 475–484, Apr. 2001.

[19] T. K. Kim, J. K. Paik, and B. S. Kang, "Contrast enhancement system using spatially adaptive histogram equalization with temporal filtering," *IEEE Trans. Consum. Electron.*, vol. 44, no. 1, pp. 82–87, Feb. 1998.

[20] S.-C. Huang, F.-C. Cheng, and Y.-S. Chiu, "Efficient contrast enhancement using adaptive gamma correction with weighting distribution," *IEEE Trans. Image Process.*, vol. 22, no. 3, pp. 1032–1041, Mar. 2013.

[21] D. Zhang, W. J. Park, S. J. Lee, K. A. Choi, and S. J. Ko, "Histogram partition based gamma correction for image contrast enhancement," in *Proc. IEEE 16th Int. Symp. Consum. Electron. (ISCE)*, Jun. 2012, pp. 1–4.

[22] Z.-U. Rahman, D. J. Jobson, and G. A. Woodell, "Retinex processing for automatic image enhancement," *J. Electron. Imag.*, vol. 13, no. 1, pp. 100–110, 2004.

[23] C.-T. Shen and W.-L. Hwang, "Color image enhancement using Retinex with robust envelope," in *Proc. IEEE Int. Conf. Image Process.*, Nov. 2009, pp. 3141–3144.

[24] B. Li, S. Wang, and Y. Geng, "Image enhancement based on Retinex and lightness decomposition," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 3417–3420.

[25] M. K. Ng and W. Wang, "A total variation model for Retinex," *SIAM J. Imag. Sci.*, vol. 4, no. 1, pp. 345–365, 2011.

[26] W. Ma, J.-M. Morel, S. Osher, and A. Chien, "An L1-based variational model for Retinex theory and its application to medical images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 153–160.

[27] Y. Li, H. Zhang, Y. You, and M. Sun, "A multi-scale retinex implementation on FPGA for an outdoor application," in *Proc. IEEE Int. Congr. Image Signal Process.*, Oct. 2011, pp. 1788–1792.

[28] D. I. Ustukov, Y. R. Muratov, and V. N. Lantsov, "Modification of Retinex algorithm and its stream implementation on FPGA," in *Proc. IEEE Medit. Conf. Embedded Comput.*, Jun. 2017, pp. 1–4.

[29] S. Marsi and G. Ramponi, "A flexible FPGA implementation for illuminance-reflectance video enhancement," *J. Real-Time Image Process.*, vol. 8, no. 1, pp. 81–93, Mar. 2011.

[30] H. Tsutsui, H. Nakamura, R. Hashimoto, H. Okuhata, and T. Onoye, "An FPGA implementation of real-time Retinex video image enhancement," in *Proc. IEEE World Autom. Congr.*, Sep. 2010, pp. 1–6.

[31] S.-A. Li and C.-Y. Tsai, "Low-cost and high-speed hardware implementation of contrast-preserving image dynamic range compression for full-HD video enhancement," *IET Image Process.*, vol. 9, no. 8, pp. 605–614, 2015.

[32] Y. H. Shin, S. Jeong, and S. Lee, "Efficient naturalness restoration for non-uniform illumination images," *IET Image Process.*, vol. 9, no. 8, pp. 662–671, 2015.

[33] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Proc. IEEE Eur. Test Symp.*, May 2013, pp. 1–6.

[34] H. Seong, C. E. Rhee, and H. Lee, "A novel hardware architecture of the Lucas–Kanade optical flow for reduced frame memory access," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 6, pp. 1187–1199, Jun. 2016.

[35] E. H. Land and J. J. McCann, "Lightness and Retinex theory," *J. Opt. Soc. Amer.*, vol. 61, no. 1, pp. 1–11, 1971.

[36] D. J. Jobson, Z.-U. Rahman, and G. A. Woodell, "A multiscale Retinex for bridging the gap between color images and the human observation of scenes," *IEEE Trans. Image Process.*, vol. 6, no. 7, pp. 965–976, Jul. 1997.

[37] D. Terzopoulos, "Image analysis using multigrid relaxation methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 2, pp. 129–139, Mar. 1986.

[38] R. Kimmel, M. Elad, D. Shaked, R. Keshet, and I. Sobel, "A variational framework for Retinex," *Int. J. Comput. Vis.*, vol. 52, no. 1, pp. 7–23, 2003.

[39] E. Land, "An alternative technique for the computation of the designator in the Retinex theory of color vision," *Proc. Nat. Acad. Sci. USA*, vol. 83, pp. 3078–3080, May 1986.

[40] A. Moore, J. Allman, and R. M. Goodman, "A real-time neural system for color constancy," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 237–247, Mar. 1991.

[41] D. J. Jobson, Z.-U. Rahman, and G. A. Woodell, "Properties and performance of a center/surround Retinex," *IEEE Trans. Image Process.*, vol. 6, no. 3, pp. 451–462, Mar. 1997.

[42] S. Wang, J. Zheng, H.-M. Hu, and B. Li, "Naturalness preserved enhancement algorithm for non-uniform illumination images," *IEEE Trans. Image Process.*, vol. 22, no. 9, pp. 3538–3578, Sep. 2013.

[43] Y. Gao, H.-M. Hu, B. Li, and Q. Guo, "Naturalness preserved nonuniform illumination estimation for image enhancement based on Retinex," *IEEE Trans. Multimedia*, vol. 20, no. 2, pp. 335–344, Feb. 2018.

[44] *AMBA AXI4-Stream Protocol Specification*. Accessed: Aug. 29, 2018. [Online]. Available: http://www.arm.com

[45] *Xilinx Intellectual Property Video and Image Processing Pack*. Accessed: Aug. 29, 2018. [Online]. Available: http://www.xilinx.com

[46] *Display Input Lag Database*. Accessed: Sep. 19, 2018. [Online]. Available: https://displaylag.com/display-database/

[47] *NASA Retinex Image Processing*. Accessed: Aug. 29, 2018. https://dragon.larc.nasa.gov

[48] *Virtex-4 FPGA User Guide*. Accessed: Apr. 22, 2019. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug070.pdf

[49] *7 Series FPGAs Configurable Logic Block*. Accessed: Apr. 22, 2019. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf

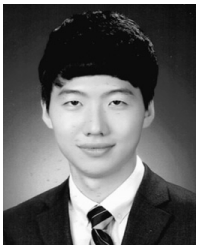
[50] *FPGA Logic Celss Comparison*. Accessed: Apr. 22, 2019. [Online]. Available: http://ee.sharif.edu/~asic/Docs/fpga-logic-cells_V4_V5.pdf
[51] J. Choi, B. Kim, H. Kim, and H.-J. Lee, "A high-throughput hardware accelerator for lossless compression of a DDR4 command trace," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 1, pp. 92–102, Jan. 2019.
[52] G. D. Licciardo, A. D'Arienzo, and A. Rubino, "Stream processor for realtime inverse tone mapping of full-HD images," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 11, pp. 2531–2539, Nov. 2015.
[53] P. Ambalathankandy, A. Hore, and O. Yadid-Pecht, "An FPGA implementation of a tone mapping algorithm with a halo-reducing filter," *J. Real-Time Image Process.*, vol. 116, no. 4, pp. 1317–1333, Sep. 2016.
[54] G. M. S. Nunes, "Evaluation of tone-mapping algorithms for focal-plane implementation," M.S. thesis, Dept. Elect. Eng., Fed. Univ. Rio de Janeiro, Rio de Janeiro, Brazil. Accessed: Apr. 22, 2019. [Online]. Available: http://www.pee.ufrj.br/index.php/pt/producao-academica/dissertacoes-de-mestrado/2018/2016033238-evaluation-of-tone-mapping-algorithms-for-focal-plane-implementation/file
[55] *Virtex-6 FPGA Configurable Logic Block*. Accessed: Apr. 22, 2019. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug364.pdf
[56] G. H. KleinJan *et al.*, "Optimisation of fluorescence guidance during robot-assisted laparoscopic sentinel node biopsy for prostate cancer," *Eur. Urol.*, vol. 66, no. 6, pp. 991–998, Dec. 2014.
[57] R. Dutta *et al.*, "Clinical comparison of conventional and mobile endockscope videocystoscopy using an air or fluid medium," *J. Endourol.*, vol. 31, no. 6, pp. 593–597, Jun. 2017.

Jin Woo Park (S'19) received the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2014, where he is currently working toward the integrated M.S. and Ph.D. degrees in electrical and computer engineering.

His current research interests include accelerating image matching and image enhancement algorithms using field-programmable gate array (FPGA).

Hyokeun Lee (S'19) received the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2016, where he is currently working toward the integrated M.S. and Ph.D. degrees in electrical and computer engineering.

His current research interests include nonvolatile memory controller design, hardware persistent model for nonvolatile memory, and computer architecture.



Boyeal Kim (S'19) received the B.S. degree in electrical and computer engineering from the Seoul National University of Seoul, Seoul, South Korea, in 2017, where he is currently working toward the integrated M.S. and Ph.D. degrees in electrical and computer engineering.

His current research interests include high bandwidth memory, computer architecture, and hardware accelerator design.

Dong-Goo Kang received the B.S. degree (*summa cum laude*) in electronics engineering from Sogang University, Seoul, South Korea, in 2000, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2002 and 2007, respectively.

From 2007 to 2015, he was a Research Staff Member with the Samsung Advanced Institute of Technology (SAIT), Suwon, South Korea. He is currently a Senior Researcher with the Korea Electrotechnology Research Institute (KERI), Seongju-dong, South Korea. His current research interests include image processing and computer vision for medical imaging.

Dr. Kang is a member of the Alpha Sigma Nu Honor Society. He was a recipient of the 11th Samsung Humantech Paper Award (Gold Prize) by Samsung Electronics, Co., Ltd., in 2005, and the SAIT Researcher of the Year by the Samsung Advanced Institute of Technology in 2012. He has served as a Program Committee Member of the International Forum on Medical Imaging in Asia (IFMIA) in 2011 and 2012, and currently serves on the Board of Directors of the Institute of Electronics and Information Engineers (IEIE) and also serves as an Editor for the IEIE.

Seung Oh Jin received the B.S. and M.S. degrees in electrical engineering from Changwon National University, Changwon, South Korea, in 1996 and 1998, respectively, and the Ph.D. degree in Nanoscale Semiconductor Engineering from Hanyang University, Seoul, South Korea, in 2005.

Since 1998, he has been with the Korea Electrotechnology Research Institute (KERI), Seongju-dong, South Korea, where he is currently a Principal Researcher. He developed several medical imaging and image processing systems for digital radiography, tomosynthesis, and computed tomography. His current research interests include embedded systems, image processing, compressed sensing, and medical imaging@perio.

Hyun Kim (M'12) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2009, 2011, and 2015, respectively.

From 2015 to 2018, he was a BK Assistant Professor with the BK21 Creative Research Engineer Development for IT, Seoul National University. In 2018, he joined the Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul, where he is currently an Assistant Professor. His current research interests include the areas of algorithm, computer architecture, memory, and SoC design for low-complexity multimedia applications and deep neural networks.

Hyuk-Jae Lee (M'04) received the B.S. and M.S. degrees in electronics engineering from Seoul National University, Seoul, South Korea, in 1987 and 1989, respectively, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 1996.

From 1998 to 2001, he was a Senior Component Design Engineer with the Server and Workstation Chipset Division, Intel Corporation, Hillsboro, OR, USA. From 1996 to 1998, he was with the Faculty of the Department of Computer Science, Louisiana Tech University, Ruston, LS, USA. In 2001, he joined the School of Electrical Engineering and Computer Science, Seoul National University, where he is currently a Professor. He is a Founder of Mamurian Design, Inc., Seoul. His current research interests include the areas of computer architecture and SoC design for multimedia applications.