Refinement of a real-time super-resolution FPGA circuit

TAKASHI MATSUMOTO†
MAYO SANADA† SUZUKA YASUNAMI† KAZUKI JOE†
† Nara Women's University

Abstract – *In this paper, we report on a method to improve the real-time super-resolution system which we developed previously in preparation for future advanced functions. In order to perform super-resolution processing in real time by using ICBI (Interactive Curvature Based Interpolation) algorithm, we developed a system with ICBI implemented by hardware using FPGA (Field Programmable Gate Array).However, in this system, the capacity limit of the block RAM used for the frame buffer is strict and it is difficult to add further functions. Recent high-performance FPGA chips are developed as SoC (System on Chip), and high-performance CPU, high-performance bus, high-speed DDR interface are included as hardware macros. By utilizing the functions of these hardware macros, we improve the super-resolution system to eliminate the constraints on capacity and prepare for further higher functionality of the system.*

Keywords: FPGA, ICBI, GPU

1 Introduction

In recent years, the resolution of display devices such as personal computers and TV displays has improved, while many low resolution video data such as small cameras and old video are still present. Namely, the resolution of input devices and contents is not catching up. When input data with low resolution is displayed on a display device with high resolution, it is coarse and blurred due to the difference in resolution. In order to compensate for this difference in resolution, a technique called super resolution for interpolating low-resolution images and generating images with high resolution has been worked out, and various interpolation algorithms have been studied.

Various interpolation techniques have been developed for a long time in order to enlarge the image at the same resolution even before it becomes important to improve the resolution. Representative examples include the nearest neighbor method, the bilinear method, the bicubic method, etc. [1].However, these image interpolation techniques do not have sufficient quality after interpolation. Under these circumstances, a super resolution algorithm has been developed that interpolate images consciously based on edges: the NEDI (New Edge - Directed Interpolation) algorithm [2].However, when incorporating super resolution technologies such as security cameras, endoscopic cameras and inspection cameras are used in the real society, it is important to see easily in a short processing time for human beings, while it require huge computational cost to calculate NEDI, which is unsuitable for real time processing.

At that time, the ICBI (Interactive Curvature Based Interpolation) algorithm has been developed [3] that yields comparable results with much less computational complexity than NEDI [3].Although the computational complexity of the ICBI algorithm has been greatly reduced as compared with NEDI, when the software implementation is performed on a high-performance CPU, processing time of several seconds per image is required of course depending on the size of the image, and it is far from the real time processing.

Since the super-resolution algorithm has a large amount of computation, it is difficult to shorten the execution time by merely improving the software aspect. For this reason, it is necessary to implement the ICBI algorithm by hardware using FPGA (Field Programmable Gate Array) which can be obtained cheaply and change the circuit many times as needed. So we proposed a method to implement as the hardware dedicated to the super resolution algorithm [4] [5].

The rest of the paper is constructed as follows. In section 2 we explain the real-time super-resolution hardware system by FPGA we have proposed and prototyped actually. In section 3, we point out the problems of the proposed hardware and describe the solution method using SoC (System on Chip) type FPGAs that have become popular recently. In section 4 details of the solution method is presented. In section 5 we discuss the smart function of the improved super resolution hardware.

2 Real time super resolution system before improvement

2.1 Basic structure of hardware system

Before designing and prototyping the real-time super-resolution hardware, we use a camera module equipped with OmniVision's OV7670 [6], which is inexpensive and easy to acquire materials, to achieve real time display by doubling the resolution both vertically and horizontally. In this system, it is necessary to handle two types of video signals: input camera signal and output display signal. The OV 7670 camera module provides a VGA compliant video signal, and the dot clock is about 25 MHz. Since the video signal on the display side has twice the resolution both in the vertical and horizontal

directions of the VGA, the dot clock is expected to be about 4 times as high as 25 MHz. In other words, this system needs to deal with at least two types of clocks, that is, a clock followed by the input side signal and a clock to be followed by the output side signal. Also, since the input video signal and the output video signal have independent dot cooks, it is inevitable that horizontal and vertical synchronization signals are independent each other. For this reason, there is no association between the position in the screen of the input signal at the same time point and the position of the output signal in the screen. Unless there is a frame buffer capable of storing the input image information for at least one frame, the video output is impossible.

When estimating the minimum capacity of this frame buffer, since the OV7670 outputs 16 bits of data per pixel and the resolution is 640 × 480, the capacity should be 307,200 × 16 bits. Built-in block RAM of XC7Z020-1CLG484C (Xilinx Corporation) [8] mounted on the FPGA evaluation board (ZedBoard [7]) used for prototyping can constitute a maximum of 262,144 entries with 16 bit width. This is because the number of entries of the block RAM is limited to the power of 2, and the number of next level entries is 524,288, which exceeds the maximum amount (286,720 × 18 bits) of block RAM resources built in the chip, .Since we did not consider using memory other than built-in FPGA at the design examination stage of the system, we use only the area within 262,144 x 16 bits of the output of OV7670.In other words, although OV7670 can be used up to 480 lines, we use only up to 400 lines, and discard the input data beyond that. Since the block RAM can be used as a 2-port memory in which reading and writing are independent, we change the input side clock and the output side clock to the frame buffer as well. We execute the image interpolation processing for super-resolution in the image processing pipeline that runs with the output clock.

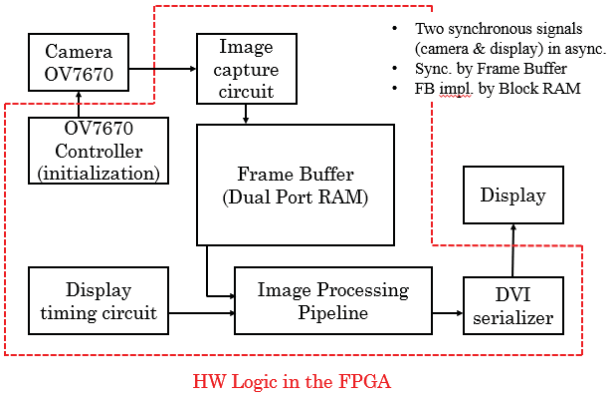

Figure 1 Structure of the real-time super-resolution system

The flow of data in the super resolution system is explained with FIG.1. First, raster data output from the camera is temporarily stored in a frame buffer (Dual Port RAM), and the image input of the camera and the image output of the display are synchronized. Next, image processing is carried out with the image processing pipeline. In the image processing pipeline, the pixel interpolation pipeline is included.

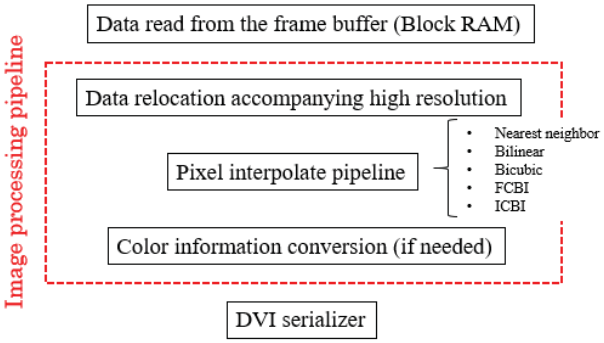

Figure 2 Data flow of image pipeline

2.2 Structure of image processing pipeline

The configuration of the image processing pipeline is shown in Fig.2.Image information read out from the frame buffer is rearranged as the resolution is doubled in the vertical and horizontal directions, interpolated (super-resolved) processing is performed in the pixel interpolation pipeline, converted to RGB Format color information for DVI output, and sent to the display. In order to proceed concretely, it is necessary to determine the representation format of pixel data. Since the output of OV7670 is 16 bits per pixel, 8 bits per pixel cannot be secured for luminance information in RGB format. Human visual sense has the resolution capability for the luminance information higher than the resolution capability for color information, so the YUV 422 format which can secure the luminance information 8 bits per pixel is adopted as pixel data. The designation of YUV 422 comes from the manual of OV 7670, and it seems that YCbCr4:2:2 is more officially named. In the YUV 422, luminance information (Y) is output to each pixel in 8 bits, and U and V which are color difference information are outputted every 8 bits at every other pixel. That is, U and V of color difference information are shared by two adjacent pixels. In the frame buffer of the super-resolution system, since the memory capacity is tight as described above, the image data is stored in YUV 422 format that is the output with luminance 8 bits of OV7670 without adopting the standard format of RGB 24 bits. Because we use YUV 422 format, when raising the resolution to twice the length and width, neighboring pixels are not adjacent after high resolution. From this fact, it is necessary to rearrange the data. Also, if interpolation processing is performed with the YUV 422 format in the pixel interpolation pipeline, since the video output of the DVI format is each 8 bits of RGB in the digital format, in order to convert it into this format, color difference information from YUV 422 to RGB conversion stage is required in the pipeline. For the pixel interpolation pipeline, the ICBI algorithm has been modified for implementation in the proposed hardware system, but other interpolation algorithms can be implemented, and other algorithms can be implemented with much less development effort than ICBI without modification. In addition, since the system which is prototyped in this paper only converts to the

resolution twice in length and width, just the luminance information (Y) is interpolated and the color information is handled by the nearest neighbor method.

2.3 Pixel interpolation pipeline configuration

The pixel interpolation pipeline is a part of the image processing pipeline. We briefly explain the pixel interpolation pipeline that realizes the modified ICBI algorithm for hardware implementation. The ICBI algorithm includes the FCBI algorithm as the preprocessing for obtaining the initial value of the pixel whose resolution is doubled in the vertical and horizontal directions. The FCBI calculates the luminance value of the pixel to be interpolated divided into twice from the direction of the diagonal directions and the direction of each of the vertical and horizontal directions. For oblique interpolation, only the original image point is used (only one point is used in the 2 × 2 pixel range), so the calculation is performed by first inputting it to the pipeline. However, since interpolation in the vertical and horizontal directions requires data of points calculated by the oblique interpolation, the calculation in the vertical and horizontal directions needs to be performed after the value of the pixel calculated in the diagonal directions is obtained. In addition, ICBI performs smoothing to curvatures after FCBI application. Therefore, in the original ICBI algorithm, FCBI in the oblique direction and curvature smoothing only in the oblique directions are performed, and FCBI in the vertical and horizontal directions and curvature smoothing in just the vertical and horizontal directions are performed. However, the process of curvature smoothing is a hill climbing process of repeatedly changing the pixel data of interest slightly, examining the change of the evaluation function, and judging whether to change the evaluation function. It is expected that the number of pipeline stages to be done is considerably larger. Since the curvature smoothing in the oblique directions and the vertical and lateral directions are independent from each other, it can be integrated into a single pipeline circuit. By using the curvature smoothing circuit that combines the above two, it becomes possible to reduce the number of cascade stages of pipeline circuits for smoothing by half. Therefore, instead of executing the procedure of the original ICBI algorithm, the pipeline complete the processing in the oblique directions and the vertical and horizontal directions of the FCBI first, and thereafter performing the curvature smoothing in the diagonal directions and the vertical and horizontal directions at the same time. We design the circuits with cascade connections in multiple stages. Since the ICBI is divided into the FCBI part and the curvature smoothing part, the possibility that the delayed convergence of the smoothing processing is worried, but no problem is found in trial production or software emulation.

Curvature smoothing processing pipeline can be arranged and wired by LSI CAD by connecting up to 8 circuits. We are checking the influence of the number of pipeline stages for the curvature smoothing circuit. For this purpose, cascade connection is made for each of 4, 7 and 8 circuits. To implement the smoothing process of ICBI, noise is generated in the output when operating the design with the 7 and 8 cascade-connected circuits , and it is impossible to operate stably. Although It is a phenomenon peculiar to the evaluation board (chip is susceptible to noise), this is also considered to be caused by the too large number of pipeline stages .There may be a difference in the number of stages where noise is generated, but it seems to be a general problem that it can be observed also by other evaluation boards if the number of stages becomes sufficiently large. Since it does not operate stably when the number of the curvature smoothing circuit stages is large, we decided that ZedBoard does not work well unless the smoothing pipeline circuit is suppressed to about 4 circuits in ICBI. At this time, the number of cascade connection stages of the smoothing processing pipeline is the number of curvature smoothing iterations. When implementing ICBI algorithm with software as 4 smoothing pipeline circuits, dusty pixels appear around the edges. This is because the dust generated by the FCBI algorithm can not be removed by the 4 ICBI curvature smoothing pipeline circuits.

Instead of decreasing the number of circuits, we increase the value of the change amount of the Hill Climb method by a factor of two to enhance the effect of one iteration of the smoothing process. This is called the ICBI boost version in the FPGA implementation. As a result of experiment by adding the boost version ICBI algorithm to the normal version algorithm, good results are obtained even when the number of circuits is 4 circuits. Comparing the boost version of 4 ICBI circuits with the normal version of 4 ICBI circuits, we obtain good output results of the boost version ICBI, so we decide to adopt the boost version ICBI algorithm for ICBI hardware implementation.

2.4 Display output

Since there is a constraint that the frame buffer storing the image data before resolution enhancement is about 640 × 400, the display output format can be a resolution of around 1280 × 800 and we adopt a standard with the lowest possible frequency. The reason for taking care not to make the dot frequency too high is that the DVI signal is taken out to the HDMI socket via the PMOD terminal to be connected to the display, and the allowable maximum frequency of the simple PMOD terminal is not high .Considering these, development is carried out adopting 1280 × 768 @ 60 Hz 15: 9, dot clock: 68 MHz Progressive as the output format of the display. Even with a dot clock of 68 MHz, a double data rate signal of 340 MHz which is five times actually is flowing to the PMOD terminal. However, since there are no cables around the wiring, problems such as noise do not occur.

3 System problems and improvement policy

3.1 System problems

Since there was an emphasis on demonstrating that the ICBI algorithm can be hardware-implemented by FPGA at the time of trial manufacture of the pre-modification system, viewpoints such as verifiability and function extensibility of

the prototype system were not taken into consideration. Since the frame buffer is configured in the block RAM inside the FPGA, it was necessary to design and develop the pattern generation circuit every time even if we tried to display the regular pattern on the frame buffer and test whether the interpolation works correctly. Furthermore, since the super-resolution (interpolation) result is only drawn on the display, it is difficult to verify whether the interpolation is completely correct numerically. That is, there is no memory that can be verified later by saving the interpolation result. Also, despite the fact that the OV7670 is a camera capable of outputting a VGA signal of 640 × 480, it was inevitable to use the OV 7670 at 640 × 400 or less because of the block RAM capacity limitation. From now on, in searching for advanced functions of the system, we are keenly aware of the need for a frame buffer that can store multiple frames regardless of input or interpolation results.

3.2 SoC type FPGA chip

The block RAM has the ability to be used as a dual-port memory and transfer data between digital circuits of different clocks while it consumes almost as a frame buffer for temporarily storing images. That means we use it as a very wasteful resource. Actually, ZedBoard used for prototype has 512 Mbyte of DDR memory installed. The data path and control part of the DDR memory do not belong to the PL (Programmable Logic) part which is the FPGA part of the XC7Z020-1CLG484C chip, but belong to the PS (Processing System) part. At the prototype before improvement, we tried to demonstrate that the ICBI algorithm can be hardware-implemented by utilizing the FPGA, so we did not consider using resources belonging to the PS part. Also, since the PS part is composed of a high-performance bus centering on an embedded processor with a hard macro, there was also a concern that skills and knowledge other than the FPGA designing technology are required for the development.

However, in recent years, a lot of SoC type FPGA chips incorporating a number of hard macro CPUs, communication interfaces and peripheral circuits have appeared, prices have also become cheaper and the availability is getting better. For example, in the case of Xilinx, the Zynq-7000 SoC [10] series used for ZedBoard, Zynq UltraScale+ MPSoC [11] series, in the case of intel (formerly Altera), Cyclone V SoC [12] series and Arria 10 SoC [13] series fall under this category. The tendency that adopts this FPGA + SoC chip configuration is not a flow for expanding the variety by merely combining heterogeneous circuits, but it is considered to be essential. This is because it is more efficient to embed frequently used circuits in hard macros as the integration degree of LSIs has increased and enormous circuits can be integrated on a chip nowadays. No matter how the synthesis technology of the hardware circuit or the optimum placement and routing technology advances, the soft macro circuit cannot compare the operation speed and the area size with respect to the hard macro circuit. Users who do not need hard macro circuits just do not use them. It is more likely that reducing the development and manufacturing costs of chips is possible by enlarging the target users of the chips rather than having users who do not use the circuits. For these reasons, SoC type FPGA chips are becoming mainstream in the future, and applications that sufficiently perform with built-in CPUs and embedded circuits are handled by programs and only in applications where the processing speed cannot be met by programs, some circuits composed of FPGA will be used.

3.3 Problem improvement policy

Considering the tendency toward the SoC type FPGA chip is essential, the reason for configuring the system with avoiding the use of the PS part function is just the lack of knowledge for using the PS part. If we take a frame buffer on the DDR memory of ZedBoard, the capacity constraint will actually be gone. Since the built-in CPU can read from and write to the frame buffer, it is easy to supply inputs to be super-resolved (interpolated) from other than the camera, and take out the conversion results to transfer them to the PC or other devices. When taking a frame buffer on the DDR memory, since the data transfer speed is not at the level enough for the software on the embedded CPU, writing and reading must be performed with DMA. By using Xivinx's development support tool Vivado [9] we find that the template of the DMA transfer circuit is generated automatically, so we modify the source generated by this function to develop the DMA transfer function for our system.

4 Details of improvement method

4.1 Specification of DMA transfer

In the Zynq-7000 SoC series including XC7Z020-1CLG484C used in the prototype system, AXI (Advanced eXtensible Interface) is used for the DMA transfer with DDR memory. Since the DMA transfer circuit is formed by the FPGA in the PL part, the high-performance AXI port which is an interface between the PS and the PL is used for the DMA transfer to the DDR memory. The Zynq chip's high-performance AXI port itself can select 32 bit or 64 bit as the transfer width of DMA transfer, but in the ZedBoard environment, because only the generation of 32 bit wide DMA transfer circuit is selected on Vivado, we decided to use 32 bit wide DMA transfer. Regarding the data transfer rate (transfer clock) in the DMA transfer, the data transfer clock had better be faster considering the bandwidth, but it is set to 200 MHz in consideration of the operation results in ZedBoard. In the DMA circuit which is semi-automatically generated by Vivado, the burst transfer length of the DMA transfer can be set to a fixed number up to 256 transfers, and as the burst length is longer, the transfer bandwidth can be increased by lowering the overhead ratio by transfer. However, since DMA transfer is performed in units of the line length of the image, if one burst transfer size is not a common divisor of the line length, an overhead due to transmission waste occurs. Since one line size is 1,280 bytes of input 640 × 16 bits, the burst length is set to 64 (64 × 32 bits = 256 bytes).Since the DDR memory is used for the frame buffer, the image data representation on the frame buffer can be used as a VRAM with RGB 8 bits each without any problem of capacity. For future functions to give extra bandwidth to the DDR memory

for expansion, we decide to follow the 16 bit data format per pixel of the YUV 422 format as usual. As a result, the configuration after the image processing pipeline is the same as the system before the improvement.

4.2 Exchange between three types of data transfer clocks

The system before improvement uses 1,280 × 768 60 Hz dot clock 68 MHz display standard but after improvement it can display 640 × 480 in full size so it can display at 1,280 × 960 in the display video standard. We decide to adopt the display video standard with the dot clock as low as possible. As a result, 1,280 × 960 60 Hz dot clock 108 MHz Progressive is adopted.

The system before improvement is operated based on two types of clocks in the FPGA circuit, that is, a data output clock from the camera and a dot clock for display, and the exchange between the clocks is performed in the frame buffer by the block RAM (See Fig. 1).In the figure, the camera output is 8 bit wide because the camera module outputs 16 bits of YUV 422 data in 2 clocks at 8 bits each.

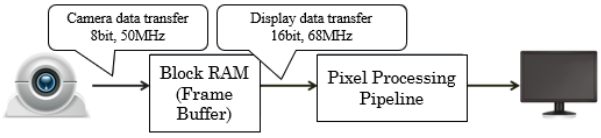

Figure 3 Data flow of the system before improvement

On the other hand, in the improved system, writing and reading of data to and from the DDR memory are performed based on a clock of 200 MHz, data transfer of 8 bit width 50 MHz of the camera output is converted to 32 bit width 12.5 MHz before data writing, and then exchange it with a 200 MHz clock for DMA transfer to the DDR memory afterwards. Since the data output clock of the camera and the DMA transfer clock of the DDR memory are basically asynchronous, it is necessary to insert a FIFO buffer by dual port memory using a block RAM for exchanging the clocks. Also, unlike dual port memory, DDR memory can only be accessed from one master at a time with one port. Both data input from the camera and data output to the display are periodically performed irrespective of the idle state of the DDR memory ports. Therefore, it is indispensable to insert the FIFO buffer. Although the capacity of the FIFO buffer is sufficient for one line, we decide to use a 1,024 × 32 bit FIFO buffer this time. Likewise, as for read data transfer from the DDR memory, by inserting a FIFO memory (1,024 × 32 bits) by the block RAM, the DMA clock of 200 MHz is exchanged to the display dot clock of 108 MHz. Figure 4 shows the data flow of the system after improvement.

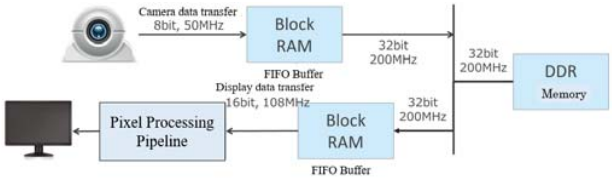

Figure 4 Data flow of the system after improvement

4.3 Semi-automatic generation of AXI DMA circuit by Vivado

The reason why we did not consider the way to utilize DDR memory in the first prototype is not only the aggressive idea of confirming the possibility of implementation with just the PL part (FPGA part) but also the problem of no developing experience using the AXI DMA circuit as a SoC type FPGA. While studying various literature and materials, we get to know Vivado which is a design support tool of Xilinx makes it possible to semi-automatically generate a DMA circuit and we develop it using the function (Create and Package IP).However, since at least Vivado v2015.4.2 we use just generates AXI side circuits as a template, we read from the source code of the generated hardware description language to add the interface of user circuit side and modify the template. For example, in the case of the DMA read circuit, a circuit in which 4 Kbyte of data is always read out with burst transfer is generated. In our case, it is sufficient to read one line of camera image at one DMA transfer, so we add an interface that terminates the DMA transfer. Also, not on the AXI side, we need to add a description in accordance with the interface on the user PL side. Although Vivado's AXI IP semi-automatic generation function enables us to develop DMA circuits without investigating access arbitration of DDR memory etc., that is, limited knowledge about AXI specifications, some ability to read, understand, and modify the generated circuit description is required. In addition, adding IP using AXI to the block diagram we automatically obtain the required reset circuit and the memory interconnect circuit to perform automatic wiring of AXI related signals. Although it is a very grateful function, if DMA-related problems occur, it would not be possible to understand the operation of circuits that we did not design, and it could rather be a factor that increases the trouble in some causes.

4.4 Operation of FPGA side interface in DMA circuit

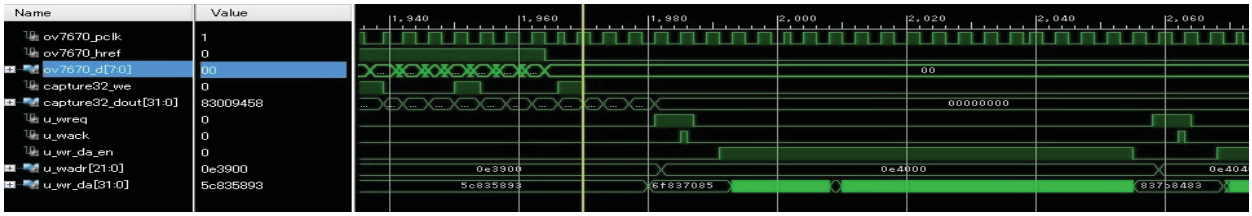

Figure 5 Operation around the FIFO buffer on the camera input side (detailed)

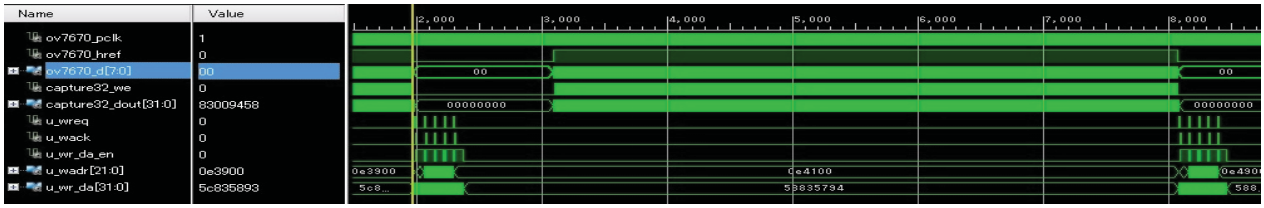

Figure 6 Operation around the FIFO buffer on the camera input side (overview)

Both the input side from the camera and the output side to the display process in units of horizontal lines (raster) of the image. Since the bandwidth of the DMA circuit is larger, it is just necessary that the data in the FIFO buffer is not exhausted or overflowed in the middle of the line. In anticipation of safety, both start the DMA transfer in a horizontal blanking period. The camera input issues a DMA write request (u_wreq) immediately after the arrival of one line data, and after receiving the u_wack signal from the DMA circuit, the DMA burst transfer of 32 bit data is performed according to the u_wa_da_en signal (FIG. 5). Output data of OV7670 for one line is written in the DDR memory by five burst transfers (FIG. 6).

Similarly on the display side, necessary data is read from the DDR memory by DMA in the next line during the blanking period of the horizontal line before display. The DMA read transfer is activated by the line_req signal, and the read data sent by the burst transfer according to the line_data_en is fetched into the FIFO buffer. To terminate the burst transfer in five times, the line_continue signal is negated at the start of the fifth burst transfer (FIG. 7).Since the display output side is interpolated not only in the horizontal direction but also in the vertical direction by a factor of 2, camera images are read from the frame buffer every other line (FIG. 8).

5 Toward further function expansion

By this improvement, since the frame buffer is transferred from the block RAM of the PL part to the DDR memory on the ZedBoard, the capacity constraint of the frame buffer has been eliminated. As a result, image data other than the camera input can be drawn in the frame buffer by the embedded CPU and processed by the interpolation pipeline. By just adding a DMA circuit that writes the interpolation result to the frame buffer, it is also possible to easily verify the interpolation result.

The super-resolution system currently has the function of improving the resolution twice in both vertical and horizontal directions with the VGA resolution camera input in this paper. When using this system as a display device for security cameras and inspection cameras, the requirement that the attention point is to be further enlarged in real time is desired. In this enlargement process, we think that the ICBI algorithm with less unnatural jaggies and edge blur is suitable. It is possible to construct a cascade of our multiple systems, four times in both vertical and horizontal directions, three times in vertical and horizontal directions by two, and sixteen in vertical and horizontal directions with four units, but the cost increases and the scale of the apparatus also increases. This apparatus can realize interpolation of 2 times length and width by the ICBI algorithm for 60 movie frames per second with no delay. However, for people watching the surveillance or examination cameras, it is highly likely that 60 movie frames per second are unnecessary. Therefore, instead of 30 images per second, there is a method of extending to a system which performs vertical and horizontal interpolation by 4 times, horizontal interpolation by 8 times instead of 20 images per second, and horizontal and vertical 16 magnification display instead of 15 images per second are conceivable. The image with the interpolation degree in the intermediate state is written back to the frame buffer and enlarged in the vertical and horizontal directions at every display period of one frame and output the image with the target magnification to the display. Since the display is maintained at about 1,280 × 960, in the case of enlarged display of 4 times or more, only a partial area of the camera input is drawn on the display. The enlarging target position should be indicated with a mouse or other devices. Since the frame rate of the display is fixed at 60 Hz, for example, in the case of a 16-fold enlarged image, a 640 × 480 camera input image, a 640 × 480 vertical and horizontal two times enlarged image, a 640 × 480 vertical and horizontal 4 times enlarged image, and a 640 × 480 vertical and horizontal 8 times enlarged image exist at the same.The original image from the camera is updated once every four frames.

6 Conclusions

In this paper, we report on a method to improve the real-time super-resolution system which we developed earlier in

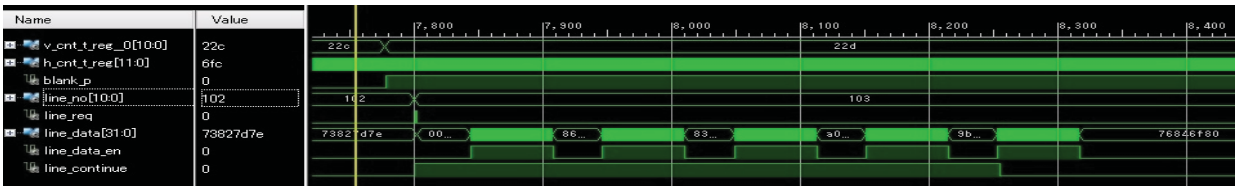

Figure 7 FIFO buffer input operation on display output side (detailed)

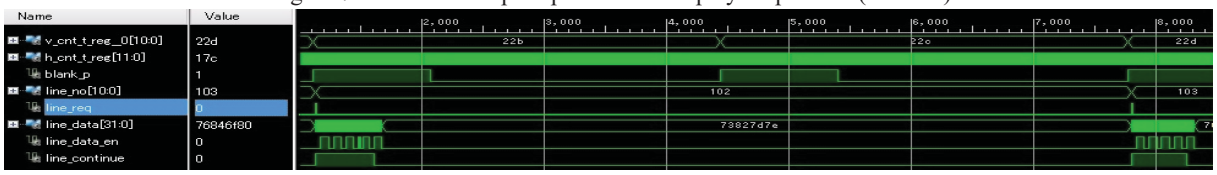

Figure 8 FIFO buffer input operation on display output side (overview)

preparation for future advanced functions. In order to perform the super resolution processing in real time using the ICBI algorithm on video, we developed a system with hardware implementation of ICBI by FPGA. However, in this system, the capacity limit of the block RAM used for the frame buffer is severe, and it is difficult to add further functions. Recent high-performance FPGA chips are becoming SoC, and high-performance CPU, high-performance bus, high-speed DDR interface are incorporated as hard macros. By utilizing the functions of these hard macros, DDR memory is used as a frame buffer, thereby improving the super resolution system and eliminating the capacity constraint. This improvement showed that it is feasible to realize the system that can change the display enlargement rate in real time using the ICBI algorithm which can be easily seen and expanded. Figure 9 shows that the image was displayed at 1,280 × 960 60 Hz which was impossible in the system before improvement, and we confirm that the improvement is successful.

7 References

[1] M. Okutomi, "Digital Image Processing", CG-ARTS asoc., 2004

[2] X. Li and M. T. Orchard, "New edge-directed interpolation" IEEE Trans. on Image Proc., 10:1521–1527, 2001.

[3] Andrea Giachetti Nicola Asuni, "Real-Time Artifact-Free Image Upscaling" IEEE Transactions on Image Processing 20(10):2760 – 2768, 2011

[4] Takashi Matsumoto, Arisa Yamamoto, Kazuki Joe: Real-Time Super Resolution: FPGA Implementation for the ICBI Algorithm, 2016 International Conference on Parallel and Distributed Processing Techniques and Applications, Final Edition, pp.415-420 (2016).

[5] EPFL, "Extension module with ov7670 CMOS camera" https://wiki.epfl.ch/prsoc/ov7670, (accessed 2018-05-06)

[6] Xilinx, "ZedBoard" https://japan.xilinx.com/products/boards-and-kits/1-8dyf-11.html, (accessed 2018-02-01)

[7] Xilinx, "Zynq-7000 All Programmable SoC" https://japan.xilinx.com/products/silicon-devices/soc/zynq-7000.html, (accessed 2018-02-01)

[8] Xilinx, "Vivado Design Suite" https://japan.xilinx.com/products/design-tools/vivado/vivado-webpack.html (accessed 2018-05-08)

[9] Xilinx, "Zynq-7000 Programmable SoC" https://japan.xilinx.com/products/silicon-devices/soc/zynq-7000.html, (accessed 2018-05-08)

[10] Xilinx, " Zynq UltraScale+ MPSoC" https://japan.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html (accessed 2018-05-08)

[11] intel. "Cyclone® V SoC" https://www.altera.co.jp/products/soc/portfolio/cyclone-v-soc/overview.html, (accessed 2018-05-08)

[12] intel, " Arria® 10 SoC" https://www.altera.co.jp/products/soc/portfolio/arria-10-soc/overview.html (accessed 2018-05-08)

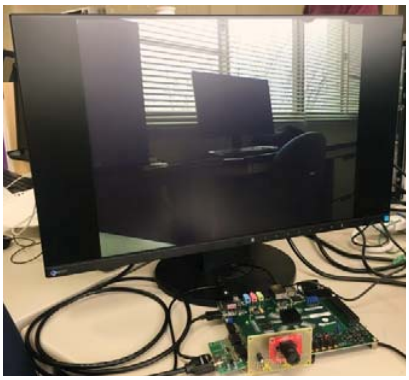

Figure 9 Display example by the improved system