A Real-Time FHD Learning-Based Super-Resolution System Without a Frame Buffer

Ming-Che Yang, Kuan-Ling Liu, and Shao-Yi Chien, *Member, IEEE*

***Abstract*—This brief presents a real-time learning-based super-resolution (SR) system without a frame buffer. The system running on an Altera Stratix IV field programmable gate array can achieve output resolution of 1920 × 1080 (FHD) at 60 fps. The proposed architecture performs an anchored neighborhood regression algorithm that generates a high-resolution image from a low-resolution image input using only numbers of line buffers. This real-time system without a frame buffer makes it possible to integrate SR operation into image sensors or display drivers carrying out computational photography and display.**

***Index Terms*—Super resolution, anchored neighborhood regression, real-time, FPGA.**

I. Introduction

SINGLE image super-resolution (SR) aims to generate high-resolution (HR) images from low-resolution (LR) images. It is a well-known, ill-posed problem since a single HR image could generate more than one LR image, and it requires enough prior knowledge to reconstruct the high-quality HR images. To solve this problem, learning-based single-image SR methods have achieved outstanding performance and gained state-of-the-art results, by learning from millions of external image patches.

In this brief, a learning-based regression super-resolution architecture without using a frame buffer is proposed. Referring to [1], we implement the system with 128 learned dictionaries on a field programmable gate array (FPGA). This system can achieve output resolution of 1920 × 1080 (FHD) at 60 fps using only numbers of line buffers.

II. Review of Super-Resolution Algorithms

This section illustrates the development of the example-based and dictionary learning-based single-image super-resolution algorithms. Different from using information of only the input testing image, example-based SR algorithms use knowledge from plenty of external HR and LR example image pairs to help generate output HR images. A classic example-based SR work was proposed by Freeman *et al.* [2].

Chang *et al.* [3] proposed a neighbor embedding method which assumes the local structure similarity of LR and HR patches' manifolds. The output HR image patch shares the same weight in its HR manifold as the input LR image patch in its local LR manifold. A reconstruction method such as locally linear embedding, proposed by Roweis and Saul [4], has also been used for getting fairly good results.

Yang *et al.* [5] takes sparsity prior into consideration using sparse coding over learning representative HR and LR patches jointly called a dictionary. Compared to Chang *et al.*, Yang *et al.* reduces the targeting candidates from numerous training patches to certain numbers of points in a dictionary, achieving encouraging results. However, the computation complexity takes quite a lot of time to resolve the sparse coding online while testing each patch input. Zeyde *et al.* [6] optimized the overall framework by means of the K-SVD algorithm [7]. They learned LR and HR dictionaries separately by obtaining the LR dictionary first and then generating the HR dictionary from the LR dictionary with sparse coding. There are great improvements in both the testing time and quality of resulting images in their work.

Timofte *et al.* [8] further proposed Anchored Neighborhood Regression (ANR) to improve the sparse decomposition, which is the bottleneck in Zeyde's algorithm. Instead of performing sparse decomposition online, Timofte *et al.* use selection of certain atoms in a dictionary called anchors, which are obtained during an offline training stage. Furthermore, for each selected anchor in the dictionary, the reconstruction operation can then be realized by corresponding linear ridge regression which is pre-trained during an offline stage also. Therefore, for each input LR testing patch $\mathbf{x}$, the output HR patch $\mathbf{y}$ is generated by a ridge regressor $\mathbf{R}_i$ which is trained on the local neighborhood $\mathbf{N}_l$ of a fixed amount of LR training patches. The learning procedure is shown as follows. First, the locally linear embedding coefficient vector α is derived by

$$\arg\min_{\alpha} \|x - N_l\alpha\|_2^2 + \lambda\|\alpha\|_2 \tag{1}$$

With α, the output HR patch $\mathbf{y}$ is then obtained by a closed-form solution:

$$y = N_h\alpha = N_h\left(N_l^T N_l + \lambda I\right)^{-1} N_l^T x = R_i x, \tag{2}$$

where $\mathbf{N}_\mathrm{h}$ are the corresponding HR training patches of the LR neighborhood $\mathbf{N}_\mathrm{l}$. These improvements make most of the

Manuscript received May 12, 2017; accepted August 30, 2017. Date of publication September 5, 2017; date of current version November 22, 2017. This work was supported by Sitronix Technology Corporation, and the EDA tools are supported by Chip Implementation Center, Taiwan. This brief was recommended by Associate Editor E. Alarcon. *(Corresponding author: Shao-Yi Chien.)*

The authors are with the Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: spencer@media.ee.ntu.edu.tw; klliu@media.ee.ntu.edu.tw; sychien@ntu.edu.tw).

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

Digital Object Identifier 10.1109/TCSII.2017.2749336

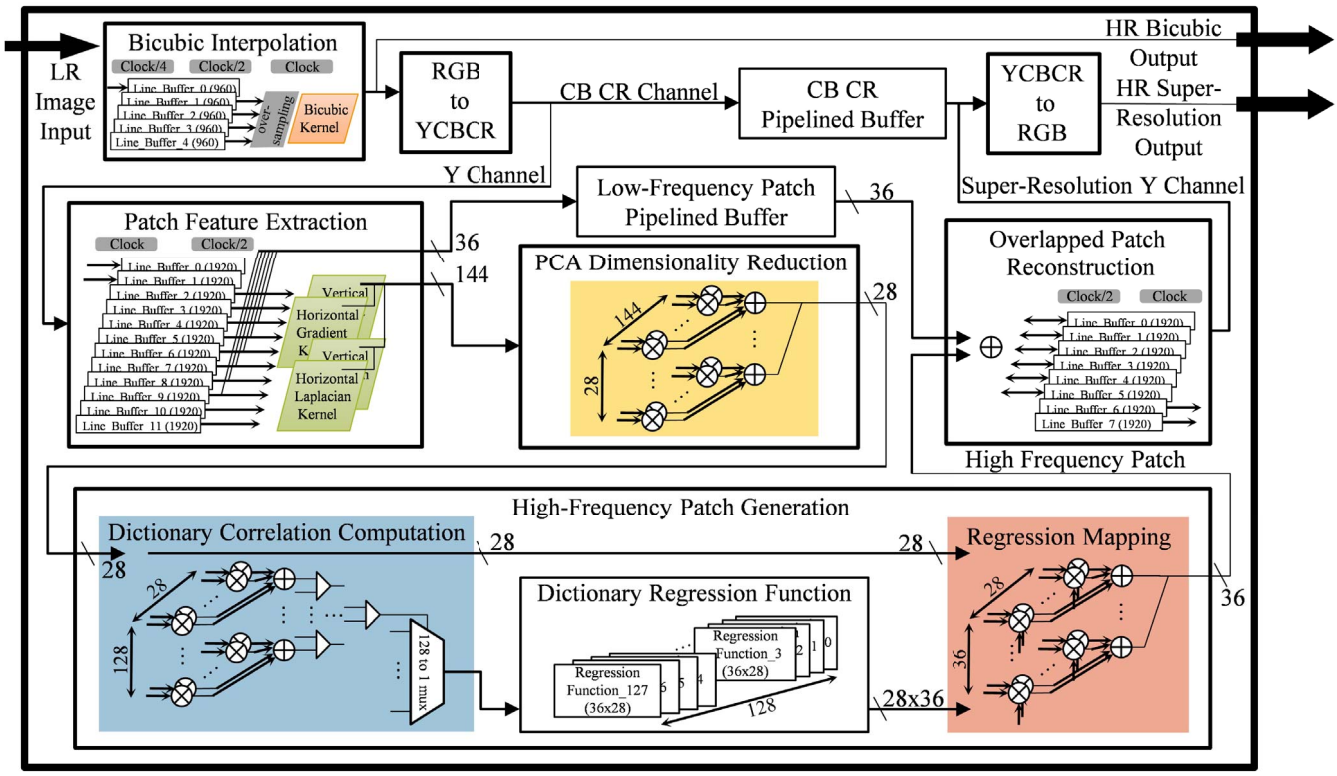

Fig. 1. Block Diagram of the Super-Resolution System.

complicated computation parts be done in offline training, leaving the online SR testing process only some comparatively simple operations, such as nearest anchor search and matrix multiplication for each input LR testing patch.

We are inspired by these developments mentioned above to make further improvement for a more hardware-friendly architecture design. Our proposed architecture is compatible with not only ANR algorithms but also those ANR-triggered follow-up algorithms such as A+ [1], SRF [9], NBSRF [10] and PSyCo [11] which can be realized by replacing certain blocks in our proposed architecture.

III. Implementation

Fig. 1 shows that there are three main stages in the proposed system. The first is a low-frequency interpolation stage, where bicubic interpolation is used for reconstructing the low-frequency parts of HR images. The second stage generates high-frequency patches by choosing the highest related pre-trained regression function according to each HR low-frequency patch. In the third stage, with the high-frequency information, the low-frequency image patches are enhanced and overlapped to construct the SR result. These operations for gaining a high-frequency result are applied to the Y-luminance channel only, while the high-resolution Cb and Cr channels are generated by bicubic interpolation.

In the second stage, which is the core of this system, those blocks handling massive computation, such as "Principal Component Analysis (PCA) Dimensionality Reduction" and "High-Frequency Patch Generation," introduce long latency for processing. Therefore, the operating frequency of the entire system is limited, further restricting the output resolution.

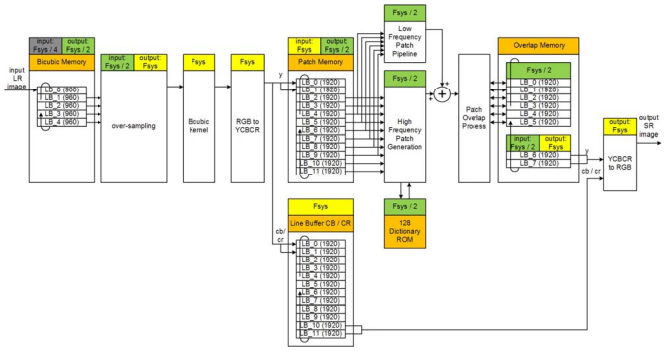

Fig. 2. Proposed Data Flow and Clock Domain Partition of Super-Resolution System.

To solve this problem, the proposed SR architecture doubles the operation period in the second stage, and the system is then designed with multiple clock domains. This is realized by making use of the available pixel time slots owing to the overlap stride while extracting features of patches.

In Section III-A, we introduce the data flow and the memory usage of the proposed super-resolution system. In Section III-B, we introduce the implementation of our FPGA verification system.

A. Data Flow and Memory Usage

Fig. 2 is the data flow of the proposed architecture and its clock domain partition. Memory not only buffers the line data but also separates the different clock domains.

Fig. 4 shows the bicubic interpolation operation. 16 integer pixels can interpolate 4 different fractional pixels by using

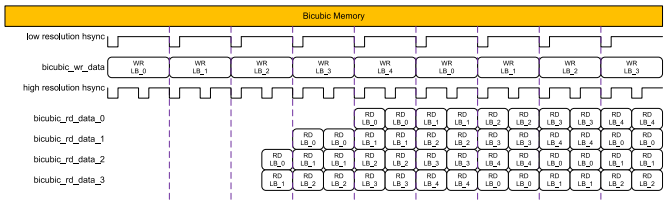

Fig. 3. Bicubic Memory Read/Write Timing.

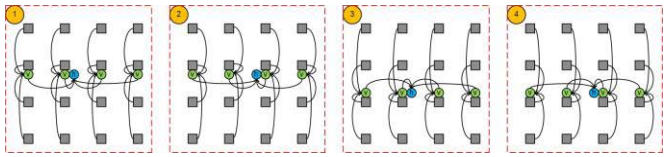

Fig. 4. Bicubic Interpolation.

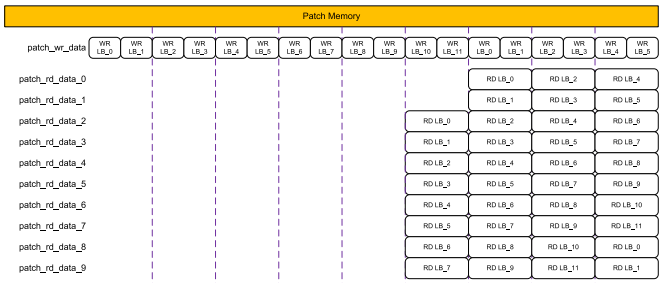

Fig. 5. Patch Memory Read/Write Timing.

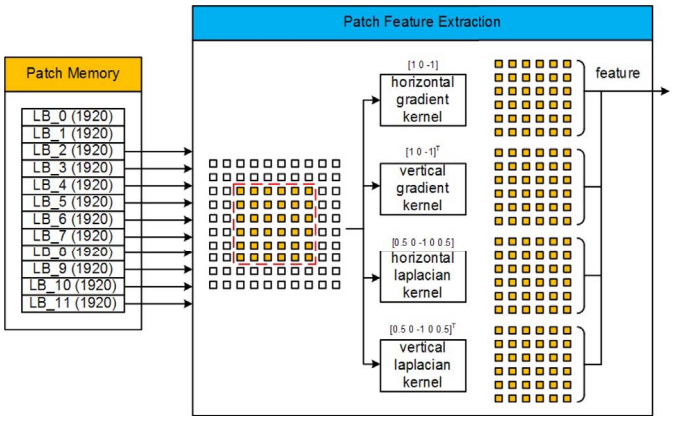

Fig. 6. Feature Extraction Operation.

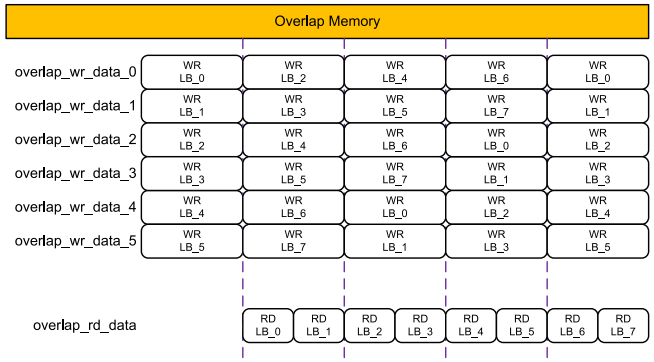

Fig. 7. Overlap Memory Read/Write Timing.

4 different bicubic interpolation weights. The first and second operation in Fig. 4 generate the first line data, and the third and fourth operations in Fig. 4 generate the second line data.

In a bicubic kernel, inputting four pixels of four consecutive lines from line buffers, we get one bicubic interpolated pixel output. Fig. 3 shows the read/write timing of bicibic memory. In bicubic memory, five line buffers are needed. While one line buffer is used for memory write by the input low resolution image, the other four line buffers are used for memory read to the bicubic kernel. In one external line period, which is indicated by low resolution Hsync signal, the bicubic memory will be read twice, which is indicated by high resolution Hsync signal. Notice that these two reading line data are the same, but the bicubic coefficients are different to interpolate two different lines, as shown in Fig. 3.

After scaling up by bicubic interpolation, pixels are converted from RGB color space to YCbCr color space. Only the Y channel will be processed by the stage-two subsystem, while the Cb Cr channel will be temporarily stored into the Cb, Cr pipeline buffer as shown in Fig. 1.

The Y channel data of bicubic interpolation data is then written into patch memory, which is composed of 12 line buffers. Fig. 5 shows the scheduling of patch memory read/write. There are two line buffers for the input Y channel. That is, we update two lines of the patch memory from the bicubic kernel output for each line scan of the input LR image.

Fig. 6 shows the architecture of feature extraction, which has four different kernels, including the horizontal gradient kernel, vertical gradient kernel, horizontal laplacian kernel, and vertical laplacian kernel. Due to the vertical laplacian kernel, which is the largest vertical kernel, we need four additional line buffers for top and bottom, and ten lines in total are required to generate 6×6 feature patches.

After the feature extraction block, the 144-dimensional (6×6×4) feature passes through the PCA dimensionality reduction block to geneate a 28-dimensional feature vector, as shown in Fig. 1.

Next, in the Dictionary Correlation Computation block shown in Fig. 1, we use this low-dimension feature vector to search the nearest anchor by computing the correlations between the input vector and those in the dicionary. Finding the index of the one with the highest correlation, we get the corresponding regressor in the form of project matrix, from the dictionary ROM, which stores 128 projection matrices. Computing the matrix multiplication of the low dimension feature and projection matrix, we get the high frequency patch in the Regression Mapping block shown in Fig. 1.

Finally, adding the low frequency patch and high frequency patch, we get the 6×6 super-resolution patch. The super-resolution patch will be written into the overlap memory in the Overlapped Patch Reconstruction block in Fig. 1. Fig. 7 shows the scheduling of overlap memory read/write. Between neighboring patches, there is a two-pixel stride in both the horizontal and vertical directions. In a patch scan line period, we will read out two lines of data. Fig. 8 shows the patch overlapping process. Each data point will be overlapped by 9 different super-resolution patches, but this does not include the boundary condition output pixel. By averging the corresponding

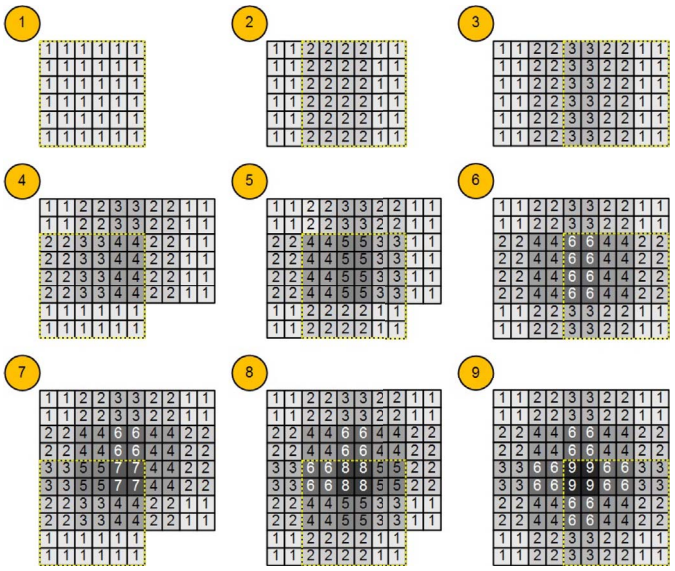

Fig. 8. Overlap Operation.

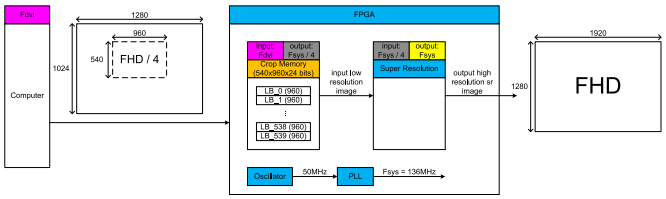

Fig. 9. Super-Resolution System on FPGA Environment.

9 pixel values from these 9 patches, the final Y-channel value of each pixel is constructed.

The Cb, Cr line buffer shown in Fig. 2 buffers the data of the Cb, Cr channel. Since there is a 12-line delay between the output of the Y channel and the output of the bicubic kernel, we need 12 line buffers to store the Cb/Cr channel data.

Finally, the Y channel data output from the overlap memory together with the Cb/Cr channel output from the Cb/Cr pipeline buffer will be converted to the RGB domain. As the output of this system, the RGB pixels form the high-resolution super-resolution image.

B. FPGA System Configuration

Fig. 9 shows the whole FPGA verification system configuration. A 1024 × 1280 resolution image is output from the computer, and inputted to the FPGA via the DVI daughter card. Since the required input resolution of the developed SR system is 540 × 960 at 60fps, only the 540 × 960 region at the center of the input image is cropped and inputted to the SR system.

Fig. 10 shows that the retiming process, which generates the 540 × 960 input frames at 60 fps as that is real applications from 1024 × 1280 60fps inputs. It takes 16.66 ms to send an image with a frame rate of 60 Hz. It takes 8.78 ms to write to cropped region to the crop memory. We then read the data from the crop memory and send the data to the super-resolution processing engine with a 60 Hz frame rate, that is, 16.66 ms for one frame.

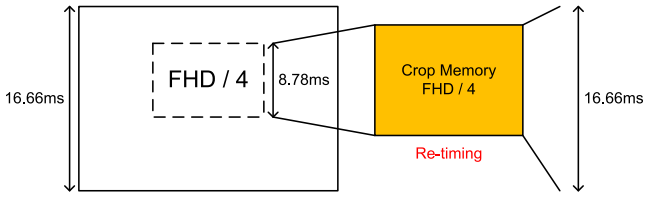

Fig. 10. Re-Timing of Crop Memory.

Fig. 11. Demonstration of the Whole Super-Resolution System.

TABLE I
ASIC IMPLEMENTATION SUMMARY OF THE SUPER-RESOLUTION SYSTEM

Module	Gate Count (K)	Memory	Size (kB)
Bicubic Interpolation	8.24	Bicubic	14.4
Patch Feature Extraction	16.16	Patch Feature	23.04
PCA Reduction	699.36	CB CR Pipelined	46.08
High-Freq. Generation	1228.62	Overlapped	19.2
Others	32.95	Regression Function	129.02
Total	1985.33	Total	231.74

TABLE II
PSNR (DB) OF THE SUPER-RESOLUTION ALGORITHM

Dataset	Software		RTL	
	Bicubic	SR	Bicubic	SR
Set5	31.76	34.00	31.76	33.83
Set14	28.39	29.97	28.39	29.77

IV. IMPLEMENTATION RESULTS

Fig. 11 is the demonstration of the whole super-resolution system. This design was evaluated and verified by an FPGA emulation board with an Altera EP4SGX530 FPGA. Our system targets a scaling factor of two, displaying images of FHD resolution. The pixel clock of this prototype system is 136 MHz, achieving a real-time SR system.

Tables I and II show the ASIC resources and performance of our implementation.

Table I shows the ASIC implementation resource of the super-resolution system. With TSMC90nm process and the target working frequency of 136 MHz, the total ASIC gate counts are about 1985K, and on-chip memory usage is about 231K Bytes.

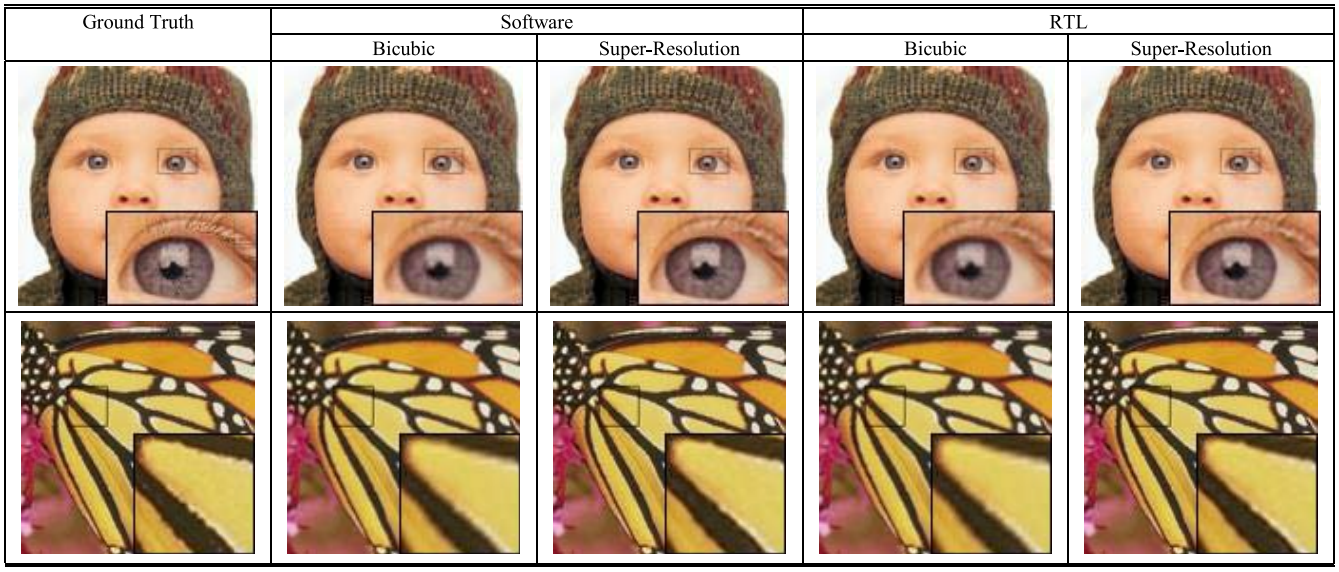

Fig. 12. 'Baby' and 'Butterfly' Images from Set5 and Set14 with an Upscaling Factor of 2.

TABLE III
COMPARISON OF DIFFERENT SR IMPLEMENTATIONS

	Method	Input	Output	Frame Rate	Throughput (pixel/s)	Hardware Realization
[12]	Multi-frame SR method	256 × 256	512 × 512	25 fps	6.5536 M	Xilinx Virtex-7 FPGA VC707 Evaluation Kit
[13]	Multi-frame SR method	160 × 120	640 × 480	150 fps	46.08 M	Xilinx Spartan-6 LX45 FPGA
This work	Single-frame SR method	540 × 960	1080 × 1920	60 fps	124.416 M	Altera DE4 Development and Education Board

Table II shows the algorithm performance in software simulation and Verilog Register-transfer level (RTL) simulation. There is no performance degradation for the bicubic interpolation. However, there is still an about 0.2 dB PSNR degradation in the whole SR process due to the fixed-point hardware architecture. The associated output images as shown in Fig. 12.

Table III shows a brief comparison of the implementations of existing super-resolution systems. In previous works, [12] and [13] proposed the hardware implementation of a multi-frame super-resolution system. However, the proposed learning-based super-resolution system architecture is the first of its kind.

V. CONCLUSION

We implement a real-time learning-based regression super-resolution system using 128 pre-trained regression functions, which is the first hardware design for such system in the literature, to the best of our knowledge. Most SR algorithms are complicated and often used only on image processing for post-production. This real-time system without a frame buffer makes it possible to integrate SR operation into image sensors or display drivers carrying out computational photography and display. Moreover, this system is applicable to all general regression SR methods by replacing the pre-trained regression functions with those obtained by other algorithms.

REFERENCES

[1] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Proc. Asian Conf. Comput. Vis.*, Singapore, 2014, pp. 111–126.

[2] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *IEEE Comput. Graph. Appl.*, vol. 22, no. 2, pp. 56–65, Mar./Apr. 2002.

[3] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Washington, DC, USA, 2004, p. 1.

[4] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[5] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, Nov. 2010.

[6] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proc. Int. Conf. Curves Surfaces*, Avignon, France, 2010, pp. 711–730.

[7] A. Michal, M. Elad, and A. Bruckstein, "The K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[8] R. Timofte, V. De Smet, and L. Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Sydney, NSW, Australia, 2013, pp. 1920–1927.

[9] S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, 2015, pp. 3791–3799.

[10] J. Salvador and E. Pérez-Pellitero, "Naive Bayes super-resolution forest," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, 2015, pp. 325–333.

[11] E. Pérez-Pellitero, J. Salvador, J. Ruiz-Hidalgo, and B. Rosenhahn, "PSyCo: Manifold span reduction for super resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 1837–1845.

[12] K. Seyid, S. Blanc, and Y. Leblebici, "Hardware implementation of real-time multiple frame super-resolution," in *Proc. IFIP/IEEE Int. Conf. Very Large Scale Integr. (VLSI SoC)*, Daejeon, South Korea, 2015, pp. 219–224.

[13] R. Redlich, L. Araneda, A. Saavedra, and M. Figueroa, "An embedded hardware architecture for real-time super-resolution in infrared cameras," in *Proc. Euromicro Conf. Digit. Syst. Design (DSD)*, Limassol, Cyprus, 2016, pp. 184–191.