A Real-time Super-resolution Accelerator Using a big.LITTLE Core Architecture

Xuan Truong Nguyen, Tuan Nghia Nguyen, Hyuk-Jae Lee
Department of Electrical and Computer Engineering
Seoul National University
Seoul, Korea
{truongnx, nghiant, hyuk_jae_lee}@capp.snu.ac.kr

Kyujoong Lee
School of AI Convergence
Sungshin Women's University
Seoul, Korea
kyujoonglee@sungshin.ac.kr

***Abstract*— Image super-resolution (SR) networks have shown a remarkable restoration performance but come with a huge memory bandwidth requirement due to a large-size input and lack of a pooling layer. As a result, SR accelerators usually adopted a streaming-like scheme to fit a highly customized and small SR network to available resources, which causes a large accuracy drop. To address this problem, this work proposes an SR accelerator using a big.LITTLE core architecture which is able to execute various networks in real-time. The proposed SR processor achieves an inference speed of 36.63 frames per second and a throughput of 221.79 GOPs at 200MHz for ×2 SR (from 960×540 to 1920×1080) while using only 1,280 eight-bit multipliers and 330 KB on-chip SRAM.**

Keywords—Image super-resolution, SR accelerator, SRNPU

I. INTRODUCTION

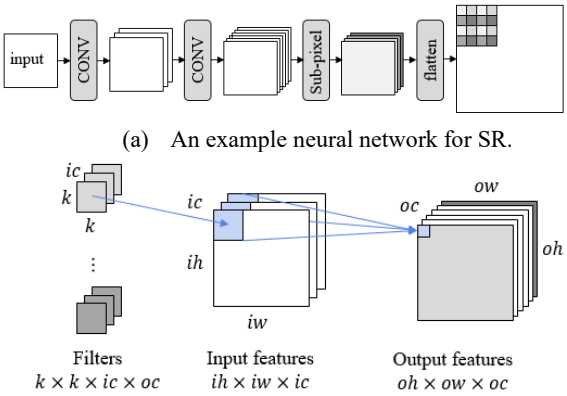
Image super-resolution (SR) is a method to reconstruct a high-resolution (HR) image from the input low-resolution (LR) one. The rapid evolution of convolutional neural networks (CNNs) has led to substantial performance improvements in SR [1]-[3]. As the result, this property has provided CNN-powered SR to build a lot of applications on mobile and home devices such as smartphones, photograph frames, and TVs. Therefore, a dedicated upscaling technique becomes essential, especially when it comes to video upscaling for an output of 2K full high definition (FHD).

Despite their unparalleled performances, the state-of-the-art SR networks (SR-CNNs) [1]-[3] own distinctive features compared to classification networks (C-CNN) [4]-[6], which results in deployment challenges for real-time applications. First, SR-CNN's input image resolution is much larger than C-CNN (×10.33(=960×540/(224×224) times larger than VGG-16 [4] in the case of ×2 scale Full-HD SR). Second, SR-CNN excludes pooling layers (Fig. 1a), therefore its feature maps remain as large as its input image through the layers. As the result, it causes both computation and memory bandwidth bottlenecks which lead to a high on-chip memory requirement and underutilization in the existing works. To address those problems, this study presents a novel SR neural processing unit (SRNPU), making three main contributions, as follows:

(1) *Design*: We propose a big.LITTLE core architecture for SR which includes both big cores and small cores to effectively utilizes computing resources for various SR networks.

(2) *Dataflow:* We propose line-based caching dataflow to fuse operations of convolutional layers, which largely reduces off-chip memory accesses with relatively small buffer size.

(3) *Implementation*: The proposed design is implemented with Verilog HDL. The simulation shows that the proposed SR accelerator with a line-based caching dataflow achieves a speed of 33.06 frames per second at 200MHz for ×2 SR for an output of 2K FHD while only using 1,280 eight-bit multipliers and 330 KB on-chip SRAM. The design can fit on a low-cost Nexys Video Artix-7 FPGA board [16] which has only 740 DSPs.



(a) An example neural network for SR.

(b) CONV layer.

Fig. 1: Example neural network and CONV layer.

II. RELATED WORK

A. SR Networks

Many lightweight SR networks have been proposed [1]-[3]. SRCNN [1] is the first CNN-based SR network that does an end-to-end mapping between an LR image and its HR image. An LR image is preprocessed by bicubic interpolation followed by three convolutional (CONV) layers. To speed up SRCNN, [2] proposed FSRCNN, a compact hourglass-shape CNN design, with three modifications. First, the mapping is directly learned from the LR image (without interpolation) to the HR one. Second, it shrank input feature dimensions before mapping and expanded back later. Third, it adopts smaller filter sizes but more mapping layers. ESPCN [3] is another approach that improves the speed of SRCNN by learning an end-to-end mapping from an LR image (without interpolation) to the HR image. ESPCN introduced a sub-pixel layer that has s^2 output features for a given scaling factor s. The sub-pixel filters effectively replace the hand-crafted interpolation filter, while reducing the computational complexity. A typical network is shown in Fig. 1. Many deep networks [7]-[9] are proposed to boost the restoration quality by stacking multiple layers or residual blocks. However, it is impossible to execute them in real-time at resource-constrained devices. As the result, FSRCNN [2] and its variants are widely used as the baselines for SR accelerators at edge-level devices.

B. SR Accelerators

Streaming accelerators [10]-[13]: Streaming-like designs are the most straightforward solutions for the hardware

implementation of SR networks. In particular, a streaming accelerator makes a strong assumption that a target DNN is small enough to fit on an available hardware resource (i.e., FPGA DSP or block RAM). Obviously, this dataflow style is not applicable for medium or large networks in SR. If the scaling factor, however, is large, it is challenging for a tiny and highly-customized SR network to achieve a reasonable restoration performance.

Fixed dataflow accelerators [14]: A fixed dataflow accelerator executes an SR network layer by layer. To start a neural network execution, it firstly loads weight values from the external memory to the unified buffers. After feeding the weights to a MAC array, the accelerator performs the layer's convolutional, batch-normalization and activation operations. Lastly, the core passes the output values to the buffers which are stored into off-chip memory. The procedure is repeated until the last sub-pixel layer is completed. However, it is challenging to design a unified NPU for multi-DNN SR workloads because it suffers from both large external memory bandwidth and low computation utilization.

III. Proposed SR Accelerator

A. SR Networks

Inspired by [2], we develop three light-weight SR models m1, m2, and m3 that have eight CONV layers and similar computing requirements, i.e. 6.05, 5.68, and 6.58 GOPs, as shown in Table I. Different from [2], they adopt a sub-pixel layer [3] as the last layer to generate an HR image. Models m1 and m2 are designed for a scaling factor of 2, while m3 is used for a scaling factor of 4. Fig. 2 illustrates a visual example.

TABLE I: SR networks having similar computation. ni and no are the numbers of input and output channels, respectively.

Layer	filter	m1		m2		m3	
		ni	no	ni	no	ni	no
1	3×3	1	32	1	64	1	64
2	1×1	32	16	64	32	64	32
3	3×3	16	16	32	32	32	32
4	3×3	16	16	32	32	32	32
5	3×3	16	16	32	32	32	32
6	3×3	16	16	32	32	32	32
7	1×1	16	32	32	64	32	64
8	3×3	32	4	64	4	64	16
Parameters		11680		43840		50752	
Scaling factors		2		2		4	
Input size		540×960		270×480		270×480	
Output size		1080×1920		540×960		1080×1920	
GOPs		6.05		5.68		6.58	

B. Quantization

In SRNPU, both weights and activations are quantized to 8-bit values, while biases and scales are quantized to 16-bit ones [15]. A uniform quantization scheme is adopted.

Output-channel-wise weight quantization: For quantizing weights, a channel-wise quantization method is applied. First, each weight output tensor is rescaled in the range [-1, 1] by using a quantization scale. Then, each value in [-1, 1] is mapped to a value in the range $[2^{-n}, 1-2^{-n}]$, where n is the bit-width. Here, n is set to 8. In particular, a quantized value is calculated by multiplying two components, one of which is scale factor m equals to the maximum absolute value of all elements x_i in the kernel. Another is the quantized value of the quotient resulting from the division between x_i by m.

$$\mathrm{Q}(x_i) = \mathrm{sgn}(x_i).m.CLIP\left(g\left(\frac{x_i}{m}, n\right), 2^{-n}, 1-2^{-n}\right) \quad (1)$$

$$m = \max(|x|) \quad (2)$$

$$\mathrm{g}(t,n) = (\mathrm{round}(t.2^{n-1}+0.5)-0.5).2^{1-n} \quad (3)$$

$$CLIP(t,l,u) = \max(\min(t,u),l) \quad (4)$$

Activation Quantization: To preserve statistical distribution of activations, the proposed SRNPU utilizes a dynamic fixed-point representation. For each layer, the number of fractional bits is adjusted via a quantization step that satisfies (5), where x and n are an activation map and the bit-width, respectively.

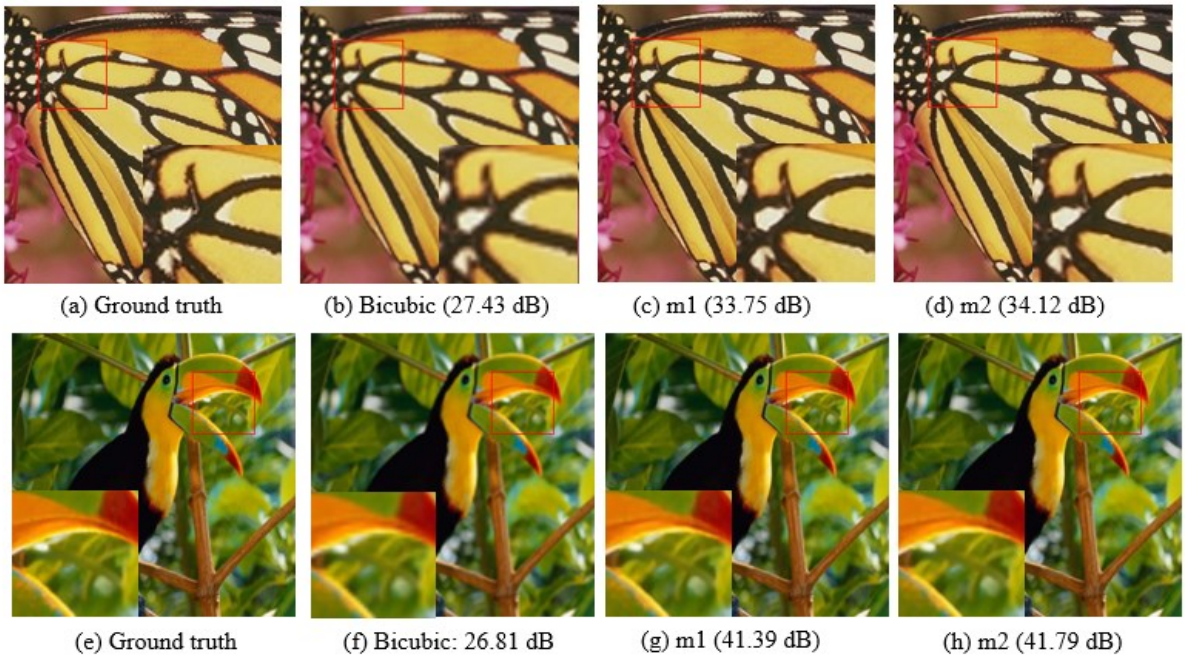
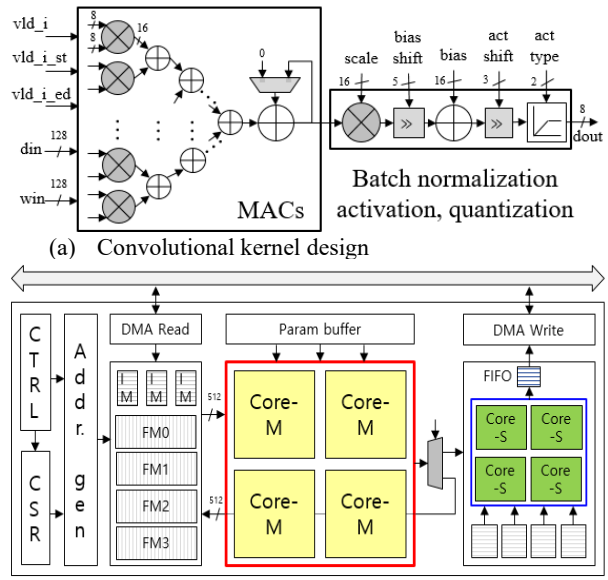


Fig. 2: Restoration results (a),(e): Ground truth, (b), (f): Bicubic results, (c), (g): results given by m1; and (d), (h): results given by m2.



(a) Convolutional kernel design

(b) Overall big.LITTLE SR accelerator

Fig. 3. Proposed SR accelerator.

The quantization step is equal to the nearest power-of-two value of the activation range over the number of levels.

$$\text{round}\left(\frac{\max(x)-\min(x)}{2^n-1}\times\frac{1}{s}\right)=1 \qquad (5)$$

C. *Proposed SR Accelerator based on big.LITTLE cores*

Fig. 3 shows the proposed SR accelerator design using a big.LITTLE core architecture.

Convolutional kernel primitive: Fig. 3a shows a design of a convolutional kernel that can calculate an output pixel. It takes a tensor of sixteen eight-bit pixels (e.g., 1×1×16) and sixteen sets of filters (e.g., 1×1×16×16) as the inputs and performs multiplication and accumulation operations. If the number of required operations (e.g., 32 and 64 multipliers for the second CONV layers in m1 and m2, respectively), an accumulated register is used to gather a partial sum in multiple cycles. The accumulated result is the next module that performs batch normalization and activation followed by quantization to output an eight-bit pixel.

Proposed big.LITTLE core SR accelerator: Fig. 3 shows the proposed SRNPU design which consists of four medium-size cores (core-M), and four small cores (core-S). Having 256 eight-bit multipliers, each medium core includes sixteen CONV kernel primitives which calculate an output tensor of sixteen pixels (e.g., 1×1×16) in parallel. Meanwhile, a small core consists of four CONV kernels which compute a four-pixel tensor (e.g., 1×1×4) at a time. It is noted that each core-S is dedicated to a sub-pixel layer that is widely adopted for modern SR networks. In addition, a sub-pixel layer usually has a small number of output channels compared to others. As a result, using a little core architecture for a sub-pixel layer avoids computing power underutilization. Furthermore, four cores are used to enhance throughput since they calculate four consecutive pixels in parallel. To this end, our SR accelerator consists of 1,280 eight-bit (=4×256+4×64) multipliers. As a result, it can fit on a low-cost Nexys Video Artix-7 FPGA board [16] which only has 740 DSPs because two eight-bit multipliers can be mapped to a single DSP.

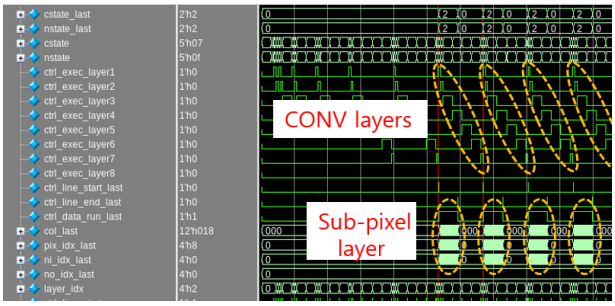


Fig. 4. Line-based pipelining dataflow.

TABLE II: Line buffer requirements for m1.

Layer	fx	fy	ni	no	# params	line buffer
1	3	3	1	32	288	3
2	1	1	32	16	512	32
3	3	3	16	16	2,304	48
4	3	3	16	16	2,304	48
5	3	3	16	16	2,304	48
6	3	3	16	16	2,304	48
7	1	1	16	32	512	16
8	3	3	32	4	1,152	96

TABLE III: On-chip buffers

	Word	Depth	Type	#	Remark
IM	32	240	Dual	3	Input
FM	128	3600	Dual	4	FM
SFM	128	1440	Single	4	subpixel
OM	32	960	Dual	1	Output

Line-based caching dataflow: Different from existing streaming-like accelerators, our SRNPU performs line-based caching dataflow for layer fusion. In particular, four core-M are assigned to one layer at a time in an as-soon-as-possible scheduling manner since they are shared among different layers. For example, when the computing units are assigned for the calculation of Layer 1, the input data comes from IM, and the output is stored FMs. Since the second layer is a CONV 1x1 layer, only a line image is required to do the calculation. Therefore, the computing units are assigned to Layer 2 when Layer 1 completes computation for a line. The key advantage of the proposed dataflow is to avoid the workload imbalance among different layers which occurs in conventional streaming-like SR accelerators, and therefore often causes computation underutilization. Fig. 4 illustrates the timing scheduling to assign the computing cores (core-M) for a layer at a time. In addition, the computations of CONV layers and a sub-pixel layer are pipelined for throughput. In particular, a core-S is used to produce the final output of a sub-pixel layer as long as it receives results computed by a core-M.

Buffer requirements: Similar to other accelerators, our accelerator consists of multiple buffers for filters or weights (WGT), the input image (IM), the input/output feature maps (FM), and the final output (OM). Table II reports the buffer requirements for model m1. Particularly the buffer WGT has a size of 11.4 KB to store 11680 parameters of m1. Moreover, thanks to the line-based caching dataflow, SRNPU allocates the minimum number of line buffers required for each layer. For example, it uses 32 lines to store 32 intermediate results

between Layers 1 and 2. In total, 319 lines are required. Those line buffers are mapped to on-chip buffers shown in Table III. Instead of using dedicated buffers for each independent layer, our SRNPU gathers on-chip buffers from multiple layers, which can further save the number of wires to access buffers.

For example, when SRNPU comes to video upscaling for an output of 2K full high definition (FHD), the input size is 960×540. As a result, a row-based caching dataflow results in a line buffer requirement of 317.8 KB (=319×960×8 bits) or approximately 330 KB in total. It is noted that the dataflow is well fit on a general commercial display panel which usually displays a frame line by line. Furthermore, if the display order is relaxed, a column-based caching data flow could be used, resulting in a line buffer requirement of 172.3 KB (=319×540×8 bits) or 183.7 KB buffers in total.

IV. Implementation and Discussion

The proposed design is implemented in Verilog HDL and mapped to a low-cost Nexys Video Artix-7 FPGA board [16] which only has 740 DSPs. The performances of the proposed design are reported in Table IV. In addition, we compare our design with the existing SRNPU [14] since the two designs target general SR networks. Thanks to the proposed line-based caching dataflow, our SRNPU achieves a real-time speed (e.g., 36.63 frames per second (fps)) for video upscaling for an output of 2K full high definition (FHD). To this end, it achieves a throughput of 221.79 GOPs while using 1,280 MACs and 330 KB on-chip buffers.

Table IV: Performance evaluation and comparison

	[14]	Ours	Diff.
SRNPU	√	√	-
Number MACs	1,152	1,280	-
Frame size	960×540	960×540	-
Frequency (MHz)	200	200	-
Frame rate (fps)	31.8	36.63	-
Buffer (KB)	572	330	↓42.31%
Throughput (GOP/s)	158.43	221.79	↑39.99%
Throughput per MAC	0.1375	0.1733	↑26.03%
PSNR (×2 Set14) (dB)	32.62	32.80	+0.18

V. Conclusion

This work presents a novel design of an SR accelerator that can do video upscaling for an output of 2K full high definition. Compared to [14], the proposed design reduces the buffer by 42,31%, improves the throughput by 39,99%, and enhances the throughput efficiency by 26.03%. Thanks to its high throughput, our SRNPU is able to execute a large SR network, which improves the restoration performance (e.g., Peak signal to noise ratio (PSNR)) by 0.18 dB on the dataset Set14 [12]. To this end, the design fully utilizes the available DSPs on a low-cost Nexys Video Artix-7 FPGA board, so that it can be used for a vast of energy-efficient applications including display panels, CIS sensor enhancement, or Internet-of-Thing.

Acknowledgment

This work was supported in part by the R&D Program of MOTIE/KEIT (No. 20010582, Development of deep learning based low power HW IP design technology for image processing of CMOS image sensors) and in part by the Technology Innovation Program (No. 20011074, Development of Open Convergence Memory Solution and Platform for Next Generation Memories) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea).

References

[1] C. Dong, C. C. Loy, K. He, X. Tang, "Learning a Deep Convolutional Network for Image Super-Resolution," in *Proceedings of European Conference on Computer Vision* (*ECCV*), 2014.

[2] C. Dong, C. C. Loy, X. Tang, "Accelerating the Super-Resolution Convolutional Neural Network," in *Proceedings of European Conference on Computer Vision* (*ECCV*), 2016.

[3] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), pages 1874–1883, 2016.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems*, vol. 1, 2012, pp. 1097–1105.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556v6, 2014.

[6] Y.Lee, C. Lee, J. Kim, and H.-J. Lee, "Fast Detection of Objects Using a YOLOv3 Network for a Vending Machine," *IEEE International Conference on Artificial Intelligence Circuits and Systems* (AICAS), Mar. 2019.

[7] Kim, J. K. Lee. and K. M. Lee, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks," in *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2016.

[8] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2017.

[9] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," In *IEEE Conference on Computer Vision and Pattern Recognition Workshop,* 2017.

[10] J.-W. Chang, K.-W. Kang, and S.-J. Kang, "An Energy-Efficient FPGA-Based Deconvolutional Neural Networks Accelerator for Single Image Super-Resolution", *IEEE Trans. On Circuits and Systems for Video Technology*, vol. 30, no.1, Jan. 2020.

[11] D. Lee, S. Lee, H. Lee, K. Lee, and H.-J. Lee, "Context-Preserving Filter Reorganization for VDSR-Based Super-resolution," *IEEE International Conference on Artificial Intelligence Circuits and Systems* (AICAS), Mar. 2019.

[12] D. Lee, H. Lee, S. Lee, K. Lee and H.-J. Lee, " Hardware Design of a Context-Preserving Filter-Reorganized CNN for Super-Resolution", *IEEE Journals on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 4, Dec. 2019.

[13] Y. Kim, J.-S. Choi, and M. Kim, "A Real-time Convolutional Neural Network for Super-Resolution on FPGA with Applications to 4K UHD 60 fps Video Services," *IEEE Trans. On Circuits and Systems for Video Technology*, vol. 29, no.8, Aug. 2019.

[14] J. Lee; J. Lee; and H.-J. Yoo, "SRNPU: An Energy-Efficient CNN-Based Super-Resolution Processor With Tile-Based Selective Super-Resolution in Mobile Devices," *IEEE Journals on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, vol. 10, no. 3, Sep. 2020.

[15] T. N. Nguyen, X. T. Nguyen, K. Lee, H.-J. Lee, "Towards Low-bitwidth Quantization of Convolutional Neural Networks for Single Image Super-Resolution," *2020 Annual IEIE Conference*, Nov. 2020.

[16] https://digilent.com/reference/programmable-logic/nexys-video/start

[17] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Procceding 7th International Conference on Curves Surface*, Jun. 2010, pp. 711–730.