International Core Journal of Engineering Volume 7 Issue 10, 2021
ISSN: 2414-1895 DOI: 10.6919/ICJE.202110_7(10).0047

Implementation of Hardware Acceleration of CNN-based Single Image Defogging Model

Guanjun Wang[1], Qiang Xiang[1,*]

[1]College of Electronic and Information, Southwest Minzu University, Chengdu 610041, China.

*xiangqiang@swun.edu.cn

Abstract

Aiming at the problem that the CNN-based single-image defogging model is not easy to deploy on the terminal conveniently, the article quoted the TFLite algorithm to realize the quantification of defogging model and used high level synthesis tools and C++ language to reconstruct and output Hardware IP core with the AXI4 bus. Then, the article verified the IP core via the Zynq7020 board. The results show that the hardware IP, under the premise of defogging performance, processed a hazy image with the resolution of 640*480 pixels only in 0.14s, which is nearly 5 times faster than inference on a CPU, and the total on-chip of power consumption is only 1.89W.

Keywords

CNN; Defogging Model; IP Core; ZYNQ; High Level Synthesis (HLS); Model Reconstruction.

1. Introduction

In recent years, the defogging algorithms have been developed rapidly. Retinex, dark channel prior [1] and CNN-based defogging algorithms [2-6] are widely used. These defogging algorithms can be divided into three categories [7-8]. In particular, CNN-based single image defogging has been unprecedentedly developed, especially the end-to-end defogging network [2, 5, 6], which was used as the pre-processing of intelligence, such as Fast-RCNN, etc. However, this type of defogging model pays more attention to model design and less researches on deployment on the embedded system, especially on all programmable system-on-chip(APSoC).

Therefore, the article adopted the Zynq7020-based hardware platform and Xilinx development kit, and used the TFLite quantification algorithm [9, 10] and C++ language, and then generated the IP core of AOD-Net [6] in the high-level synthesis tool(HLS) [11, 12]. Subsequently, we used VDMA-based storage architecture [13] to build hardware system and test the IP core. The results show that under the premise of dehazing performance, it is possible to achieve inference acceleration of single image.

2. Methodology

2.1 Quantization method of weight parameters

In order to reduce the amount of calculation and the consumption of hardware logic resources, this article quoted the quantification algorithm of TFLite [9, 10]. This algorithm is a post-training static quantization method and does not need to be retrained after quantization. Its purpose is to implement only integer and shift operations, which is exactly what FPGA needs. The conversion formula between the floating point and the integer type is:

$$r = S(q - Z) \tag{1}$$

$$q = round(r/S + Z) \tag{2}$$

where r is floating point, q is a quantized integer. S is a scale factor, which represents the proportional relationship between a real number and an integer. Z is a zero point, which represents the integer value corresponding to zero in the real number after quantization. The calculation formulas of S and Z are:

$$S = (r_{max} - r_{min})/(q_{max} - q_{min}) \tag{3}$$

$$Z = round(q_{\max} - r_{\max} / S) \tag{4}$$

Therefore, the quantification process in this article can be expressed as follows:

Input: The model parameters with floating point after pre-training.

Output: The model parameters after quantization; the model parameters after inverse quantization.

a) For the trained model, counting r_{min} , r_{max} of each layer and determine the number of quantization bits.

b) Using Eqs. (3) (4) to calculate S and Z of each layer.

c) Using Eq. (2) to perform quantification and saving all model parameters after quantification to the file.

d) Using Eq. (1) to perform dequantization to simulate the effect of quantization on the dehazing effect and determine the minimum number of quantization bits, and then save it as a file.

2.2 Model hardware reconstruction and generating the IP core

In general, the convolution operation is the most time-consuming in the model. Therefore, in order to realize the IP coreization of the defogging model AOD-Net [6], this article quoted two synthesizable data structures of the HLS video library [11]: window buffer (hls:: Window) and line buffer (hls:: LineBuffer). Then, we used these data structures to implement 2D convolution processing, as illustrated in Fig. 1.

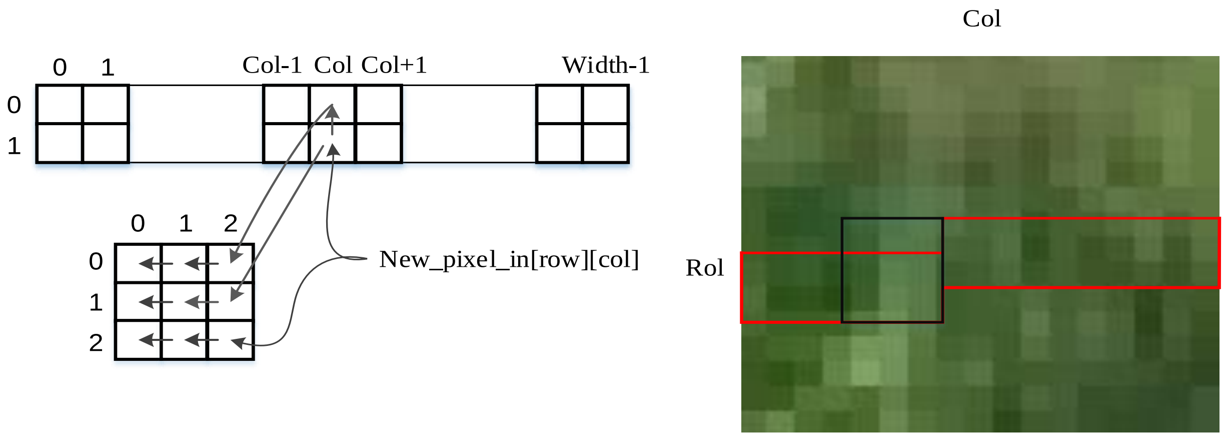

Fig. 1 Diagram of window buffer and line to achieve 2D convolution

Where the pixels in the black box are stored in the window buffer, and the pixels in the red box are stored in the line buffer. In addition, the port of the IP core is constrained into an AXI4-Stream interface through constraint directive. Particularly, we directly outputed original pixels for 2D-convolution boundary pixels to reduce the complexity of hardware implementation and save hardware resources. Finally, we used advanced synthesis tools to convert the reconstructed code into a hardware IP core.

2.3 System design for testing the hardware IP

In order to verify the time-consuming and effectiveness of the generated hardware IP core, we built a test system based on the vivado 2018.3 software platform, zynq7020 processor and VDMA-based video stream memory architecture [13]. The system block diagram is shown in the Fig. 2.

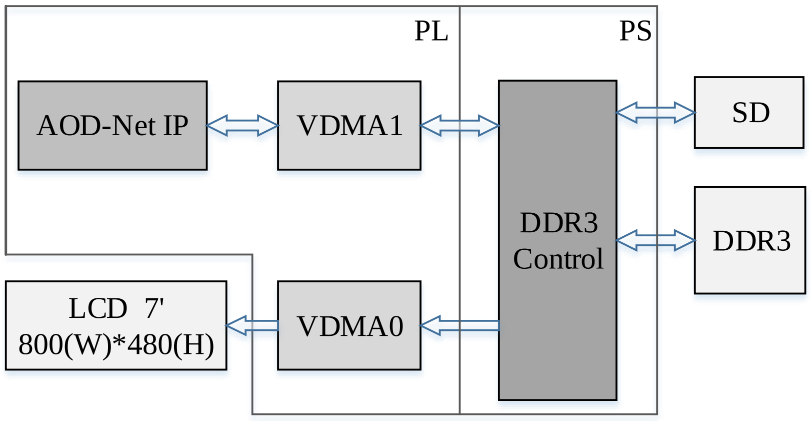

Fig. 2 The system block diagram for testing the hardware IP

In this block diagram, we used the SD storage device to read the original hazy images, and display the defogging images on the LCD screen and write back to the SD storage device. In addition, we checked the function XTime_GetTime() in the SDK to count the average time of defogging single hazy image with the resolution of 640*480 pixels.

3. Results and discussion

3.1 Quantitative experiment and result analysis

In order to objectively evaluate the dehazing performance, we selected the total of five evaluation indicators, such as mean squared error (MSE), peak signal to noise ratio (PSNR), structural similarity (SSIM), image mean, etc. The experimental results adopted the method of averaging the defogging results of 50 random hazy images, and obtained the statistical data shown in Tab. 1. Note that all indicators are compared with floating point dehazing.

Tab. 1 Evaluation of the defogging effects of different quantization bits

AOD-Net	Mean	Std	MSE	PSNR	SSIM
float32	103.72	59.28	—	—	—
8bit	-0.02	0	0.32	27.81	0.99
7bit	-0.02	+0.11	0.55	26.13	0.99
6bit	-1.88	+0.58	2.46	22.36	0.99
5bit	+1.25	-0.76	2.06	23.07	0.98
4bit	**+8.85**	**-3.65**	**35.86**	16.53	0.94
3bit	**-13.96**	**+3.22**	81.13	14.72	0.88
2bit	-33.67	+5.43	483.89	11.00	0.56

As illustrated in Tab.1, when the quantization bits is above 4bit, the magnitude of the data change is comparatively small; when the quantization bits less than 5bit, the data fluctuates significantly. After analysis, the mainly reason is the lower number of quantization bits limits the range of values that can be represented, which is very difference in the distribution of the original floating-point parameters, so as to affects the final inference result [14]. Therefore, we chosen 5bit to quantify the AOD-Net model, and then reconstructed the quantified model to output the hardware-accelerated IP core. Even quantized to 5bit, it also can save 84.4% of storage resources for model parameters.

3.2 The results and analysis of generating and testing the IP

We selected Zynq7020 as the target processor in HLS, and set the initial clock to 20ns, and perform synthesis and implementation. The results are shown in Tab. 2.

Tab. 2 Performance and estimated consumption of IP after synthesis

Name	Clock	BRAM	DSP48E	FF	LUT
Total	13.5ns	115	57	21182	39006
Availalble	—	280	220	106400	53200
Utilization(%)	—	41	25	19	73

As illustrated in Tab.2, the minimum pixel clock cycle of the generated hardware IP core is 13.5ns. Then, if the pixel clock cycle of the PL part is 20ns, the theoretical time for processing a hazy image with the resolution of 640*480 pixels should be 0.12 seconds. In addition, we built the hardware system in vivado according to the system design for testing the hardware IP in Figure 2, and the results are shown in Fig. 3.

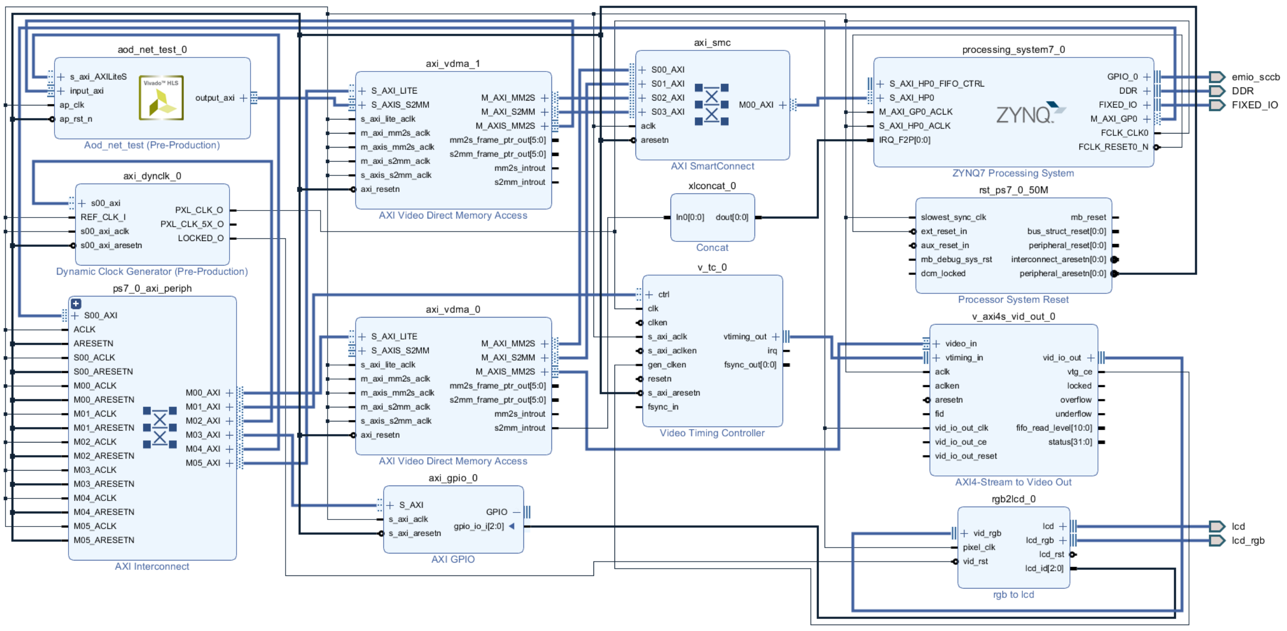

Fig. 3 The hardware system for testing the IP

Synthesize and implement the above systems, the results are shown in Tab. 3.

Tab. 3 Synthesis and implementation result of hardware system

Name	BRAM	DSP48E	FF	LUT	Power(W)
Total	118	125	35112	31388	1.89
Availalble	280	220	106400	53200	—
Utilizaton(%)	42	27	33	59	—

After completing the configuration of VDMA, SD and the IP core of AOD-Net module in the SDK, we started the system to defogging inference. Part of the results are shown in Fig. 4, the experimental results show that there is no obvious difference between the defogging effect based on hardware IP core and pytorch based on floating-point when except for three-circle pixels of the border. As for the reason, it is mainly caused by not dealing with the 2D convolution boundary when reconstructing the model.

In addition, we checked the function XTime_GetTime() in the SDK to count the average time of defogging single frame hazy image with the resolution of 640*480 pixels, and the other contrast defogging models were run on the CPU based on pytorch. The results are shown in Tab. 4.

Tab. 4 Average time taken by various methods to process single image (in seconds)

AOD-Net [6]	FAOD-Net [2]	FAMED-Net [3]	GCANet [4]	Ours
0.65	0.34	0.89	0.23	**0.14**

The actual processing time in this paper is 0.14s, which is more than the theoretical value of 0.12s. This extra time comes from the moving of pixel stream data, but it is also better than the inference time of the same type of defogging model on a CPU.

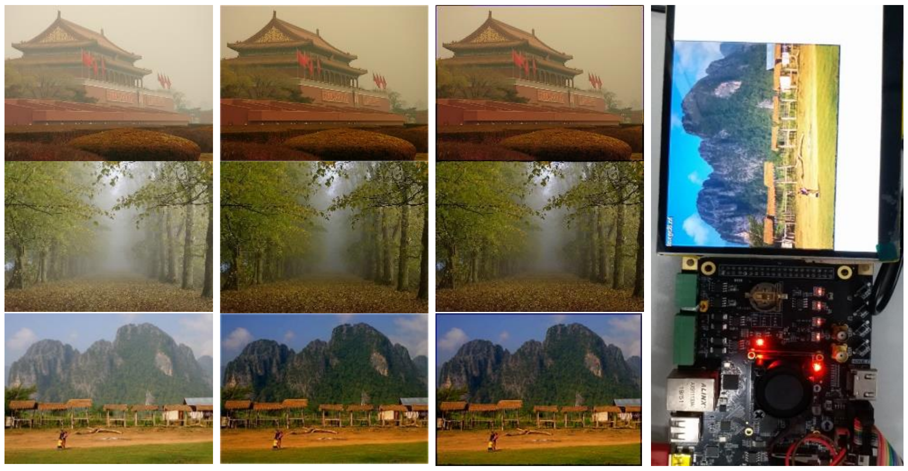
(a) (b) (c) (d)

Fig. 4 Comparison of effects: (a) Fog images.(b) Pytorch defogging based on CPU. (c) Defogging based on hardware IP core.(d) Screenshot of the effect on the 7020 board.

4. Conclusion

In this paper, we used the TFLite quantization algorithm and the hardware reconstruction of the end-to-end and light-weight defogging model, and the acceleration of single-image defogging was realized and deployed on the APSoC platform. It creates conditions for expanding the application scope of this type of defogging model, and also provides a reference for the terminal deployment of other models. Next, the authors will study how to achieve defogging from single image to real-time video defogging and a quantitative method that saves more hardware resources.

Acknowledgments

This work was financially supported by the innovative Research Project for Master Students of Southwest Minzu University (CX2021SZ46).

References

[1] K. He, J. Sun and X. Tang (2010), Single image haze removel using dark channel prior[J]. IEEE transactions on pettern analysis and machine intelligence, vol.33, no.12, p.2341-2353.

[2] W. Qian, C. Zhou, D. Zhang (2020). FAOD-Net: A fast AOD-Net for dehazing single image [J]. Mathematical Problems in Engineering, vol. 2020, no.02, p.1-11.

[3] J. Zhang, D. Tao (2019). FAMED-Net: A Fast and Accurate Multi-scale End-to-end Dehazing Network[J]. IEEE Transactions on Image Processing, vol.29, no.7, p.72-84.

[4] D. Chen, M. He and Q. Fan, et al (2019). Gated context aggregation network for image dehazing and derain-ing [C]// 2019 IEEE winter conference on Applications of computer vi-sion(WACV). IEEE, p.1375-1383.

[5] Y. Liu, G. Zhao. (2018). Pad-net: a perception-aided single image dehazing network. arXiv preprint arXiv: 1805. 03146, 2018.

[6] B. Li, X. Peng and Z. Wang, et al (2017). An all-in-one network for dehazing and beyond[J]. arXiv preprint arXiv:1707.06543, 2017.

[7] A. S. Parihar, Y. K. Gupta, Y. Singodia, V. Singh and K. Singh (2020). A Comparative Study of Image Dehazing Algorithms," 2020 5th International Conference on Communication and Electronics Systems (ICCES), p. 766-771.

[8] D.L. Wang, T.Y. Zhang (2020). Review and analysis of image defogging algorithm. Journal of Graphics [J], vol.41, no.06, p.861-870.(in Chinese)

[9] R. Krishnamoorthi. (2018). Quantizing deep convolutional networks for effic-ient inference: a whitepaper [J]. arXiv preprint arXiv: 1806.08342, 2018.

[10] B. Jacob, S. Kligys and B. Chen, et al (2017). Quantization and training of neural networks for efficient integer-arithmetic-only inference [J]. arXiv preprint arXiv: 1712.05877, 2017.

[11] Xilinx Inc. Vivado Design Suite User Guide:High-Level Synthesis[EB/OL]. (2021-05-04)[2021-06-01]. https://china.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug902-vivado-high-level-synthesis.pdf .

[12] Xilinx Inc. Vivado Design Suite Tutorial:High-Level Synthesis [EB/OL]. (2020-08-07)[2021-03-23]. https://china.xilinx.com/sup port/documentation/sw_manuals/xilinx2021_1/ug871-vivado-high- level -synthesis-tutorial.pdf.

[13] Xilinx Inc. AXI Video Direct Memory Access: LogiCORE IP Product Guide[EB/OL]. (2017-10-04) [2021-04-02]. https://china.xilinx.com/ support/documentation/ip_documentation/axi_vdma/v6_3/pg020 _axi_ vdma.pdf .

[14] NVDIA Inc. 8 bit inference with TensorRT [EB/OL]. (2017-05-08) [2021-05-23]. https://on-demand. gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf.