2020 IEEE International Conference on
Integrated Circuits, Technologies and Applications

A High-Speed Parallel FPGA Implementation of Harris Corner Detection

Hongtao Zhou[1,2], Runjiang Dou[1], Shuangming Yu[1], Nan Qi[1,2], Jian Liu[1,2], Nanjian Wu[1,2], and Liyuan Liu[1,2,*]

[1] State Key Laboratory of Superlattics and Microstructures, Institute of Semiconductors, Chinese Academy of Science, Beijing, China
[2] Center of Materials Science and Optoelectronics Engineering, University of Chinese Academy of Sciences, Beijing, China

***Abstract*—Harris corner detection is a common operation in the field of image processing, but suffering from computation complexity in embedded system. The current method of corner detection is to process one pixel in one clock cycle. With the improvement of image acquisition method in modern society, higher processing speed requirements have been proposed for the implementation of corner detection. This paper presents a new pixel-level parallelism FPGA implementation for Harris corner detection, which double the processing speed or more with almost no decreasing in accuracy. It is implemented on FPGA in the high-speed camera, and the experimental results show that the implementation is suitable for real-time applications.**

***Keywords*—FPGA, corner detection, real-time, pixel-level parallel**

I. INTRODUCTION

Corner is an important feature of the image. It is widely used in computer vision, such as object tracking, 3D image reconstruction, image stitching, etc. Harris corner[1] is commonly used for corner detection. To get higher processing speed, many Harris corner detection acceleration methods have been proposed, including using CPU-GPU heterogeneous device, FPGA implementation, and HW/SW collaborative implementation.

Some shortcomings still exist in these implementations. Firstly, it is difficult to achieve high frame rate using float point calculation, for the constraint of the limited resource. Secondly, since the current implementation can only process one pixel in a clock cycle, the detection performance is highly related to the clock frequency. In practical applications, multiple pixels were output in one clock cycle to get high frame rate. Current implementation can not work real-time. So we proposed a new FPGA implementation using fixed point and processing multiple pixels in one clock cycle to meet real-time requirements.

The main contribution of this paper is: use fixed point in processing flow and propose a pixel-level parallel implementation to calculate multiple pixels in one clock cycle. At the same frequency, it can efficiently improve performance and achieve higher throughput. Compared with the existing implementation, the performance has been improved and the resource utilization is reasonable.

II. OVERVIEW OF THE HARRIS CORNER DETECTION

A. Harris corner detection

Harris corner is located at the edge intersection point, so that the pixel grayscale changes significantly around the corner. Based on the property, the local window is moved slightly around pixel to obtain the grayscale change.

There are several steps in Harris corner detection, as shown in Fig.1: 1) Calculate the gradient image of the original image, 2) Calculate the Gaussian filtering result of the gradient image, 3) Get the Hessian matrix at each pixel and Harris corner response value, 4) Non-maximum suppression in a local window, 5) Select points with response value greater than the threshold as corner.

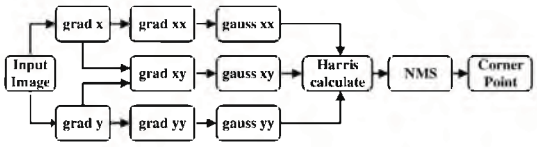

Fig. 1. The calculation process of the Harris corner detection.

B. Parallelism Analysis

1) Parallelism between different tasks:

Process different tasks in parallel reduce storage resource occupation and improve processing efficiency, such as gradient filtering and Gaussian filtering.

2) Parallelism in computing level:

Computing parallel represents the different operations pipelined. In FPGA implementation, the use of pipeline stages further improves computational parallelism.

3) Parallelism in pixel level:

In current corner detection implementation, parallelism in pixel-level was not considered. Multiple pixels processing can share the same buffer resource in one clock cycle.

So we use fixed-point calculations to reduce resource occupation. Take advantage of parallelism in different levels, especially in pixel level, this implementation can break through the frequency limit and double the processing speed or more.

III. THE PIXEL-LEVEL PARALLELISM ARCHITECTURE

A. Harris corner response value calculation

This paper uses high level synthesis to implement corner detection. The gradient and Gaussian filtering are performed using the sliding window method. The pixel parallelism of the implementation is P. Input data is stored in line buffer and P pixels are output every clock cycle, as shown in Fig.2.

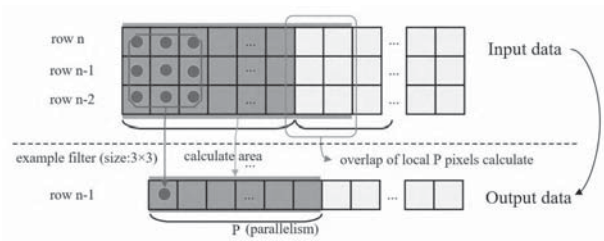

Fig. 2. Slide window implementation in pixel level.

During the gradient image calculation process, the horizontal and vertical directions data can be calculated using the same filtering kernel. The cross term and the squared term calculation in Hessian matrix use the pipeline to process the AXI-Stream data. Similarly, Gaussian filtering is also completed in pixel-level parallel.

B. Non-maximum suppression

Non-maximum suppression is required in the horizontal and vertical neighborhoods to obtain stable corner. Suppose the pixel parallelism is P. Fig.3 shows that the horizontal adjacent response values are compared step by step to get the maximum value after P response values output. At the same time, the current response value is compared with the adjacent rows upper and lower in vertical direction,as shown in Fig.4.

2020 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA) | 978-1-7281-8032-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICTA50426.2020.9332111

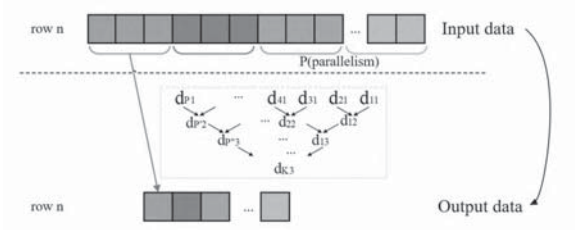

Fig. 3. Non-max suppression in horizontal rows.

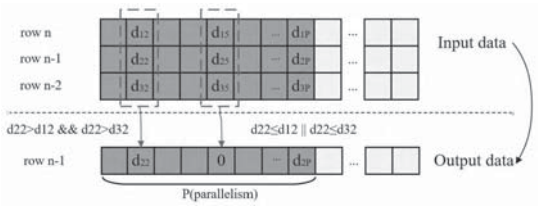

Fig. 4. Non-max suppression in vertical columns.

IV. EXPERIMENTAL RESULTS

A. Resource and speed:

1) Different pixel parallelism analysis

Fig.5.a shows the consumption of BRAM, FF, and LUT resources. As pixel parallelism (P) increases, LUT becomes the resource bottleneck of the implementation. As shown in Fig.5.b, the processing speed is assumed to be 1 and the LUT occupancy is 15.3% when P is 1. The processing speed doubles and the resource occupancy of the LUT per pixel parallelism decreases to 10.5% when P increases to 2, reducing LUTs usage about 30% per pixel parallelism. As P further increases, the LUT used per pixel parallelism decreases first and then increases, indicating that the acceleration benefits are decreasing. The occupied LUTs per pixel parallelism are the least and the processing speed becomes ×3.8 when P is 4. Therefore, the optimal resource consumption achieves when P is 4.

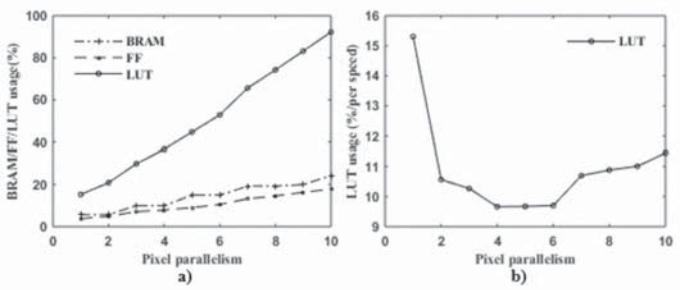

Fig. 5. a) FPGA resource occupancy under different degrees of parallelism. b) LUT resource occupancy under different acceleration ratios.

2) Compared with different implementations

TABLE I. RESOURCE COMPARISON

Implement	[2]	[3]	this work
LUTs	9849	6267	8035
FF	4335	-	8404
BRAM(18K)	5	11	17
BRAM(36K)	64	0	0
DSP	0	0	0
Frame/s	144	295	626

In comparison, the same FPGA chip xc7z020clg484 is used. The input image resolution is 640×480, and the results are shown in Table 1. Compared with [2], this paper uses fewer LUTs and BRAM resources. Compared with [3], the number of LUTs used is about 128%, but the processing speed is doubled. This implementation can work at the maximum frequency of 143 MHz with 626 frame/s@640×480.

B. Image test result

Fig.6 shows the effect of corner detection by OpenCV and FPGA. The left is detected by OpenCV using float point, and the right is the corner detected by FPGA using fixed point. This implementation gets almost the same effect as OpenCV.

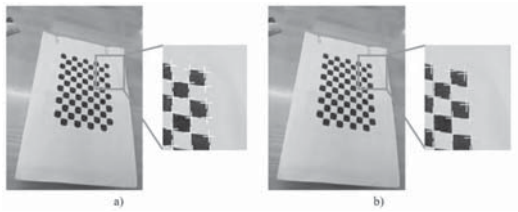

Fig. 6. Harris corner detected by OpenCV and FPGA.

This implementation is also applied to a high-speed camera [4]. Cropping out the image around the target, the test result is shown in Fig.7. These show that the implementation can effectively detect Harris corner on the target.

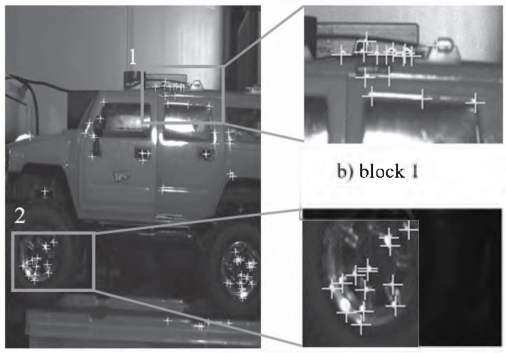

Fig. 7. Harris corner in image sequence captured by the high-speed camera

V. CONCLUSION

A high-performance Harris corner detection is implemented on FPGA. The pixel-level parallelism of implementation significantly improved the speed of corner detection and the use of fixed-point data reduced resource consumption. Its high-speed processing capabilities can be adopted in the increasing real-time requirement of different applications.

ACKNOWLEDGMENT

This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2204300, the Beijing Municipal Science and Technology Project under Grant Z181100008918009, the National Natural Science Foundation of China under Grant (61874107, 61704167).

REFERENCES

[1] Harris, Christopher G., and Mike Stephens. "A combined corner and edge detector." Alvey vision conference. Vol. 15. No. 50. 1988.

[2] Chao, Tak Lon, and Kin Hong Wong. "An efficient FPGA implementation of the Harris corner feature detector." 2015 14th IAPR International Conference on Machine Vision Applications (MVA). IEEE, 2015.

[3] Amara, Abdelkadder Ben, et al. "Zynq FPGA Based Memory Efficient and Real-Time Harris Corner Detection Algorithm Implementation." 2018 15th International Multi-Conference on Systems, Signals & Devices (SSD). IEEE, 2018.

[4] Dou, Runjiang, et al. "Development of high-speed camera with image quality evaluation." 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC). IEEE, 2019.