Architecture/algorithm joint exploration of Hough transform on FPGA

Houda Omari, Dominique Houzet, Virginie Fresse, Abdellatif Mtibaa

Architecture/algorithm joint exploration of Hough transform on FPGA

Houda OMARI
Images and signal
GIPSA Lab
Grenoble, France
houda.omari@gipsa-lab.grenoble-inp.fr

Dominique HOUZET
Images and signal
GIPSA Lab
Grenoble, France
dominique.houzet@gipsa-lab.grenoble-inp.fr

Virginie FRESSE
Electronics
LaHC
Saint-Etienne, France
virginie.fresse@univ-st-etienne.fr

Abdellatif MTIBAA
Electronics
Monastir, Tunisia
abdellatif.mtibaa@enim.rnu.tn

***Abstract*—The Hough transform is a robust algorithm used in shapes and objects recognition and it proves a high quality of results. This algorithm is also known for its use of large computing power. This paper presents a dedicated accumulator cache to optimize Hough space access. We explored the design space by evaluating the locality of accesses and found up to a speedup of 30 with a 2-line cache.**

Keywords—Hough transform; exploration of parameters; FPGA; cache memory

I. INTRODUCTION

Energy is now the key limiter of performance, forcing digital circuit designs to use large-scale parallelism with heterogeneous cores operating at low frequency and low voltage. Aggressive use of customized accelerators yields the highest performance and greatest energy efficiency on many applications. This customization is both algorithm and hardware dependent, and also depends on the other applications implemented in the same circuit. There is a global system exploration to conduct in order to optimize a full system. This can be only done with parameterization of both the hardware and the algorithm to implement.

Linear Hough transform is highly used to detect lines in a M x N image, but is compute intensive as it writes for each edge point a sinusoid in the input Hough space of size $180 \times \text{sqrt}(M^2 \times N^2)$ for 180 angles. All the publications on Hough transform optimization either study it without real hardware consideration [1] or with fixed criteria like no multiplication operators [2] or minimum memory footprint [2,3] or no limit on number of operators to achieve highest performance [3].

Our original key idea is to explore different hardware parameters and algorithm parameters together to find an optimal solution according to a compromise between resources and performance, in the case of FPGA. On contrary to [2], we consider that multipliers are available in almost all FPGA, thus we can benefit from them by optimizing their use. The aim is to well use all the limited resources of FPGA (DSP, block RAM, LUT, latch) in a balanced way.

We propose also a hardware mechanism based on a register file to accumulate neighboring votes before storing them in the vote memory. This is already done by [2] and [3] but in a fixed way, with a single register. We propose to explore a larger register file used like a cache memory with some writing policy. We tested here only the Least Recently Used polity. We also explore a cache-line based file allowing aligned memory accesses to the vote memory.

This paper is organized as follows. Section II presents our proposed architecture for Hough transform. Results are given in Section III. Finally, Section IV concludes the paper.

II. PROPOSED ARCHITECTURE FOR HOUGH TRANSFORM

The hardware mechanism proposed in this paper is based on a register file or cache structure to accumulate neighboring votes of Hough space before storing them in the memory. The register file varies from 4 to 32 registers and the cache-line structure varies from 1 line of 4 registers to 8 lines. The performance of the implementation is directly proportional to the number of writes to the vote memory implemented in a block RAM. This is true if we can extract some parallelism of writes in the intermediate registers. This can be done by reading several input edge points together. This is possible with compression of input data with zero run-length encoding like in [2]. This is another parameter to explore. Fig. 1 presents the architecture explored.

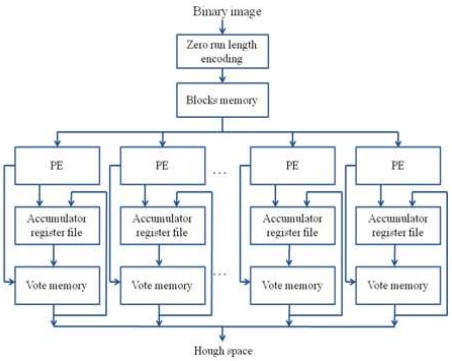

Figure 1. Proposed architecture for Hough transform.

We consider here blocks of 2 x 4 and 1 x 8 edge points from the input binary edges image. The aim of the zero run-length encoding proposed here is to simplify the processing as the memory constraint is mainly on the vote memory that has to be duplicated for each angle processed in parallel.

The first hardware parameters considered are: size of register file/ cache-line structure, number of vote memories, size of vote memory bus, size of zero run-length encoded blocks memory, and the first algorithm parameters are: threshold of binarization that is the number of edge points in the input image and size of zero run-length encoding blocks.

We explore those parameters with different images exhibiting different quantities of lines to be detected. We evaluate both resources and performances.

III. Results

The proposed architecture has been explored on three images of size 512 x 512. Two edge detection algorithms have been used: a Prewitt 2 x 2 filter and a Deriche filter.

The input data from the binary image are grouped in blocks of 1x8 or 2x4 edge points by the zero run-length encoding. The following table shows the non-zero blocks for the House image showing the compression ratio. We note that the number of non-zero blocks for edge detection based on Deriche filter is more important than for 2x2 filter. This is explained by the fact that the contours of Deriche are thinner and thus spatial locality of data is smaller.

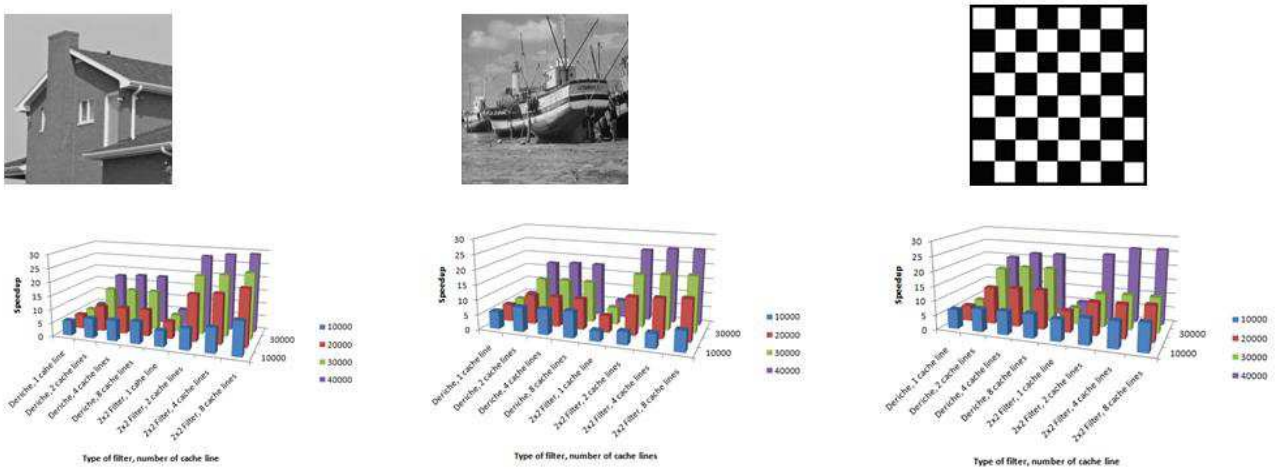

Figure 2. Exploration of cache-line structure with 1x8 block size

Table 1. Number of non zero blocks in House image.

Type of filter	Block of edge points	Number of edge points			
		10000	20000	30000	40000
Filter 2x2	1x8	6202	7176	8773	12389
	2x4	4137	5457	7167	10502
Deriche filter	1x8	5233	10981	15641	19256
	2x4	4275	8868	13115	16734

The number of memory access is a very important parameter to consider because it influents the execution time. Deriche contours are thinner than the contours of the filter 2x2, so there is less temporal locality which produces more memory accesses. The speedup presented in Fig. 2 corresponds to the number of vote memory accesses compared to the number of edge points processed. With multi-bank memory for edge points we can read and process in parallel many edge points at each clock cycle to achieve the highest performance given by this speedup. The best execution time is thus: (clock period * number of edge points)/speedup.

IV. Conclusion

We can conclude here that a low-cost register file of 8 registers can reduce the number of accesses to the vote memory by a ratio of 3 to 14 (speedup), and by a ratio of 6 to 30 with a cache-line structure. With a pipelined design, we can reduce the execution time by this ratio to obtain a better execution time than any published solution with the same resources. For instance an optimal compromise with reduced resources can be obtained to achieve the same performances as [3] but with no resources limitation. By exploring other parameters and implementing our solutions in a real FPGA we can measure the different resource/performance results validating the approach.

References

[1] S. S. Sathyanarayana, R. K. Satzoda, T. Srikanthan, "Exploiting inherent parallelisms for acclerating linear Hough transform", IEEE Transactions On Image Processing, Vol. 18, No. 10, October 2009.

[2] Z. H. Chen, A. W. Y. Su, M. T. Sun, "Resource-Efficient FPGA architecture and implementation of Hough transform", IEEE Transactions On Very scale Integration Systems, Vol. 20, No. 8, August 2012.

[3] X. Zhou, Y. Ito, K. Nakano, "An FPGA implementation of Hough transform using DSP blocks and block RAMs", Bulletin of Networking, Computing, Systems and Software, Vol. 2, No. 1, pp 18-24, January 2013.