International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:268
Vol.2, No.6, June 2013 DOI:10.15693/ijaist/2013.v2i6.49-53

Implementation of Hough Transform Using Resource Efficient FPGA Architecture

S.Subbiah
PG Student/VLSI Design
Dr. Sivanthi Aditanar College of Engineering,
Tiruchendur,India

A.Rega
Assistant Professor/Department of ECE
Dr. Sivanthi Aditanar College of Engineering,
Tiruchendur,India

Abstract **- Hough transform is used for detecting straight lines, circles in an image. To reduce the huge computations in Hough transform, I want to generate a resource efficient architecture and implement the Hough transform on an FPGA. Resource efficient and reduction in processing time are achieved with data parallelism. In parallelism, the images are divided into blocks and increment property is applied within a block and between the blocks. The implementation of Hough transform on an FPGA by exploiting both angle level and pixel level parallelism. I use two accumulators to facilitate the inter block increment and zero blocks are skipped by a run length coding scheme. The intra block increment is used to reduce the number of resources required. The matrix based lossless compression reduces the compression ratio. This architecture is implemented using altera device at operating frequency of 200MHz.**

Keywords – Hough transform, FPGA, parallelism, resource.

I. INTRODUCTION

The Hough transform [1] is a technique used in image analysis, computer vision, and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform. The classical Hough transform was concerned with the identification of lines in the image.

The simplest case of Hough transform is the linear transform for detecting straight lines. In the image space, the straight line can be described as $y = mx + b$ and can be graphically plotted for each pair of image points (x, y). In the Hough transform, a main idea is to consider the characteristics of the straight line not as image points (x_1, y_1), (x_2, y_2), etc., but instead, in terms of its parameters, i.e., the slope parameter m and the intercept parameter b. Based on that fact, the straight line $y = mx + b$ can be represented as a point (b, m) in the parameter space. However, one faces the problem that vertical lines give rise to unbounded values of the parameters m and b. For computational reasons, it is therefore better to use a different pair of parameters, denoted ρ and θ (*theta*), for the lines in the Hough transform. These are the polar coordinates. Using this parameterization, the equation of the line can be written as,

$$x \cos\theta + y \sin\theta = \rho \quad (1)$$

The parametric equation of the circle can be written as

$$(x-a)^2 + (x-b)^2 = r^2 \quad (2)$$

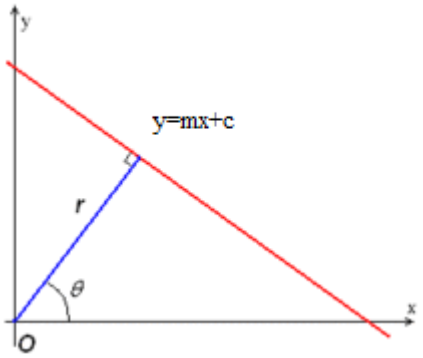

Fig. 1 Straight line passing through the axis

For embedded applications it requires hardware accelerators to achieve real time Hough transform. Compared with application specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs) usually target smaller markets and require much less development time. In FPGA, high throughput is often achieved by exploiting the parallelism of the design rather than by operating the chip at a very high clock frequency. In addition, a better architecture should have more efficient utilization of the function blocks in the FPGA. Here the implementation of Hough transform on an FPGA by exploiting both angle-level and pixel-level parallelism. The goal is to achieve the highest throughput with the minimum hardware resource.

II. RELATED WORKS

Because the general Hough transform is very computationally intensive, other line-detection schemes have been proposed, such as gradient-based Hough transform and kernel-based transform [2]. These schemes require fewer computations than Hough transform, but they still require a high-end CPU, which is often unavailable in practical applications, or special hardware devices to achieve real-time performance. The gradient-based Hough transform has been implemented in special hardware [3], but kernel-based Hough transform, which adopts a link list data structure, is difficult to implement efficiently in hardware. There has been some research to implement the Hough transform on special hardware, such as graphic processors [5], scan line array

processors [4], and pyramid multiprocessors [5]. However, these devices are unsuitable for low-cost embedded systems. This paper focuses on the implementation of the general Hough transform using FPGA.

One straightforward method to implement Hough transform is using multipliers [6]. However, multipliers are less available on low-end FPGAs. Hence, some researchers implement Hough transform using a coordinate rotation digital computer (CORDIC) [7] or a simplified CORDIC algorithm such as the multisector algorithm. CORDIC is an arithmetic technique developed by Volder [8] to solve trigonometric problems by rotating a vector in small angles until the desired angle is achieved. The CORDIC algorithms for FPGA are surveyed in [9]. CORDIC could implement the Hough transform using only shifters and adders rather than multipliers. The major disadvantage is that it requires multiple iterations to obtain one in the parameter space. Hence, pipelined CORDIC implementations are proposed to improve the throughput; however, the required resources are also increased. Another drawback of CORDIC is that the result produced is not the correct value but the correct value with a constant gain. The gain can be eliminated by applying an inverse gain to the initialization vector. However, this also requires additional resources. In [10], an FPGA platform is proposed to implement Hough transform by using hybrid-log arithmetic. In [11], distributed arithmetic (DA) architecture is proposed to implement Hough transform with shift–add operations. Unfortunately, it also requires multiple iterations to obtain one in the parameter space.

An accumulator-based architecture is proposed in [12]. An angle-level parallelism is applied in this architecture. It could obtain a point in the parameter space by a single accumulation. However, all of the pixels in the binary feature image need to go through the computation pixel by pixel, which limits its throughput. Additive Hough transforms utilizes the properties of Hough transform. Although the pixel-level parallelism is facilitated in this architecture, the memory requirement is also increased in proportion to the parallelism. Another drawback is that it requires additional cycles to get the entire parameter space.

Incremental Hough transforms are modified Hough transforms for the hardware implementation. It reuses previously computed values to derive another point in the parameter space. Hough transform is not only computation-demanding but also memory-demanding. In [13], a line-based implementation is proposed to reduce the bandwidth requirement on SIMD architecture. Here authors propose a memory-efficient implementation of Hough transform by using circular buffers of DSP processors. The memory requirement is reduced by storing the coordinates of a binary feature image rather than entire binary feature image.

In [14], a modified cache-friendly Hough transform is proposed to reduce the memory requirement and parallelism overhead on multiprocessors. In this paper we utilize the incrementing property of Hough transform to achieve an efficient architecture for implementing Hough transform. The proposed architecture facilitates both pixel-level and angle-level parallelisms. Unlike [12], the memory requirement is not increased in proportion to the parallelism. We propose using run-length encoding to skip unnecessary computations and memory accesses. Run-length encoding is widely used for data compression.

III. MATRIX BASED LOSSLESS COMPRESSION

The algorithm is based on only two matrices named as Binary matrices and Gray scale matrices. Hence it is simple to compress and decompress the data than the other methods. The main steps of the proposed algorithm are as follows:

Step 1: Read the original data matrix [OR]

Step 2: Construct the Binary Matrix [BM] and Grayscale Matrix [GSM] based on the following steps

Step 3: Compare each pixel in the matrix [OR] with the previous pixel in the same matrix.

Step 4: The binary matrix elements are calculated as follows:

$$[BM]_{i,j} = \begin{bmatrix} 0 & \text{if} \quad [OR]_{i,j} = [OR]_{i,j+1} \\ 1 & \text{otherwise} \end{bmatrix} \tag{3}$$

Step 5: First element in [GSM] is set to be equal to the value of the first pixel of [OR]

Step 6: The rest of the elements of [GSM] are calculated as follows:

$$[GSM]_k = \begin{bmatrix} \text{nul} & \text{if} \quad [OR]_{i,j} = [OR]_{i,j+1} \\ [OR]_{i,j} & \text{otherwise} \end{bmatrix} \tag{4}$$

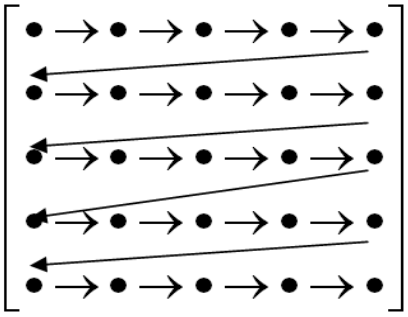

Fig. 2 Original image pixel comparison

Step 7: The original data can be reconstructed as follows:

$$[rec\_img]_{i,j} = \begin{bmatrix} [GSM]_k & \text{if} \quad [BM]_{i,j} = 0 \\ [GSM]_{k+1} & \text{if} \quad [BM]_{i,j} = 1 \end{bmatrix} \tag{5}$$

The criteria used for the comparison is the compression ratio achived which is defined as follows:

$$\text{Compression ratio (CR)} = \frac{\text{Original file size}}{\text{Compressed file size}}$$

IV. PROPOSED HOUGH TRANSFORM ARCHITECTURE USING FPGA

Matrix based lossless compression method has little effect on the complexity of the overall circuits and can reduce the data bandwidth, in the system that implemented for our

specific application. A block diagram of proposed architecture is shown in Fig.2.

The functional blocks in the PE are

- Inter Block Incrementing
- Intra Block Incrementing
- Vote consolidation
- Vote alignment

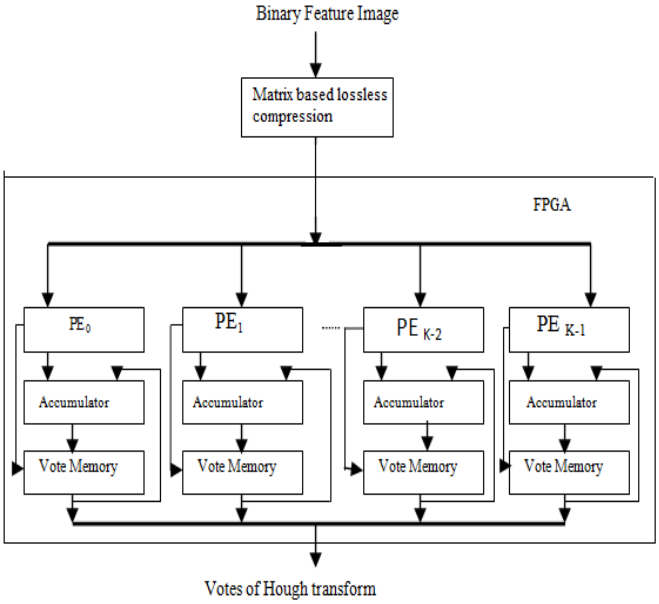

Fig. 3 Proposed architecture of Hough transform using FPGA

A. Inter Block Incrementing

Two accumulators can be used to implement the inter-block incrementing. In order to skip zero blocks, a step table is introduced in the proposed architecture. In Fig. 3 two PEs used for calculating the votes for θ and sinθ can share the same step table. In the implementation, the maximum number of successive zero blocks is set to 15 to limit the size of the step table. Hence, there are only 16 entries in each step table. The result is represented in a fixed-point format, i+f, where i is the integer part and f is the fractional part.

B. Intra Block Incrementing

The computed ρ values can be used to calculate all the other values of the pixels in the block used to find the cosθ and sinθ values. This will result in seven more ρ values. For the whole block, the eight votes in the memory addressed by the eight ρ values will need to be accumulated. The computed ρ is divided into the integer part i and the fractional part f. To result in an efficient circuit, only the fractional part f is used for calculating the vote-offsets relative to for the pixels in the block. The first stage calculates the vote-offsets for the *i*'th pixel in the block. The vote-offsets range from 0 to 4, and are represented by 3-b numbers. These numbers are decoded by 3:8 decoders. Each decoder output contains eight lines indicating the vote offset value for each pixel. These lines will be used with combination logic to produce consolidated votes.

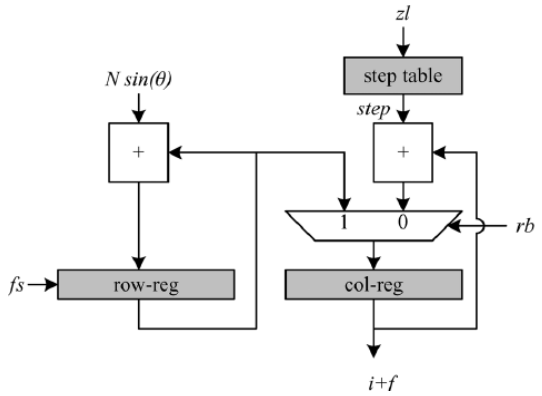

Fig. 4 Inter block incrementing

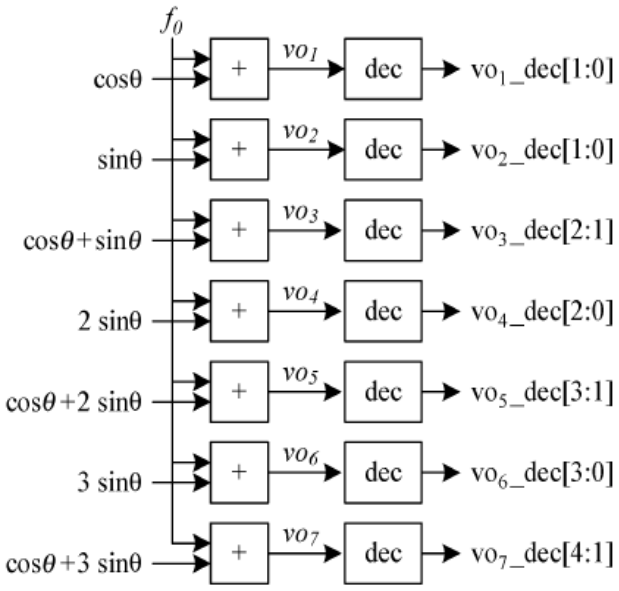

Fig. 5 Intra block incrementing

C. Vote Consolidation

The outputs of the decoders are combined with the values of the corresponding pixels (1 for feature pixels and 0 for non-feature pixels) using a combination logic circuit to determine the value of v, which represents the consolidated number of votes for each different vote offset. Instead of accessing the memory multiple times to accumulate the votes, we only need to access the memory once and each time we accumulate the memory contents in parallel. This is not possible without the vote consolidation, since the generated votes from the pixels may refer to the same memory location. Vote consolidation process shown in Fig. 5. This process produces the output which represents the number of votes with the value of ρ.

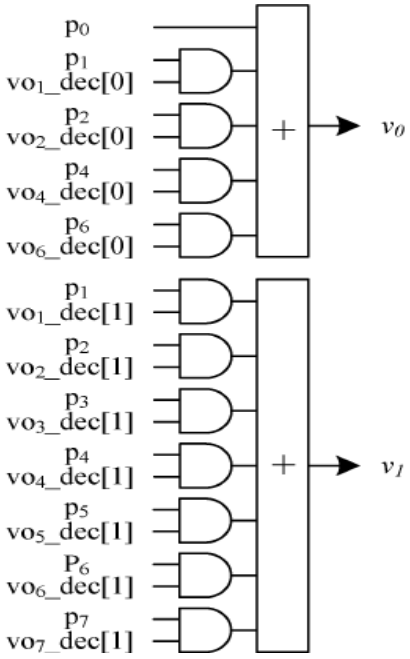

Fig. 6 Vote consolidation

D. Vote Alignment

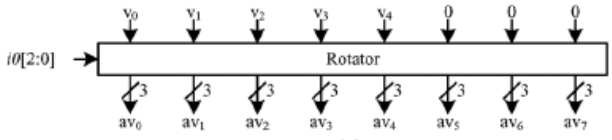

Fig. 7 Vote alignment

To update the contents of the memory locations i to i+4, instead of accessing the memory five times sequentially, we update the five memory contents in one clock cycle. This is possible since the five memory contents which need updating are stored in continuous memory locations with addresses from i to i+4 with our vote consolidation scheme. Here two 4K RAM blocks in the FPGA to implement the vote memory, because the accumulation requires one read and one write memory operation. The consolidated votes must be aligned based on the base address before they are accumulated to the Vote Memory. The votes are grouped in two groups, each group containing four votes. The lower four votes are stored in one 4K RAM and the upper four votes are stored in the other 4K RAM. In the circuit implementation, the upper bits of i are used to address the memory and the lower bits of i are used to align the votes.

V. RESULTS AND COMPARISON

TABLE I

SYNTHESIS RESULT OF PROPOSED ARCHITECTURE

Parameters	Existing	Proposed
Memory Bits	233360	159068
Number of IOs	34	25
Adders/Subtractors	45	34
Comparators	24	17

Table I shows requirement of resources comparison of existing and the proposed method. It deals the number of inputs, adders, sub-tractors and comparators requirement while Hough transform was implementing in FPGA.

TABLE II

EXECUTION TIME OF IMAGES

Image	Number of Symbols	Execution Time (ms)
Airplane	4873	2.28
Baboon	7800	3.61
Brick	5854	2.73
House	6203	2.88
Lena	4991	2.33
Peppers	4423	2.07

Table II shows that the execution time of Hough transform for different images and number symbols used in the image.

VI. CONCLUSION

The simple approach is proposed in this project is a resource efficient architecture for calculating Hough Transform. The incrementing property for both inter block and intra block incrementing are used to reduce the resource requirement. Two accumulators are used to facilitate the inter block incrementing, and zero blocks are skipped by introducing matrix based lossless compression scheme. The intra block incrementing could efficiently reduce the resource requirement. Instead of computing the ρ of every pixel in a block, vote offset is more efficient to determine the corresponding votes. The locality of a block is analyzed and the votes corresponding to an identical are consolidated in order to reduce the memory access and fully utilize the FPGA memory bandwidth. The result shows that the proposed PE

could achieve the best throughput with the same amount of resources compared to previously reported architectures.

REFERENCES

[1] R. O. Duda and P. E. Hart, "*Use of the Hough transformation to detect lines and curves in pictures*," Commun. ACM, vol. 15, pp. 11–15, 1972.
[2] F. O'Gorman and M. B. Clowes, "*Finding picture edges through collinearity of feature points*," IEEE Trans. Comput., vol. C-100, pp. 449–456, 1976
[3] L. Lin and V. K. Jain, "*Parallel architectures for computing the Hough transform and CT image reconstruction*," in Proc. Int. Conf. Applic. Specific Array Processors, 1994, pp. 152–163.
[4] A. L. Fisher and P. T. Highnam, "*Computing the Hough transform on a scan line array processor*," IEEE Trans. Pattern Anal. Mach. Intell., vol. 11, no. 3, pp. 262–265, Mar. 1989.
[5] M. Atiquzzaman, "*Pipelined implementation of the multiresolution Hough transform in a pyramid multiprocessor*," Pattern Recognit. Lett., vol. 15, no. 9, pp. 841–851, 1994.
[6] K.Hanahara, T.Maruyama, and T. Uchiyama, "*A real-time processor for the Hough transform*," IEEE Trans. Pattern Anal. Mach. Intell., vol. 10, no. 1, pp. 121–125, Jan. 1988.
[7] F. Zhou and P. Kornerup, "*A high speed Hough Transforms using CORDIC*", Univ. Southern Denmark, Tech. Rep. PP-1995-27, 1995.
[8] J. E. Volder, "*The CORDIC trigonometric computing technique*," IRE Trans. Electron. Comput., vol. 8, no. 3, pp. 330–334, 1959.
[9] R. Andraka, "*A survey of CORDIC algorithms for FPGA based computers*," in Proc.ACM/SIGDA 6th Int. Symp. Field Programmable Gate Arrays, Monterey, CA, 1998, pp. 191–200.
[10] P. Lee and A. Evagelos, "*An implementation of a multiplierless Hough transform on an FPGA platform using hybrid-log arithmetic*," in Proc. SPIE, 2008, vol. 6811, p. 68110G.
[11] K.Mayasandra, S.Salehi, W. Wang, and H. M. Ladak, "*A distributed arithmetic hardware architecture for real-time Hough-transform-based segmentation*," Can. J.Electr. Comput. Eng.,vol. 30, no. 4, pp. 201–205, 2005.
[12] M.-Y.Chern and Y.-H. Lu, "*Design and integration of parallel Houghtransform chips for high-speed line detection" in Proc. 11th Int. Conf. Parallel Distrib.* Syst. Workshops, 2005, vol. 2, pp. 42–46.
[13] Y. He, Z. Zivkovic, R. Kleihorst, A. Danilin, and H. Corporaal, "*Realtime implementations of Hough transform on SIMD architecture*," in Proc. 2nd ACM/IEEE Int. Conf. Distrib. Smart Cameras, 2008, pp. 1–8.
[14] Y.-K. Chen, W. Li, J. Li, and T. Wang, "*Novel parallel Hough transform on multi-core processors*," in Proc. IEEE Int. Conf. Acoust., Speech Signal Process., 2008, pp. 1457–1460.

Authors Profile

S.Subbiah received the **B.E.** degree in electronics and communication engineering from the Sethu Institute of Technology, Kariapatti, Anna University, Chennai, India, in 2010. Currently doing **M.E.** in electronics and communication engineering (VLSI Design) in Dr.Sivanthi Aditanar College of Engineering, Tiruchendur, Anna University, Chennai, India. His research interest includes digital electronics and image processing.

A.Rega received the **B.E.** degree in electronics and communication engineering from the C.S.I Institute of Technology, Thovalai, Anna University, Chennai, India, in 2007 and received **M.E.** degree in electronics and communication engineering (Communication Systems) from PET Engineering College, Vallioor, Anna University, Chennai in 2009. She is presently working as a Assistant Professor in Dr.Sivanthi Aditanar College of Engineering, Tiruchendur, Anna University, Chennai, India. Her research interest includes communication systems and digital electronics.