FPGA BASED EDGE DETECTION: INTEGER INVERSE TANGENT ALGORITHM

Dimitre Kromichev

Department of Marketing and International Economic Relations, University of Plovdiv
24 Tzar Asen Str, Plovdiv 4000, Bulgaria

dkromichev@yahoo.com

Abstract

FPGA based edge detection targeting ultimate execution speed and relying on gradient direction to define image contours has to tackle the problem of computing integer inverse tangent both accurately and fast. This paper presents an integer inverse tangent algorithm. It is designed to be used in the gradient direction submodule of FPGA based edge detection computations. Having investigated the mathematical accuracy, maximum operating frequency and minimum number of clock cycles on the basis of ten Intel (Altera) FPGA families, it is ascertained that the proposed algorithm is capable of securing total accuracy and working at a frequency higher than the maximum operating frequency of embedded memory under all test conditions. It takes only three clock cycles to provide an accurate result.

1. INTRODUCTION

Because the hardware implementation of integer inverse tangent is complicated and slow, approximation patterns are employed as a tool of choice. In edge detection which uses gradient direction, detected contours' quality is negatively impacted by the inaccuracy of inverse tangent. Therefore, in FPGA based edge detection focused on ultimate execution speed, it is a must that the inverse tangent can guarantee total mathematical accuracy. The latter must be achieved within the same number of clock cycles required by the gradient magnitude submodule to execute, taking into account that gradient magnitude and gradient direction submodules work in parallel in FPGA.

In [3] FPGA based design targets the speeding up of $\tan^{-1}$ computations. The results are: execution time is 320 ns; accuracy is <0.01°. Taylor series expansion method is applied to transfer $\tan^{-1}$ to a polynomial form [3][11]. In [6] two-argument $\tan^{-1}$ is implemented in FPGA by using the piecewise polynomial approximation method with non-uniform segmentation. The inputs (x and y) are divided using radix-2 non-restoring division and the result is used as an input to Atan. The results are: maximum error ratio - 2.62%; execution time in Xilinx Spartan 6 - 260.5 ns. In FPGA based design of two-argument $\tan^{-1}$ [9][8][10] division of the two inputs is implemented by a logarithmic transformation using subtraction. In [2] studied is the FPGA implementation of fixed-point two-argument $\tan^{-1}$ by comparing CORDIC with two multiplier based techniques. It is concluded that CORDIC is fadter than the multiplier and table-based methods. In [12] presented are several approximations for four quadrant $\tan^{-1}$ using Lagrange interpolation and optimization techniques. It is concluded that second-order polynomial provides a favorable compromise between accuracy and computational cost and is well suited for implementation in hardware. In [5] it is pointed out that long latency is a main disadvantage of methods based on CORDIC, conventional LUTs and polynomial approximation. In [1] proposed is atan2 using look-up table with 101-points. The accuracy is increased by linear interpolation. The achieved frequency is 60 MHz. The conclusion is: the accuracy of the proposed method is better than the approximation techniques. In [4] proposed is a high-accuracy computation of fixed-point $\tan^{-1}$ using CORDIC and fast magnitude estimation. Maximum phase error is reduced from 414 LSB (angle error of 0.6355 rad) to 4 LSB (angle error of 0.0061 rad). In [7] described is an FPGA implementation of $\tan^{-1}$ which is based on using CORDIC using serial and pipelined CORDIC architectures.

The objective of this paper is to propose an integer inverse tangent algorithm targeting the computation of gradient direction in FPGA based edge detection. The mathematical accuracy, maximum operating frequency and minimum number of clock cycles of the algorithm must be investigated using ten Intel (Altera) FPGA families. The employed tools are: Scilab, Intel (Altera) Quartus, VHDL, TimeQuest

Timing Analyzer, ModelSim, The analyses and conclusions are relevant to gray scale images.

2. THE PROPOSED INTEGER INVERSE TANGENT ALGORITHM

Application of the algorithm: computation of gradient direction. Goal of the algorithm: guarantee that

$$F_{\max}(\tan^{-1} Int) > F_{\max}(embMem)$$

$$nTclk_{\min}(\tan^{-1} Int) = const = 3 \tag{1}$$

where

$F_{\max}(embMem)$	is maximum operating frequency of embedded memory,
$F_{\max}(\tan^{-1} Int)$	is maximum operating frequency of the proposed algorithm,
$nTclk_{\min}(\tan^{-1} Int)$	is minimum number of clock cycles required by the algorithm to execute.

Because the gradient direction values can only be 0, 90, 45 and 135, four equations are defined:

$$\tan^{-1}\left(\frac{G_Y}{G_x}\right) = 0 \tag{2}$$

where

G_y is y gradient, $G_y \in [-255, 255]$

G_x is x gradient, $G_x \in [-255, 255]$,

$$\tan^{-1}\left(\frac{G_Y}{G_x}\right) = 45 \tag{3}$$

$$\tan^{-1}\left(\frac{G_Y}{G_x}\right) = 90 \tag{4}$$

$$\tan^{-1}\left(\frac{G_Y}{G_x}\right) = 135 \tag{5}$$

Solving (2), (3), (4) and (5) requires finding the domains of four functions with predefined ranges

$$A_0 = \tan^{-1}\left(\frac{m_0}{n_0}\right) \tag{6}$$

$$A_{45} = \tan^{-1}\left(\frac{m_{45}}{n_{45}}\right) \tag{7}$$

$$A_{90} = \tan^{-1}\left(\frac{m_{90}}{n_{90}}\right) \tag{8}$$

$$A_{135} = \tan^{-1}\left(\frac{m_{135}}{n_{135}}\right) \tag{9}$$

where

A_0, A_{45}, A_{90}, A_{135}	are angular sectors in which the axes 0, 45, 90 and 135 are bisectors,
m_0, n_p, m_{45}, n_{45},	
$m_{90}, n_{89}, m_{135}, n_{135}$	are independent variables and m_0, n_p m_{45}, n_{45}, m_{90}, n_{89}, m_{135}, n_{135} are within $[-255, 255]$.

Therefore, the task is to define the complete set of values for the independent variables in (6), (7), (8) and (9). To accomplish this task, the following aspects must be considered:

1) In order to avoid division by 0 the operation division must not be used.

2) The axes for 0, 45, 90 and 135 are further divided into two pairs. The ingredients of each pair are orthogonal. Hence, the angular sectors for 45 and 135 are symmetrical with respect to the x-axis. Therefore

$$|m_{45}| = |m_{135}| \tag{10}$$

and

$$|n_{45}| = |n_{135}| \tag{11}$$

As a result, the difference between 45 and 135 is based on sign relations.

3) Unlike 45 and 135, the difference between 0 and 90 is defined by the fact that their angular sectors are symmertrical with respect to axis 45. As a result, the difference between 0 and 90 is based on comparison with respect to the boundaries of the angular sector for 45.

3. COMPUTATIONAL MECHANISM IN FPGA

The algorithm includes:

Step #1. Determine all combinations between the signs of Gy and Gx.

Step #2. Define two reference points: 22.5° and 67.5°.

Step #3. Determine a numerical equivalent to angle 22.5°. The accurate representation of angle 22.5° is the fraction $\frac{99}{239}$:

$$\tan^{-1}\left(\frac{99}{239}\right) = 22.500605394851°$$

Step #4. Determine a numerical equivalent to angle 67.5°. The accurate representation of angle 67.5° is the fraction $\frac{169}{70}$:

$$\tan^{-1}\left(\frac{169}{70}\right) = 67.500605394851°.$$

Step #5. Calculate gradient direction Dir by simultaneously executing expressions:

If $(Gy > 0 \& Gx > 0)$ or $(Gy < 0 \& Gx < 0)$ then

If $|Gx|*99 \geq |Gy|*239$

$Dir = 0$

If $|Gx|*99 < |Gy|*239 \& |Gx|*169 > |Gy|*70$
$Dir = 45$

If $|Gx|*169 \leq |Gy|*70$ $\quad Dir = 90$

If $(Gy > 0 \& Gx < 0)$ or $(Gy < 0 \& Gx > 0)$ then

If $|Gx|*99 \geq |Gy|*239$ $\quad Dir = 0$

If $|Gx|*99 < |Gy|*239 \& |Gx|*169 > |Gy|*70$
$Dir = 135$

If $|Gx|*169 \leq |Gy|*70$ $\quad Dir = 90$

If Gy = 0 & Gx ≠ 0 $\quad Dir = 0$

If Gy ≠ 0 & Gx = 0 $\quad Dir = 0$

If Gy = 0 & Gx = 0 $\quad Dir = 0$. (12)

The computational mechanism in FPGA is presented in Figure 1.

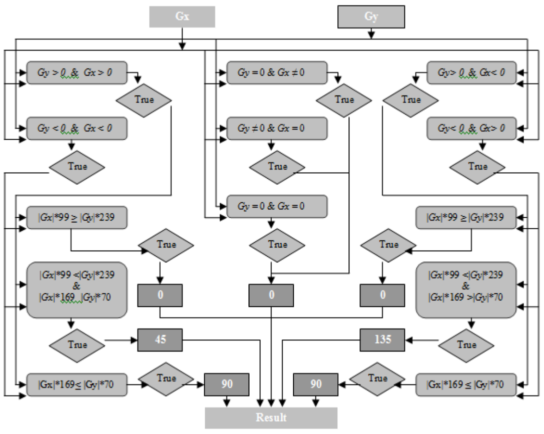

Figure 1. The model of computational mechanism in FPGA

The RTL design of the algorithm is shown in Figure 2.

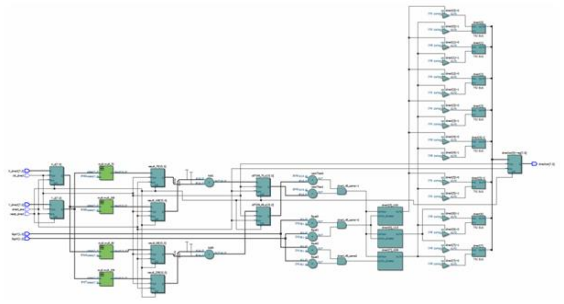

Figure 2. RTL design of the algorithm (Source: Intel (Altera) Quartus)

Resource utilization is presented in Table 1.

Table 1. Resource utilization of the proposed algorithm

FPGA family	Utilization by different resource types				
	Logic utilization	Total registers	Total memory bits	Total DSP blocks	Embedded multiplier 9 bit elements
Cyclone	118 (LEs)	-	-	-	-
Cyclone II	94 (LEs)	74	-	-	4
Cyclone III	80 (LEs)	38	-	-	4
Cyclone IV	80 (LEs)	38	-	-	4
Cyclone V	30 (ALMs)	38	-	4	-
Stratix	104 (LEs)	64	-	3	-
Stratix II	80 (ALUTs)	50	-	3	-
Stratix III	65 (ALUTs)	40	-	3	-
Stratix IV	65 (ALUTs)	40	-	3	-
Stratix V	32 (ALMs)	40	-	2	-

4. PROVING THE ALGORITHM'S MATHEMATICAL ACCURACY

Mathematical accuracy is tested for all values of Gy and Gx in the interval $[-255, 255]$. Four sample test results are presented below.

Check # 1

Gy = -160 Gx = -66

|-66|*99 ≥ |-160|*239 (false)

|-66|*99 < |-160|*239 &

|-66|*169 > |-160|*70 (false)

|-66|*169 $\leq$ |-160|*70 (true)

Therefore, $Dir = 90$.

Using the conventional method:

$\tan^{-1}\left(\frac{-160}{-66}\right)$ = 67.583852520656°.

Check # 2

Gy = -48 Gx = 55

55*99 $\geq$ |-48|*239 (false)

55*99 < |-48|*239 &

55*169 > |-48|*70 (true)

55*169 $\leq$ |-48|*70 (false).

Therefore, $Dir = 135$.

Using the conventional method: $\tan^{-1}\left(\frac{-48}{55}\right)$ =

- 41.1120904°.

Check # 3

Gy = -19 Gx = -46

|-46|*99 $\geq$ |-19|*239 (true)

|-46|*99 < |-19|*239 &

|-46|*169 > |-19|*70 (false)

|-46|*169 $\leq$ |-19|*70 (false)

Therefore, $Dir = 0$.

Using the conventional method: $\tan^{-1}\left(\frac{-19}{-46}\right)$ =

22.442753365294°.

Check # 4

Gy = 19 Gx = 45

|45|*99 $\geq$ |19|*239 (false)

|45|*99 < |19|*239 &

|45|*169 > |19|*70 (true)

|45|*169 $\leq$ |19|*70 (false).

Therefore, $Dir = 45$.

Using the conventional method: $\tan^{-1}\left(\frac{-19}{45}\right)$ =

22.890551656248°.

Accuracy tests using the entire range of values in $[-255, 255]$ provide the data:

1) Total calculated results: 261121

2) Total results different from 0: 260100

3) Total results equal to 0: 1021

4) Distribution of non-zero results:
- direction 0: 65025
- direction 45: 65025
- direction 90: 65025
- direction 135: 65025.

Thus it is proved that the fractions $\frac{99}{239}$ and $\frac{169}{70}$ are accurately calculated and the algorithm guarantees total accuracy.

5. EXPLORING $F_{\max}(\tan^{-1} Int)$ AND $nTclk_{\min}(\tan^{-1} Int)$ IN FPGA

Exploration methodology:

- The algorithm is implemented using all values in $[-255, 255]$.

The obtained results are in Table 2.

Test results prove the functional capabilities of the proposed algorithm:

- Total mathematical accuracy
- $F_{\max}(\tan^{-1} Int) > F_{\max}(mem)$ for all values of Gy and Gx

Table 2. Results for $F_{\max}(\tan^{-1} Int)$ and $nTclk_{\min}(\tan^{-1} Int)$

FPGA family	$F_{\max}(\tan^{-1} New)$ (in MHz)	$nTclk(\tan^{-1} New)$
Cyclone	179	3
Cyclone II	259	3
Cyclone III	331	3
Cyclone IV	334	3
Cyclone V	337	3
Stratix	313	3
Stratix II	428	3
Stratix III	534	3
Stratix IV	538	3
Stratix V	540	3

- $nTclk(\tan^{-1} Int) = const = 3$ under all test conditions.

The input data widths ≤ 8 bits for both the numerator and denominator in the reference points $\frac{99}{239}$ and $\frac{169}{70}$. Because image pixel is within $[0, 2^8 - 1]$, for Cyclone II-V and Stratix I-V, $F_{\max}(\tan^{-1} Int)$ is defined by the maximum operating frequency of 9x9 hard multiplier. For Cyclone, $F_{\max}(\tan^{-1} Int)$ is defined by the maximum operating frequency of 8x8 logic elements based multiplier.

6. CONCLUSION

This paper presents an integer inverse tangent algorithm. Its application is focused on computing gradient direction in FPGA based edge detection which targets ultimate execution speed. The designed algorithm is explored for mathematical accuracy, maximum operating frequency and minimum number of clock cycles in ten Intel (Altera) FPGA families.

References

[1] A. Ukil, V. H. Shah, and B. Deck, "Fast computation of arctangent functions for embedded applications: A comparative analysis", *2011 IEEE International Symposium on Industrial Electronics*, 2011, pp. 1206-1211

[2] F. De Dinechin and M. Istoan, "Hardware Implementations of Fixed-Point Atan2*", 2015 IEEE 22nd Symposium on Computer Arithmetic*, 2015, pp. 34-41

[3] Kung, Y.-S., Wu, M.-K., Linh Bui Thi and, H., Jung, T.-H., Lee, F.-C., and Chen, W.-C., " FPGA-based hardware implementation of arctangent and arccosine functions for the inverse kinematics of robot manipulator", *Engineering Computations*, Vol. 31 No. 8, 2014, pp. 1679-1690

[4] Luca Pilato, Luca Fanucci, and Sergio Saponara, "Real-Time and High-Accuracy Arctangent Computation Using CORDIC and Fast Magnitude Estimation", *Electronics*, 2017, pp. 6-22

[5] M. Saber, Y. Jitsumatsu, and T. Kohda, "A low-power implementation of arctangent function for communication applications using FPGA", *2009 Fourth International Workshop on Signal Design and its Applications in Communications,* 2009, pp. 60-63

[6] Omar Zeyad, "Design and Implement ation Hardware Architecture for Four–Quadratic Arctangent", *International Journal of Computer Applications,* Volume 176 – No.1, October 2017, pp. 10-13

[7] P. A. Kumar, FPGA "Implementation of the Trigonometric Functions Using the CORDIC Algorithm", *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS),* 2019, pp. 894-900

[8] Roberto Gutierrez and Javier Valls, "Low-power fpga-implementation of atan (y/x) using look-up table methods for communication applications, Journal of Signal Processing Systems, Volume 56, Issue 1, July 2009, pp. 25–33

[9] R. Gutierrez and J. Valls, "Implementation on FPGA of a LUT-Based atan(Y/X) Operator Suitable for Synchronization Algorithms", *2007 International Conference on Field Programmable Logic and Applications*, 2007, pp. 472-475

[10] R. Gutierrez, V. Torres, and J. Valls, "FPGA-implementation of atan(Y/X) based on logarithmic transformation and LUT-based techniques", *Journal of Systems Architecture*, vol. 56, 2010

[11] S. Nandi, S. Prasad, C. M. Ananda, and S. S. Rekha, "Fixed point implementation of trigonometric function using Taylor's series and error characterization", *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2016, pp. 442-446

[12] S. Rajan, S. Wang, and R. Inkol, "Efficient Approximations for the Four-Quadrant Arctangent Function", *2006 Canadian Conference on Electrical and Computer Engineering. IEEE*, 2006