FPGA BASED EDGE DETECTION: INTEGER SQUARE ROOT ALGORITHM

Dimitre Kromichev

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

dkromichev@yahoo.com

Abstract

In FPGA based edge detection which uses gradients to find contours, the calculation of integer square root focused on both accuracy and speed presents a serious problem. In this paper, proposed is an integer square root algorithm. Its application is focused on computing the gradient magnitude in FPGA based edge detection. The algorithm is explored for 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 algorithm guarantees total mathematical accuracy. Its maximum operating frequency is higher than the maximum operating frequency of embedded memory, and it requires a single clock cycle to execute. The proposed algorithm's capabilities are assessed on a comparative basis.

1. INTRODUCTION

In FPGA based edge detection which relies on gradient to detect image contours the accuracy of the obtained gradient magnitude value depends on the accuracy of square root calculations. Hardware implementation of integer square root is a serious problem due to the complexity of computations. Hence the most widely used approach is to prioritize speed over mathematical accuracy by resorting to different approximation patterns. The latter generally have a strong negative impact on the quality of detected contours. Therefore, when the focus is on both achieving the ultimate execution speed and guaranteeing the detected contours' quality FPGA based edge detection using gradient requires an integer square root algorithm which is capable of: 1) providing mathematically accurate result for the smallest possible count of clock cycles; 2) working at a clock frequency which is higher than the maximum operating frequency of the component defining the upper limit of clock frequency in FPGA based edge detection.

The algorithm which most current FPGAs use is radix 2 digit reccurence square root [5][6]. Digit recurrence methods rely on subtractions and iterations [8]. Hence, they have limited performance in hardware [7][15]. Another approach is the functional iteration which is divided into additive and multiplicative according to the operation used in each iterative step [3][9]. Newton–Raphson method has the disadvantage of using division [1]. FPGA focused modifications include: modified nonrestoring square root using only subtraction [12][13][14]; nonrestoring pipelined square root using only subtraction [4]; square root based on linear approximation subsystem with Look-up tables [10][11]; square root based on subtractors and multipliers - appropriate only for small numbers [16]; square root based on successive subtraction of odd integers [2].

The objective of this paper is to propose an integer square root algorithm. Its application is focused on computing the gradient magnitude in FPGA based edge detection. The task is to explore the algorithm for mathematical accuracy, maximum operating frequency and minimum number of clock cycles in ten Intel (Altera) FPGA families. Used tools: Scilab, Intel (Altera) Quartus, TimeQuest Timing Analyzer, ModelSim. The hardware description language is VHDL. Relevant to the conducted analyses and drawn conclusions are gray scale images.

2. THE PROPOSED INTEGER SQUARE ROOT ALGORITHM

The difference between the squares of any two integers is presented by

$$n^2 - (n-p)^2 = 2(n-p)p + p^2 \qquad (1)$$

$$p \in N, p \geq 1, p < n,$$

$$n \in N.$$

For $p = 1$ (1) becomes

$$n^2 - (n-1)^2 = 2(n-1) + 1. \qquad (2)$$

From (2) it follows that:

- The difference Dsq between the squares of any two consecutive integers $n-1$ and n is a constant represented by an odd number

$$Dsq = 2(n-1)+1. \qquad (3)$$

- The difference between any two consecutive differences is a constant

$$Dsq-(Dsq-1)=2. \qquad (4)$$

Hence, the following sequence can be defined

$$(((n+2)^2-(n+1)^2)-((n+1)^2-n^2))^{\infty}_{n=0} \qquad (5)$$

Thus, every single number $r \in N$ represents a radical which pertains to a specific interval $[n^2-(n-1), n^2+n]$ by satisfying the inequalities

$$r \geq n^2-(n-1) \,\&\, r \leq n^2+n. \qquad (6)$$

All radicals included in $[n^2-(n-1), n^2+n]$ are associated with a single integer *n* which represents the square root result according to

$$n = \sqrt{[n^2-(n-1), n^2+n]}. \qquad (7)$$

The accuracy of integer square root result depends on rounding. On the basis of (3), (4) and (5), the left- and rightmost values of $[n^2-(n-1), n^2+n]$ include rounding and guarantee mathematical accuracy.

3. COMPUTATIONAL MECHANISM IN FPGA

Application of the algorithm: gradient magnitude computation in FPGA based edge detection.

The upper limit of clock frequency of FPGA based edge detection which relies on gradient is defined by the maximum operating frequency of embedded memory. The smallest possible count of clock cycles required by an integer arithmetic operation to execute is 1. Therefore, the goal of the algorithm is: guarantee that for all possible values of the radical in edge detection

$$F_{max}(squareInt) > F_{max}(embMem)$$

$$nTclk_{min}(squareInt) = const = 1 \qquad (8)$$

where

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

In edge detection, all radicals are within $[0,(2^8-1)^2+(2^8-1)^2]$. In FPGA, the computational mechanism includes four steps:

Step #1. The value of gradient magnitude is within $[0,2^8-1]$. Because $\sqrt{0}=0$, $r=0$ is a special case and a separate interval is not used. Thus, radicals are distributed across 2^8-1 intervals according to (3), (4), (5), (6) and (7).

Step #2. Each of these intervals is associated with a single integer within $[1,2^8-1]$.

Step #3. The boundaries of all intervals are checked simultaneously. Of all checks, only for a single interval the boolean result is true. Thus the square root calculation is checking if a radical fits within the boundaries of a particular interval.

Step #4. The integer associated with this interval is the accurate integer square root result.

The model of computational mechanism is presented in Figure 1.

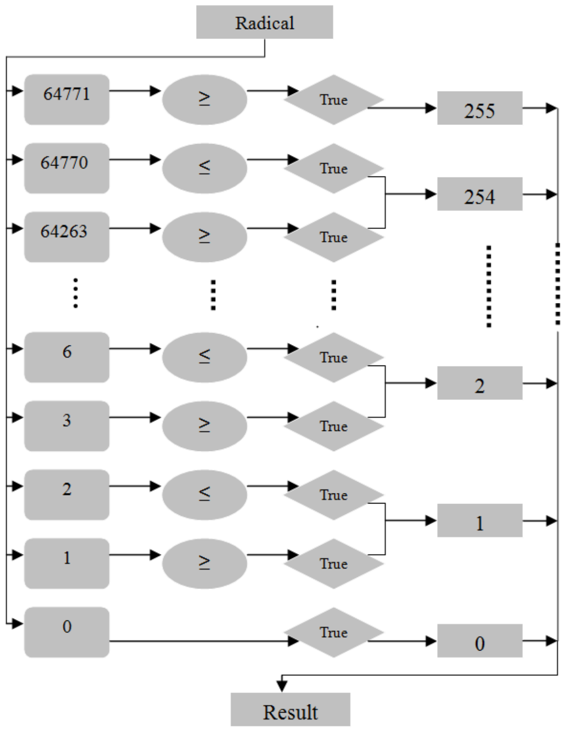
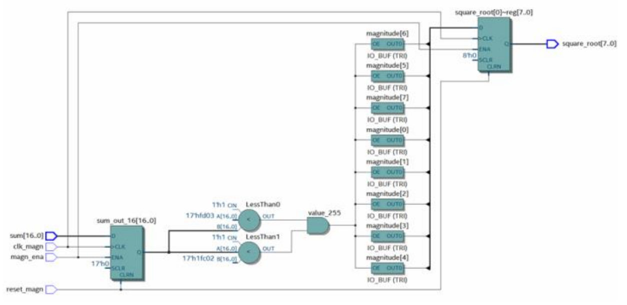
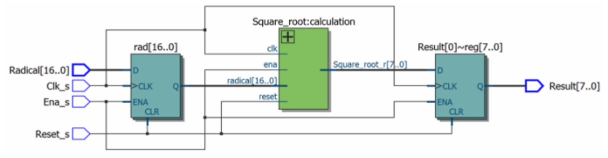


Figure 1. The model of computational mechanism of the algorithm

The RTL design is in Fig. 2.



a)



b)

Figure 2. RTL design of a single interval calculation (a) and the entire algorithm (b) (Source: Intel (Altera) Quartus)

Resource utilization is in Table 1.

Table 1. Resource utilization of the proposed algorithm

FPGA family	Utilization by different resource types				
	Logic utilization (in LEs/ ALUTs/ALMs)	Total registers	Total memory bits	Total DSP blocks	Embedded multiplier 9 bit elements
Cyclone	1461 (LEs)	-	0	-	0
Cyclone II	1386 (LEs)	272	0	-	0
Cyclone III	1290 (LEs)	264	0	-	0
Cyclone IV	1290 (LEs)	264	0	-	0
Cyclone V	487 (ALMs)	264	0	0	-
Stratix	1249 (LEs)	288	0	0	-
Stratix II	1014 (ALUTs)	282	0	0	-
Stratix III	992 (ALUTs)	270	0	0	-
Stratix IV	992 (ALUTs)	279	0	0	-
Stratix V	483 (ALMs)	270	0	0	-

4. PROVING THE ALGORITHM'S MATHEMATICAL ACCURACY

The proposed algorithm is tested for mathematical accuracy using all possible values of the radical in FPGA based gradient magnitude.

Critical to the mathematical accuracy of an integer square root algorithm is the accurate rounding. Sample results are presented below.

Check # 1

Radical: 35984. Therefore the interval is [35911, 36290]. The reference value is 190. This is the result. Conventional square root result: **189.6944**912220700203.

Check # 2

Radical: 33800. Therefore the interval is [33673, 34410]. The reference value is 184. This is the result. Conventional square root result: **183.8477**631085023563.

Check # 3

Radical: 42353. Therefore the interval is [42231, 42642]. The reference value is 206. This is the result. Conventional square root result: **205.7984**450864486002.

Check # 4

Radical: 37370. Therefore the interval is [37057, 37442]. The reference value is 193. This is the result. Conventional square root result: **193.3132**173442881789.

Another approach to proving accuracy is checking the boundary values of the intervals. Ten sample checks are presented in Table 2.

Table 2. Proving the accuracy by checking the boundary values of the intervals

Intervals under test	Conventional square root result (before rounding)		Result calculated with integer square root algorithm based on intervals
	Result of square root for the interval's leftmost value brfore rounding	Result of square root for the interval's rightmost value before rounding	
[63253, 63756]	251.501491	252.499504	252
[46441, 46872]	215.501740	216.499422	216
[43057, 43472]	207.501807	208.499400	208
[32221, 32580]	179.502089	180.499307	180
[26083, 26406]	161.502321	162.499230	162
[10921, 11130]	104.503588	105.498815	105
[7657, 7832]	87.504285	88.498587	88
[4161, 4290]	64.505813	65.498091	65
[343, 380]	18.520259	19.493588	19
[91, 110]	9.539392	10.488088	10

These checks prove that the proposed algorithm guarantees total mathematical accuracy.

5. EXPLORING $F_{max}(squareInt)$ AND $nTclk_{min}(squareInt)$ IN FPGA

Exploration methodology:

- The algorithm is implemented using all values in $[0,(2^8-1)^2+(2^8-1)^2]$.

The results are shown in Table 3.

Table 3. $F_{max}(squareInt)$ and $nTclk_{min}(squareInt)$ of the proposed algorithm

FPGA family	$F_{max}(squareInt)$ (in MHz)	$nTclk_{min}(squareInt)$
Cyclone	195	1
Cyclone II	286	1
Cyclone III	320	1
Cyclone IV	322	1
Cyclone V	329	1
Stratix	289	1
Stratix II	414	1
Stratix III	538	1
Stratix IV	586	1
Stratix V	613	1

Test results prove the functional capabilities of the proposed integer square root algorithm:

- Total mathematical accuracy of results

- $F_{\max}(squareInt) > F_{\max}(embMem)$ for all values of radical which can be calculated in FPGA based gradient edge detection
- $nTclk_{\min}(squareInt) = const = 1.$

All existing integer square root algorithms have two stages:

Stage #1. Execute operation square root over a radical r

$$\sqrt{r} = n^{*} \tag{9}$$

where

n^{*} is the square root result before rounding.

Stage #2. Execute rounding to obtain mathematically accurate square root result n

$$n^{*} + 0/1 = n \tag{10}$$

In the proposed integer square root algorithm based on intervals, Stage #1 and Stage #2 are combined into a single operation executed as a number of parallel comparisons. Therefore, with respect to the technology of executing comparison in FPGA, $F_{\max}(squareInt)$ depends on the propagation delay of ripple carry adder and sign check operation.

The proposed algorithm's speed characteristics are assessed on a comparative basis using the radix-2 iterative square root algorithm. The comparison between $F_{\max}(squareInt)$ and the highest operating frequency of the radix-2 iterative square root $F_{\max}(IterativeSquare)$ is conducted on the basis of executing the iterative square root within two clock cycles – a separate clock cycle is used for rounding. The comparison results show that $F_{\max}(squareInt)$ is higher than $F_{\max}(IterativeSquare)$ from 82.9% to 87.1%.

6. CONCLUSION

Proposed is an integer square root algorithm. Its application is focused on calculating gradient magnitude in FPGA based edge detection. The algorithm is explored for mathematical accuracy, maximum operating frequency and minimum number of clock cycles in ten Intel (Altera) FPGA families. Exploration results are assessed on a comparative basis using radix-2 iterative square root.

References

[1] Addanki Purna Ramesh and I. Jayaram Kumar, "Implementation of Integer Square Root," International Journal of Engineering Science and Innovative Technolog (IJESIT), Volume 4, Issue 1, January 2015, pp. 105-113

[2] Aiman Zakwan Jidin and Tole Sutikno, "FPGA Implementation of Low-Area Square Root Calculator," TELKOMNIKA, Vol. 13, No. 4, December 2015, pp. 1145~1152

[3] Anuja Nanhe, Gaurav Gawali, Shashank Ahire, and K. Sivasankaran, "Implementation of Fixed and Floating Point Square Root Using Nonrestoring Algorithm on FPGA," International Journal of Computer and Electrical Engineering, vol. 5, no. 5, 2013, pp. 533-537

[4] Arpita Jena and Siba Ku Panda, "FPGA-VHDL Implementation of Pipelined Square root Circuit for VLSI Signal Processing Applications," International Journal of Computer Applications, Volume 142 – No.5, May 2016

[5] Fernando Martin del Campo, Alicia Morales-Reyes, Roberto Perez-Andrade, Rene Cumplido and Aldo G. Orozco-Lugo Claudia Feregrino, "A multi-cycle fixed point square Root and module for FPGAs," IEICE Electronics Express, Vol. 1 – No. 7, 2008, pp. 957-966

[6] Florent de Dinechin, Mioara Maria Joldes, Bogdan Pasca, and Guillaume Revy, "Multiplicative square root algorithms for FPGAs.," International Conference on Field Programmable Logic and Applications, Aug 2010, pp.14-17

[7] G. Anupama and A. Raghuram, "Novel Square Root Algorithm and its FPGA Implementation," International Journal of Engineering and Technical Research (IJETR), Vol. 3 (10), 2015, pp. 64-67

[8] Muhazam Mustapha, Burinieamman Jayabalan and Anis Shahida Niza Mokhtar, "Intel/Altera FPGA Implementation of CORDIC Square Root Algorithm," Journal of Computer Science & Computational Mathematics, Vol. 11, Issue 3, September 2021, pp. 52-56

[9] Palchaudhuri and Rajat Subhra Chakraborty, "High Performance Integer Arithmetic Circuit Design on FPGA: Architecture, Implementation and Design Automation," Springer, 2016

[10] Purna Ramesh Addanki, "Implementation of Integer Square Root," International Journal of Engineering Science and Innovative Technology (IJESIT), Vol. 4 (1), 2015, pp. 104-112

[11] S. Lachowicz and H-J. Pfleiderer, "Fast Evaluation of the Square Root and Other Nonlinear Functions in FPGA," IEEE International Symposium on Electronic Design, Test & Applications - DELTA, 2008, pp. 474-477

[12] Tole Sutikno, "An Optimized Square Root Algorithm for Implementation in FPGA Hardware," TELKOMNIKA, Vol. 8, No. 1, 2010, pp. 1 – 8

[13] Tole Sutikno, "An Efficient Implementation of the Non Restoring Square Root Algorithm in Gate Level," Inter-

national Journal of Computer Theory and Engineering, Vol.3, (1), 2012, pp. 46-51

[14] Tole Sutikno, Aiman Zakwan Jidin, Auzani Jidin and Nik Rumzi Nik Idris, “VHDL Coding of Modified Non-Restoring Square Root Calculator,” International Journal of Reconfigurable and Embedded Systems (IJRES) , Vol. 1 (1), 2012, pp. 37~42

[15] Vladitiu, M., Computer Arithmetic: Algorithms and Hardware Implementations, 2012

[16] Zhongcheng Zhou and Jingchun Hu, “A Novel Square Root Algorithm and its FPGA Simulation,” Journal of Physics: Conference Series, Volume 1314, 3rd International Conference on Electrical, Mechanical and Computer Engineering 9–11 August 2019, pp. 168-176