International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 15 (2014), pp. 1531-1537

http://www. irphouse.com

Design and FPGA Implementation of a Novel Square Root Evaluator based on Vedic Mathematics

Jaspreet Kaur [1] and Nirmal Singh Grewal [2]

[1]Student, Department of Electronics and Communication Engineering, Guru Nanak Dev Engineering College, Ludhiana, Punjab
[2]Executive Director, Science and Technology Entrepreneurs' Park, Guru Nanak Dev Engineering College, Ludhiana, Punjab

Abstract

Square root is a vital operation in many signal processing and computer graphics applications. The growing demand for faster computation of arithmetic operations leads to the implementation of square root on hardware, but the complexity of the associated algorithms is a limiting factor. This paper focuses on the designing of a square root algorithm based on ancient Indian Vedic mathematics sutras. Vedic mathematics is well known for its formulae yielding faster results, both for mental calculation and hardware design. The square root for an 8-bit radicand has been implemented on Xilinx Spartan-3E FPGA. The synthesis results indicate a cost of only 26 LUTs and a latency of 12.879ns which is quite efficient in terms of area and speed as compared to conventional methods. The proposed work can be easily extended for higher radix square root implementations.

Keywords: Duplex, FPGA, square root, Vedic mathematics

Introduction

Square root is a very useful and necessary operation in scientific computations. It holds importance in many computer graphics and signal processing applications [1]. Earlier the square root operation was performed by software means on the processors causing longer delays in the completion of the task. The traditional methods for computing the square root include Rough estimation, Babylonian method, Digit-by-digit calculation, Exponential identity, Goldschmidt's algorithm and Taylor series method.

The hardware implementation of square root operation also sounds promising, but because of their algorithmic complexity they are difficult to implement on Field

Programmable Gate Arrays (FPGAs) [2]. This research work focuses on evaluation of square root by a method based on ancient Indian literature, the Vedic Mathematics. Vedic mathematics, written by Sri Bharati Krishna Tirthaji Maharaja, contains sixteen simple mathematical formulae from the Vedas. It deals with carrying out cumbersome mathematical operations to execute them mentally or to find results in a single line [3]. It yields faster results for hardware design also. Thus, adopting Vedic methods to implement mathematical operations can be efficient.

Proposed Vedic square root algorithm

In this work, the authors used Vedic algorithm to compute the square root of an 8-bit radicand.

Basic Vedic bit-by-bit algorithm.

Vedic mathematics offers a bit-by-bit algorithm for calculating the square root of a number. It makes use of the *Dwandwa Yoga* or *Duplex operation*, which is a vedic method of finding square of a number [4]. The method to find Duplex of a binary number has been illustrated as follows –

Let D denotes Duplex. Then for a 1-bit number, D is the number itself.

$D(1) = 1$ or $D(0) = 0$

For a 2-bit number, D is twice the product of both the bits.

$D(11) = 2 * 1 * 1 = 2$ or $(10)_2$

For a 3-bit number, D is twice the product of outermost pair of bits plus the middle bit.

$D(101) = (2 * 1 * 1) + 0 = 2$ or $(10)_2$

For a 4-bit number, D is twice the product of outermost pair of bits plus twice the product of next pair of bits.

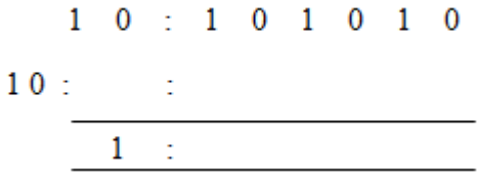
$D(1110) = (2 * 1 * 0) + (2 * 1 * 1) = 2$ or $(10)_2$

Note:

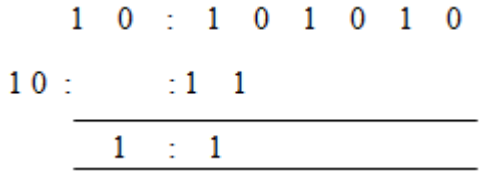
The multiplication of two single bit numbers reduces to *logical and* between them.

The bit-by-bit Vedic algorithm for computing square root is illustrated by means of an example. The actual method is written for decimal numbers, but here its binary extension has been presented.

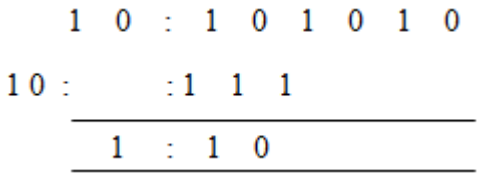
Let the number whose square root is to be calculated is 170 (binary 1010 1010). Group the bits in pairs, starting from the LSB. The nearest square root for the MSB pair 10 is 1. Thus, 1 is written in first place as the MSB quotient bit. Now, twice of this bit, i.e. 10, will serve as the divisor for rest of the process which is kept in the left of the middle row.



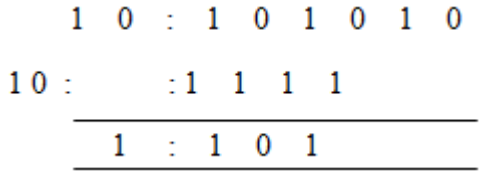
Next, the square of first quotient bit is subtracted from 10 leaving 1 as remainder, which is appended to the left of the next input bit 1. Now, 11 becomes the new gross dividend. Divide 11 by the divisor. This gives 1 as the result, which becomes the second quotient bit, and 1 as the remainder, which is appended to the next input bit 0.



Now, 10 becomes the new gross dividend. To obtain the actual dividend, the Duplex of all the previous quotient bits obtained so far, but the first bit, is computed and subtracted from the gross dividend. So, here the Duplex of 1 is subtracted from 10. This gives 1 as the result which now becomes the actual dividend. Dividing this actual dividend with the divisor yields 0 as the result, which is the third quotient bit, and 1 as the remainder, which is appended to the left of the next input bit 1.



Now, 11 becomes the new gross dividend. From 11, the Duplex of 10 is subtracted, which yields 11 as the result itself. Therefore, 11 becomes the new actual dividend. Dividing 11 with the divisor gives 1 as the result, which is the fourth and the final quotient bit, and 1 as the remainder.



In this way, we can obtain the square root of any higher radix number also. In the above example, 1 is left as the remainder, which can be further appended to the left of the next input bit and the process can be continued up to the last input bit or even further to obtain the bits contributing to the fractional part of the result. An illustration has been shown below –

```
      1  0 : 1  0  1  0  1  0
10 :     : 1  1  1  1  0  1
      ---------------------
      1  : 1  0  1 . 0  0  0
      ---------------------
```

Thus, we obtained the square root of 170 (binary 1010 1010) as 13 (binary 1101) in a single line by following the Vedic algorithm.

Modified bit-by-bit algorithm.

The proposed algorithm presented in Fig. 1 differs from the basic bit-by-bit algorithm in a sense that the procedure followed for individual bits in the basic algorithm is here adapted for a group of bits.

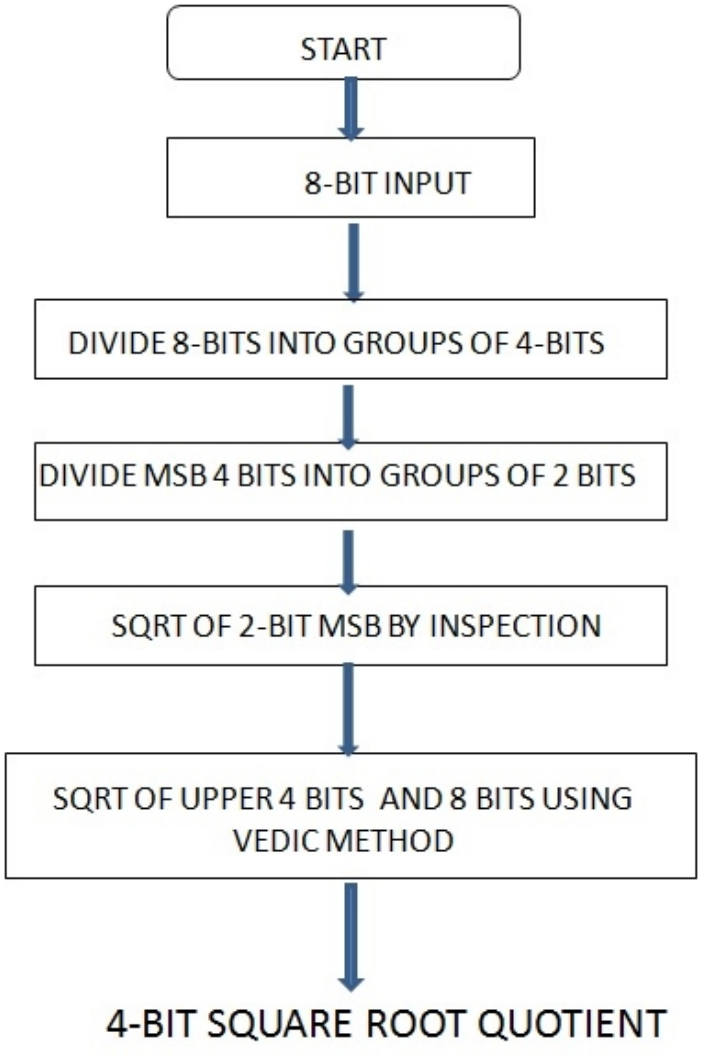
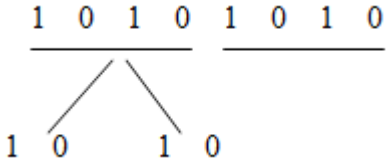
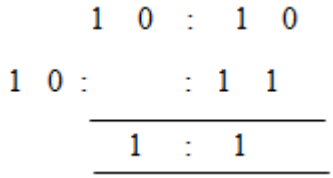


Fig. 1 Proposed square root algorithm

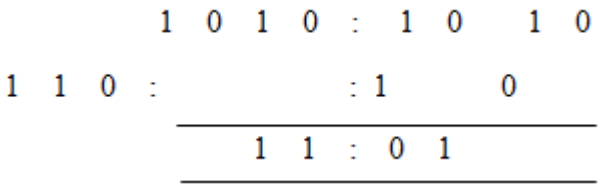
The square root of 170 is obtained again using the proposed algorithm to illustrate the process. The input 8-bit binary number is divided into two groups of 4 bits each. The first group of 4 bits is further divided into two groups of 2 bits each.



Starting with the first pair of bits 10, its square root 1 is calculated by inspection. Then, the basic bit-by-bit method is used to find the 2-bit square root of the 4-bit binary number 1010.



Now, the square root of 1010, i.e. 11, is used to find the square root of upper 8 bits. Here, 11 becomes the first pair of bits of the desired 4-bit quotient.



In this case, twice of 11, i.e. 110, serves as the divisor. The square of 11, i.e. 1001, is subtracted from 1010, and the result of subtraction, i.e. 1, is appended to next pair of the input bits 10. Therefore, 110 becomes the gross dividend, which is divided by the divisor and gives 01 as the result and 0 as the remainder. Now, 01 becomes the second pair of bits of the desired 4-bit quotient.

Results and discussion

The square root evaluator for an 8-bit radicand has been designed, implemented, synthesized, and simulated using the Xilinx ISE 8.2i tool and mapped onto a Xilinx Spartan 3E xc3s100e-5cp132 FPGA [5]. The algorithm has been coded in Verilog HDL at the behavioral level of abstraction. It is observed that for an 8-bit input, the maximum combinational path delay is 12.879 ns with a minimal look-up-table (LUT) utilization of 26 out of 1,920 available. The proposed work shows an improvement in performance over the previously reported work [1]. Table 1 shows a brief synthesis summary.

Table 1 FPGA design summary

Number of Slices	15
Number of 4 input LUTs	26
Number of bonded IOBs	12
Maximum combinational path delay	12.879 ns

The simulation has been done using the ISE simulator. The simulation result of the square root of 170 is shown in Fig. 2.

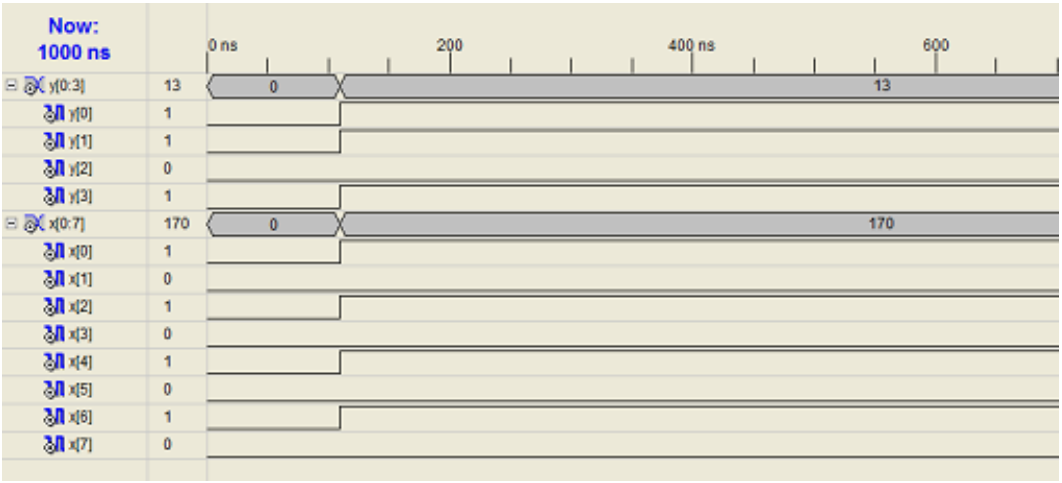


Fig. 2 Simulation result of the proposed work

Conclusion

Square root operation is an important function and many processors nowadays implement it on hardware. In this paper the authors presented a novel square root algorithm which is based on Vedic mathematics. The idea behind using the Vedic methods for the design of various arithmetic circuits is their simplicity and efficiency in speed, both for mental calculations and hardware implementation. The proposed work results indicate a minimal logic utilization and faster computation. This work can be easily extended to more number of input bits, thus proving the excellence of the Vedic algorithms.

References

[1] Samavi S, Sadrabadi A, Fanian A. *Modular array structure for non-restoring square root circuit.* Jounal of Systems Architecture 2008; 54(10):957-966.

[2] Putra R.V.W. *A novel fixed-point square root algorithm and its digital hardware design.* International Conference on ICT for smart society 2013; Jakarta, Indonesia:1-4.
[3] Maharaja J.S.S.B.K.T. *Vedic mathematics.* Delhi: Motilal Banarsidass Publishers Pvt. Ltd 1992.
[4] Thapliyal H, Kotiyal S, Srinivas M. *design and analysis of a novel parallel square and cube architecture based on ancient Indian vedic mathematics.* 48th Midwest Symposium on circuits and systems 2005; 2:1462-1465.
[5] http://www.xilinx.com