Advances in Pure Mathematics, 2015, 5, 428-441
Published Online June 2015 in SciRes. http://www.scirp.org/journal/apm
http://dx.doi.org/10.4236/apm.2015.58042

Scientific Research Publishing

High Performance Novel Square Root Architecture Using Ancient Indian Mathematics for High Speed Signal Processing

Arindam Banerjee, Aniruddha Ghosh, Mainuck Das

Department of ECE, JIS College of Engineering, Kalyani, India
Email: banerjee.arindam1@gmail.com, ghoshani19@gmail.com, mrmainuckdas@gmail.com

Received 22 April 2015; accepted 8 June 2015; published June 2015

Abstract

Novel high speed energy efficient square root architecture has been reported in this paper. In this architecture, we have blended ancient Indian Vedic mathematics and Bakhshali mathematics to achieve a significant amount of accuracy in performing the square root operation. Basically, Vedic Duplex method and iterative division method reported in Bakhshali Manuscript have been utilized for that computation. The proposed technique has been compared with the well known Newton-Raphson's (N-R) technique for square root computation. The algorithm has been implemented and tested using Modelsim simulator, and performance parameters such as the number of lookup tables, propagation delay and power consumption have been estimated using Xilinx ISE simulator. The functionality of the circuitry has been checked using Xilinx Virtex-5 FPGA board.

Keywords

Vedic Mathematics, Bakhshali Mathematics, Duplex, Yavadunam Sutra

1. Introduction

Arithmetic circuits are indispensable in DSP and image processing applications. Many researchers have already designed different arithmetic circuits like adder, subtractor, multiplier, divider, squarer, square root architecture etc. [1]-[9], but those techniques exhibit more propagation delay and power consumption. Square root design is a very challenging work in modern research area. Newton-Raphson's (N-R) method of square root computation has been the most efficient approach so far. After N-R method, there has been no remarkable approach for square

How to cite this paper: Banerjee, A., Ghosh, A. and Das, M. (2015) High Performance Novel Square Root Architecture Using Ancient Indian Mathematics for High Speed Signal Processing. *Advances in Pure Mathematics*, **5**, 428-441.
http://dx.doi.org/10.4236/apm.2015.58042

root computation so far. We are showing an efficient technique using ancient Indian mathematics.

In ancient times, mathematicians made calculations based on 16 Sutras (Formulae). The idea for designing the square root processor has been adopted from ancient Indian mathematics "Vedas" [10] [11]. The well known duplex method is an ancient Indian method of extracting the square root. The algorithm incorporates digit by digit method for calculating the square root of a whole or decimal number one digit at a time. As per the proposed algorithm, the square root of any number is obtained in one step which reduces the iterations. Bakhshali method is used for finding a proximal square root described in ancient Indian mathematical manuscript called the Bakhshali manuscript. A prototype of 16 bit square root processors has been implemented and their functionality has been experimented on a Virtex-5 FPGA board. For calculating integer part of a number, we have used the Vedic method and for calculating the floating point part, we have adopted Bakhshali method. Many researchers have used the most popular Newton-Raphson's method for computing square root of a number. In our methodology, we are combining both Vedic duplex and Bakhshali approximation to generate more accurate result with comparatively less time and power consumption.

The paper has been organized as follows. Section 2 describes the back ground mathematics for calculating the square root. Section 3 gives the architectural description of the proposed technique. Section 4 describes the error calculation and accuracy analysis of the proposed technique. Result analysis has been described in Section 5 and Section 6 is the conclusion.

2. Square Root Algorithm

Square root technique using Division method was introduced by ancient Indian Mathematicians and the procedure has been lucidly discussed in "Vedas". The technique stands upon the execution by mere observation. Square root of a perfect square can be easily determined by the Division method. But for those numbers, which are not perfect square, Division method is not enough to compute with high precision. In this paper an iterative method described in Bakhshali Mathematics of Quran has been proposed to achieve high speed and high precision square root calculation. The mathematical formulation of the iterative approach is shown below.

Consider a number X whose square root is to be calculated. The Bakhshali manuscript approach computes the square root of the perfect square which is nearest but less than the number X by the method of observational inspection. Consider A is the above mentioned perfect square whose square root is R.

So it can be expressed as

$$X = A \pm Y = R^2 \pm Y \tag{1}$$

where Y is the residue.

Equation (1) can be reformulated as

$$X = R^2\left(1 \pm \frac{Y}{R^2}\right) \tag{2}$$

So the square root of X can be written as

$$X^{1/2} = R\left(1 \pm \frac{Y}{R^2}\right)^{1/2} \tag{3}$$

Using binomial expansion and the mathematical formulae of Bakhshali manuscript, Equation (3) can be expressed as

$$X^{1/2} \cong R\left[1 \pm \frac{Y}{2R^2} - \frac{1}{8}\left(\frac{Y}{R^2}\right)^2 \pm \frac{1}{16}\left(\frac{Y}{R^2}\right)^3 - \frac{1}{32}\left(\frac{Y}{R^2}\right)^4 \pm \cdots\right] \tag{4}$$

$$= R\left[1 \pm \frac{Y}{2R^2} - \frac{1}{2R^2}\left(\frac{Y}{2R}\right)^2 \pm \frac{1}{2R^2} \times \frac{1}{R}\left(\frac{Y}{2R}\right)^3 - \frac{1}{2R^2} \times \frac{1}{R^2}\left(\frac{Y}{2R}\right)^4 \pm \cdots\right] \tag{5}$$

$$= R\left[1 \pm \frac{Y}{2R^2} - \frac{1}{2R^2}\left(\frac{Y}{2R}\right)^2 \left\{1 \mp \frac{Y}{2R^2} + \left(\frac{Y}{2R^2}\right)^2 \mp \cdots\right\}\right] \tag{6}$$

$$= R\left[1 \pm \frac{Y}{2R^2} - \frac{1}{2R^2}\left(\frac{Y}{2R}\right)^2\left(1 \pm \frac{Y}{2R^2}\right)^{-1}\right] \tag{7}$$

$$= R\left[1 \pm \frac{Y}{2R^2} - \frac{\left(\frac{Y}{2R}\right)^2}{2R^2\left(1 \pm \frac{Y}{2R^2}\right)}\right] \tag{8}$$

$$= R \pm \frac{Y}{2R} - \frac{\left(\frac{Y}{2R}\right)^2}{2R\left(1 \pm \frac{Y}{2R^2}\right)} \tag{9}$$

$$= R \pm \frac{Y}{2R} - \frac{\left(\frac{Y}{2R}\right)^2}{2\left(R \pm \frac{Y}{2R}\right)} \tag{10}$$

Equation (10) is the modified Bakshali expression for computing the square root of non-squared numbers.

In modified Bakshali approach the accuracy is more as comparable to in Bakshali approach which is shown in the table. So in respect of accuracy modified Bakshali approach is beneficial. Also Bakshali methodology contains some drawbacks mentioned below:

1) Bakshali approach does not gives the idea for computing the nearest squared number and its squareroot. To obtain thenearest square number and its square root we have to go for Vedic approach.

2) The division method for calculating the square root using Vedic mathematics is based on "Vilokanam" (Inspection) method which incorporates an extra squaring and extra subtraction operations.

The modified Bakshali approach eliminates the extra squaring and subtraction operations. thus the technique reduces the stage delay.

The square root determination technique combining the Vedic Duplex and Bakhshali approach is shown in the flow chart of **Figure 1**.

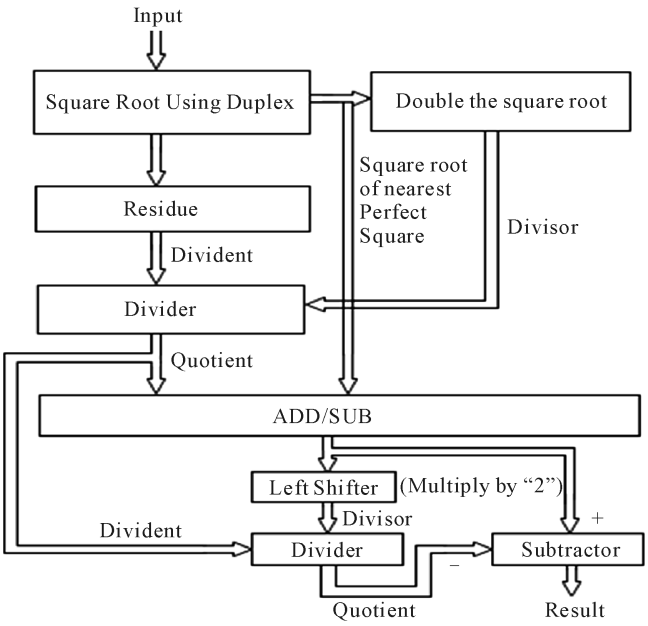

Figure 1. Architecture for square root determination technique.

2.1. Integral Square Root Determinant Using Vedic Methodology

Mathematical Modeling of the Algorithm to Calculate Perfect Square Root

Consider X is an N bit number whose square root is to be determined. X can be expressed as

$$X = \sum_{N-1}^{i=0} x_i 2^i = 2^{N/2} \sum_{i=0}^{\frac{N}{2}-1} x_{i+N/2} 2^i + \sum_{i=0}^{\frac{N}{2}-1} x_i 2^i \tag{11}$$

$$= 2^{N/2} \sum_{i=0}^{\frac{N}{2}-1} y_{i+N/2} 2^i + 2^{N/2} \sum_{i=0}^{\frac{N}{2}-1} \left(x_{i+\frac{N}{2}} - y_{i+\frac{N}{2}} \right) 2^i + \sum_{i=0}^{\frac{N}{2}-1} x_i 2^i, \tag{12}$$

Considering $\sum_{i=0}^{\frac{N}{2}-1} y_{i+N/2} 2^i$ to be the nearest perfect square number of $\sum_{i=0}^{\frac{N}{2}-1} x_{i+N/2} 2^i$. Now assume that the square root of $\sum_{i=0}^{\frac{N}{2}-1} y_{i+N/2} 2^i$ is $Z = 2^{N/4} \sum_{i=0}^{\frac{N}{4}-1} z_{i+N/4} 2^i$. Then, Equation (12) can be rewritten as

$$X = Z^2 + 2Z \left(\frac{2^{\frac{N}{2}} \sum_{i=0}^{\frac{N}{2}-1} \left(x_{i+\frac{N}{2}} - y_{i+\frac{N}{2}} \right) 2^i + \sum_{i=0}^{\frac{N}{4}-1} x_{i+n/4} 2^i}{2Z} \right) + \sum_{i=0}^{\frac{N}{4}-1} x_i 2^i \tag{13}$$

Now consider

$$\left(\frac{2^{\frac{N}{2}} \sum_{i=0}^{\frac{N}{2}-1} \left(x_{i+\frac{N}{2}} - y_{i+\frac{N}{2}} \right) 2^i + \sum_{i=0}^{\frac{N}{4}-1} x_{i+n/4} 2^i}{2Z} \right) = Q + \frac{R}{2Z}, \tag{14}$$

where Q is the quotient and R is the remainder. Inserting Equation (14), Equation (13) can be reformulated as

$$X = Z^2 + 2ZQ + \left(R + \sum_{i=0}^{\frac{N}{4}-1} x_i 2^i \right) \tag{15}$$

Equation (15) can be rewritten as

$$\begin{aligned} X &= Z^2 + 2ZQ + Q^2 + \left(R + \sum_{i=0}^{\frac{N}{4}-1} x_i 2^i - Q^2 \right) \\ &= (Z+Q)^2 + \left(R + \sum_{i=0}^{\frac{N}{4}-1} x_i 2^i - Q^2 \right) \end{aligned} \tag{16}$$

Assume $R + \sum_{i=0}^{\frac{N}{4}-1} x_i 2^i - Q^2 = p$. Then X can be represented as,

$$X = (Z+Q)^2 + p \tag{17}$$

$$\sqrt{X} = \sqrt{(Z+Q)^2 + p} = \sqrt{(Z+Q)^2 \left(1 + \frac{p}{(Z+Q)^2} \right)} \tag{18}$$

$$= (Z+Q) \sqrt{\left(1 + \frac{p}{(Z+Q)^2} \right)} \tag{19}$$

$$= (Z+Q) \pm \frac{p}{2(Z+Q)} - \frac{\left(\frac{p}{2(Z+Q)}\right)^2}{2\left((Z+Q) \pm \frac{p}{2(Z+Q)}\right)} \tag{20}$$

The procedure to calculate the square root by division method can be described in the following steps:

Step 1: Obtain the nearest square root of the $N/2$ Most Significant Bits (MSB). Assume that the output is Z.

Step 2: Determine the square of Z by combining Yavadunam and Duplex methodology.

Step 3: Subtract the squared output from $N/2$ MSB.

Step 4: Obtain the double of Z.

Step 5: Combine the output of the subtractor and the next $N/4$ bits. Divide the combination by $2Z$. Assume the quotient as Q and the remainder as R.

Step 6: Determine the square of Q and subtract $Q2$ from $R + \sum_{i=0}^{\frac{N}{4}-1} x_i 2^i$. If the residue ($p$) is zero then X is the perfect square number whose square root is $(Z + Q)$ otherwise $(Z + Q)$ is the square root of the perfect square nearest to X. Again if "p" is positive then the nearest perfect square number is less than the given input else if "p" is negative then the nearest perfect square number is greater than the input.

Step 7: Compute $\left|\frac{p}{2(Z+Q)}\right|$ by Straight Division ("Dhvajanka") methodology.

Step 8: Determine the square of $\frac{p}{2(Z+Q)}$.

Step 9: Compute $\left((Z+Q) \pm \frac{p}{2(Z+Q)}\right)$ by add/sub unit.

Step 10: Finally subtract $\frac{\left(\frac{p}{2(Z+Q)}\right)^2}{2\left((Z+Q) \pm \frac{p}{2(Z+Q)}\right)}$ from $(Z+Q) \pm \frac{p}{2(Z+Q)}$ to obtain the exact square root.

Step 11: Compute the maximum power of first left most "1" of $(Z+Q) \pm \frac{p}{2(Z+Q)} - \frac{\left(\frac{p}{2(Z+Q)}\right)^2}{2\left((Z+Q) \pm \frac{p}{2(Z+Q)}\right)}$ by left shifting operation.

3. Proposed Architecture

Square Root architecture following the methodology given in Step 1 to Step 6, consists of four sub sections: 1) Nearest Square Root Generation Unit (NSRGU) for first $N/2$ numbers; 2) Squaring Unit; 3) Sub-tractor; 4) Divider. **Figure 2** exhibits a fully optimized system level architecture for square root generation using Duplex method adopted by ancient Indian mathematicians. In this architecture all the left shifters are N/4 bit left shifters. Quotient register is used to store the fixed point or integral result of the square root. The most significant two bits fed to the doubler whose output is fed to the subtraction unit.

3.1. Nearest Square Root Generation Unit

In ancient Indian Mathematics, the nearest square number for a given number and its square root was calculated by the method of inspection. In this paper an easy and efficient architecture has been provided to compute the nearest square root generation. It is obvious that $N = N/2$ in "Equation (12)", the number bits in square root

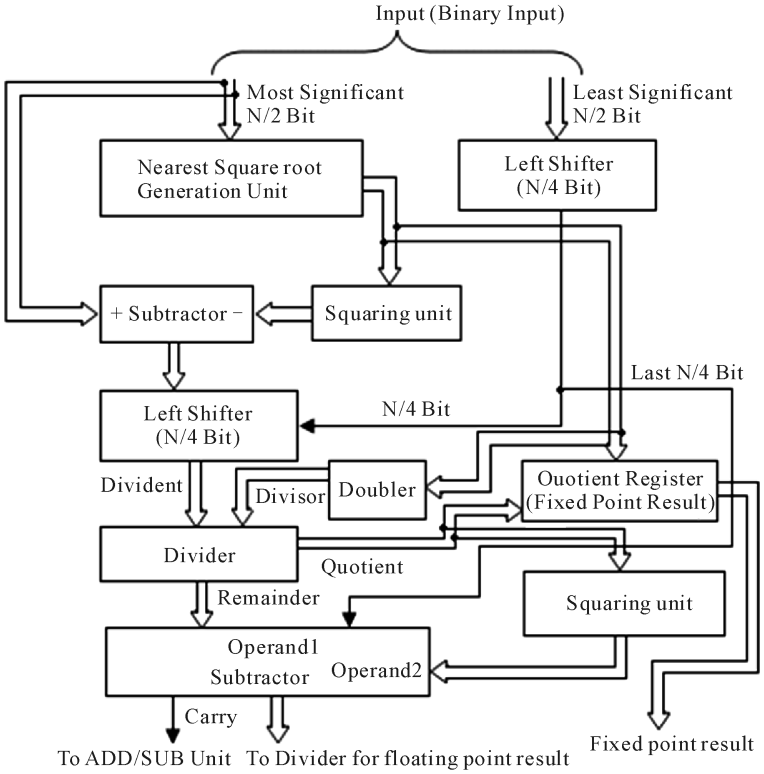

Figure 2. System level architecture for Integral part of Square root generator, *N* indicating number of input bits.

must be of *N*/4 bits. The architecture of the nearest square root generation unit is same as the figure depicted in **Figure 2**. The above mentioned Step 1 to Step 6 are needed to obtain the nearest square root for *N*/2 bit number.

3.2. Squaring Unit

Squaring algorithm and the corresponding architecture was implemented with the aid of "Yavadunam Sutra".

Mathematical Formulation of Yavadunam Sutra

If a number X is between $2n-1$ and $2n$ then the average of $2n-1$ and $2n$ is $A=\frac{2^{n-1}+2^n}{2}=3\times 2^{n-2}$. If $X>A$ then $2n$ is chosen as radix and if $X\le A$ then $2n-1$ is the selected radix.

In Binary mathematics, the number $X=\sum_{i=0}^{n-1}x_i 2^i$ can be reformulated as,

$$X=x_{n-1}2^{n-1}+\sum_{i=0}^{n-2}x_i 2^i,\ \text{when Radix}=2n-1 \tag{21}$$

$$X=2^n-\left(2\text{'s complement of } X\right),\ \text{when Radix}=2n \tag{22}$$

From Equation (21), the square of the number X can be expressed as,

$$X^2=\left(x_{n-1}2^{n-1}\right)^2+2\left(x_{n-1}2^{n-1}\right)\sum_{i=0}^{n-2}x_i 2^i+\left(\sum_{i=0}^{n-2}x_i 2^i\right)^2 \tag{23}$$

$$=2^{n-1}\left(X+\sum_{i=0}^{n-2}x_i 2^i\right) \tag{24}$$

Similarly, the expression of X^2 that can be obtained from Equation (22) is given as,

$$X^2=2^n\left(X-2\text{'s complement of } X\right)+\left(2\text{'s complement of } X\right)^2 \tag{25}$$

Equation (24) and Equation (25) are the mathematical formulations of Yavadunam Sutra in Binary mathematics.

The architecture of squaring algorithm using "Yavadunam Sutra" is shown in **Figure 3**. The basic building blocks of the architecture are 1) RSU, 2) Subtractor, 3) Add-Sub unit and 4) Duplex squaring architecture. The test bench waveform of RSU is shown in **Figure 4**. The Subtractor architecture using "Nikhilam" sutra has been elucidated in sub-section-(C).

The input bits X (of length n) is fed to the RSU. The RSU produces proper radix and exponent for the input. The radix is subtracted from the input and the result is added to or subtracted from the input again depending upon the control signal generated from the borrow output of the sub-tractor. The output of add/sub unit is the residue which is again squared by duplex operation discussed in [1] and it is also fed to the left shifter which shift the data depending upon the exponent value. Finally the outputs of duplex squarer and left shifter are added to obtain the squared result.

3.3. Subtractor

Mathematical Modeling of "Nikhilam" Sutra for Binary Subtraction

The Subtraction method has been implemented using normal binary arithmetic. It is a bit wise subtraction method. The rule is mathematically expressed in Equation (26). Consider the number, $Y = \sum_{n-1}^{n-1} y_i 2^i$ is to be subtracted from the number $X = \sum_{i=0}^{n-1} x_i 2^i$. So, it can be written as

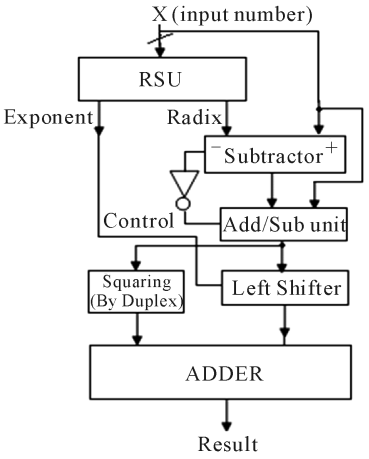

Figure 3. Architecture of squaring algorithm using "Yavadunam" sutra.

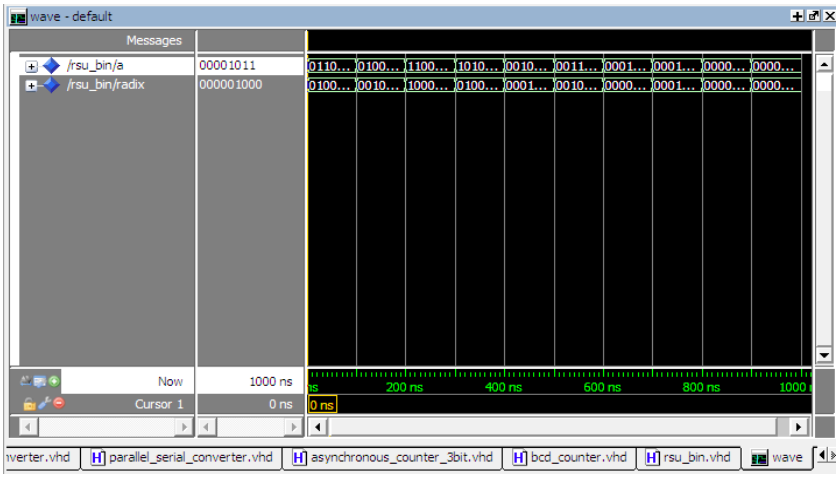

Figure 4. Test bench window for VHDL implementation of RSU.

$$X - Y = \sum_{i=0}^{n-1}\left(x_i 2^i - y_i 2^i\right) \tag{26}$$

There is a borrow bit generated at the end of operation which signifies that the result is positive or negative.

3.4. Divider Using "Dhvajanka" Sutra ("On Top of the Flag")

3.4.1. Mathematical Modeling of Division Operation

Consider the number $A = \sum_{i=0}^{n/2-1} a_i x^i$ to be divided by $B = \sum_{i=0}^{n/2-1} b_i x^i$ assuming that both the numbers are of same length (here the length is $(n/2) - 1$) and x is the radix. To execute the division operation easily and efficiently by "Dhvajanka" sutra ("On top of the flag") methodology described in the ancient Indian Vedic Mathematics, Dividend must have greater length than Divisor. Consider again a number A' which can be represented as $A' = x^{\frac{n}{2}} \times A = x^{\frac{n}{2}} \times \sum_{i=0}^{n/2-1} a_i x^i = \sum_{i=0}^{n-1} a_i x^i$, $a_i\left(\forall i \in \left\{0,1,2,\cdots,\left(\frac{n}{2}\right)-1\right\}\right) = 0$. $B = \sum_{i=0}^{n/2-1} b_i x^i$ can be mapped with $B' = \sum_{i=0}^{\frac{n}{2}-1} b_i' x^i$ where the condition of mapping is $b_i = b_i' \forall i \in \left\{0,1,2,\cdots,\left(\frac{n}{2}\right)-1\right\}$. The quotient Q can be determined as

$$Q = \frac{A}{B} = \frac{A' \times x^{\left(-\frac{n}{2}\right)}}{B'} = \left(\frac{A'}{B'}\right) \times x^{\left(-\frac{n}{2}\right)} \tag{27}$$

The term $x^{\left(-\frac{n}{2}\right)}$, where x is the radix is to be extracted by Exponent Extraction Unit (EEU). In Binary number system, the radix "x" is equal to 2. Then from Equation (27), it is obvious that the result of $\frac{A}{B}$ is needed to be shifted right by $n/2$ terms which has been shown in **Figure 5**, to get the actual quotient. The mathematical modeling of division operation by "Vertically and Crosswise" methodology has been described in the following subsection.

The architecture shown in **Figure 6** consists of some elementary building blocks like left shifter, right shifter, incrementer and demultiplexer. Incrementer calculates the exponent of the number which is controlled by the shifted bit. The clock applied to the shifter is also controlled by the shifted bit. If the shifted bit is "1" then the shifters stops further shifting.

3.4.2. Mathematical Modeling of "On Top of the Flag" Sutra

Consider the number $A' = \sum_{i=0}^{n-1} a_i' x^i$ is divided by $B' = \sum_{i=0}^{\frac{n}{2}-1} b_i' x^i$. So A' can be expressed in terms of B' as

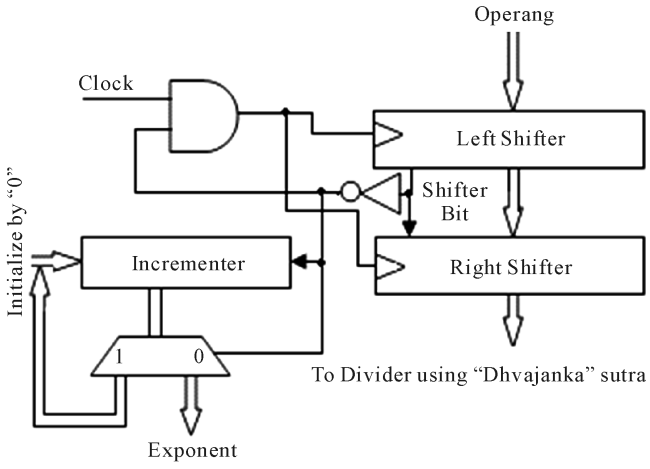

Figure 5. RTL representation of exponent extraction unit.

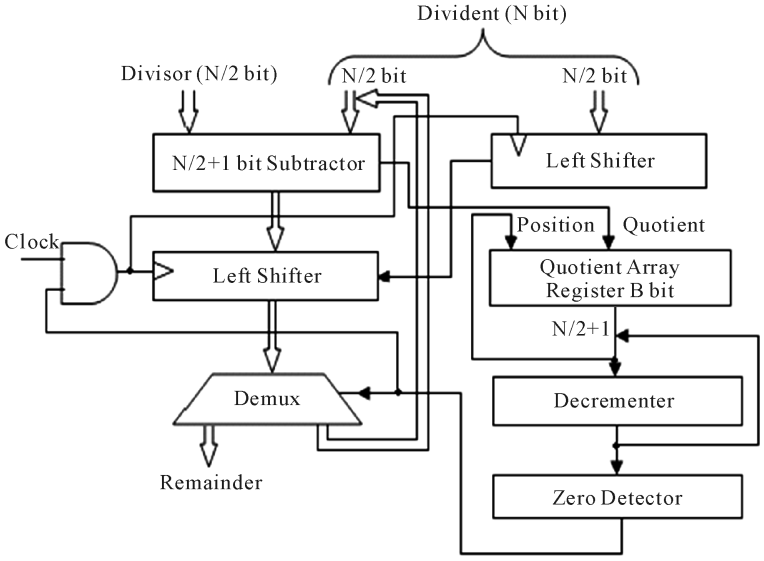

Figure 6. Divider architecture using "Dhvajanka" sutra.

$$A' = \sum_{i=0}^{n-1} a'_i x^i = a'_{n-1}x^{n-1} + a'_{n-2}x^{n-2} + a'_{n-3}x^{n-3} + a'_{n-4}x^{n-4} + \cdots + a'_3x^3 + a'_2x^2 + a''_1x^1 + a'_0x^0 \tag{28}$$

$$= \left(b'_{\frac{n}{2}-1} x^{\frac{n}{2}-1} + b'_{\frac{n}{2}-2} x^{\frac{n}{2}-2} + \cdots + b'_0 x^0 \right)$$

$$\times \left(\frac{a'_{n-1}}{b'_{\frac{n}{2}-1}} x^{\frac{n}{2}} + \frac{\left(a'_{n-2} - \left(\frac{a'_{n-1}}{b'_{\frac{n}{2}-1}} \right) b'_{\frac{n}{2}-2} \right)}{b'_{\frac{n}{2}-1}} x^{\frac{n}{2}-1} + \frac{\left(a'_{n-3} - \left(\frac{a'_{n-1}}{b'_{\frac{n}{2}-1}} \right) b'_{\frac{n}{2}-3} - \frac{\left(a'_{n-2} - \left(\frac{a'_{n-1}}{b'_{\frac{n}{2}-1}} \right) b'_{\frac{n}{2}-2} \right) \times b'_{\frac{n}{2}-2}}{b'_{\frac{n}{2}-1}} \right)}{b'_{\frac{n}{2}-1}} x^{\frac{n}{2}-2} + \cdots \right) \tag{29}$$

Here "x" signifies the radix.

Figure 6 shows the architectural description of division operation using "On top of the flag" Sutra. The Dividend of N bits is divided into two equal $N/2$ bits. The most significant part is fed to the subtractor which is of $N/2 + 1$ bits size. Single bit zero padding is done in subtractor module to the left. The division procedure incorporated here is non-restoring type of division. The carry output after the subtraction is fed to the quotient array register as quotient.

The quotient array register stores the quotients based on the iterated value which is used as position signal. The value $N/2 + 1$ is stored in the decrementer initially to count and check the specified iteration. After each iteration, both the left shifters are updated by shifting until the decremented value reaches zero.

The Divider architecture combining the exponent extraction and the "Dhvajanka" Sutra is shown in **Figure 7**. The basic building blocks of the architecture are 1) Exponent Extraction Unit, 2) Divider using "Dhvajanka"

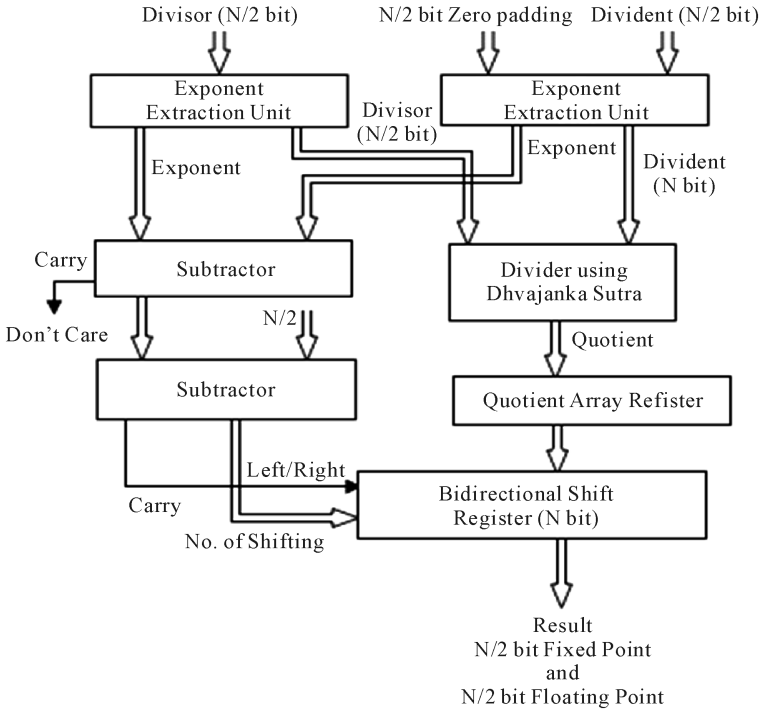

Figure 7. Divider architecture combining exponent extraction and "Dhvajanka" sutra.

sutra, 3) Subtractor, 4) Quotient Array Register and 5) Bidirectional Shift Register. Zero padding is needed to represent a number to higher number of bits. For example if a four bit number "0111" is represented in eight bit format then the representation becomes "00000111". Here, zero padding has been executed based on the requirement. Bidirectional shift register performs the operation of shifting both to the left or right depending upon the control signal "carry" from the second subtractor. If carry = 0 then the input bits are shifted to the left and if carry = 1 then input bits are shifted to the right. The result of the second subtractor is fed to the shifter to control number shifting. The contents of the Quotient Array Register are the input to the Bidirectional Shift register.

Illustration: Consider a fourth order function $f(x)=a_3x^3+a_2x^2+a_1x^1+a_0x^0$ and a second order function $g(x)=b_1x^1+b_0x^0$. We have to compute $\frac{f(x)}{g(x)}$ with the help of "On top of the flag" sutra. Now,

$$\frac{f(x)}{g(x)}=\frac{a_3x^3+a_2x^2+a_1x^1+a_0x^0}{b_1x^1+b_0x^0}=Q(x)+\frac{R}{g(x)}\Rightarrow f(x)=Q(x)g(x)+R$$

where, $Q(x)=\left(\frac{a_3}{b_1}x^2+\frac{\left(a_2-\frac{a_3}{b_1}b_0\right)}{b_1}x^1+\frac{\left(a_1-\frac{\left(a_2-\frac{a_3}{b_1}b_0\right)}{b_1}b_0\right)}{b_1}x^0\right)$ and $R=a_0-\frac{\left(a_1-\frac{\left(a_2-\frac{a_3}{b_1}b_0\right)}{b_1}b_0\right)}{b_1}b_0$

3.4.3. Adder-Subtractor Unit Using "Nikhilam" Sutra for Subtraction

From Equation (5), it is obvious that to achieve the final result an Adder-Subtractor unit is required which has

been designed with the help of well-known binary addition/subtraction methodology. The architecture of Adder-Subtractor unit is shown in **Figure 8**. The carry signal from the square root determinant shown in **Figure 2** is fed to the adder-subtractor unit at the control pin to assign the type of operation (addition/subtraction). If the carry signal is low then addition operation is executed else subtraction operation is executed.

4. Error Calculation and Accuracy Analysis

The exact expression for square root can be recapitulated from Equation (3) as,

$$X^{1/2} = R\left(1 \pm \frac{Y}{R^2}\right)^{1/2} = R\left[1 \pm \frac{Y}{2R^2} - \frac{1}{8}\left(\frac{Y}{R^2}\right)^2 \pm \frac{1}{16}\left(\frac{Y}{R^2}\right)^3 - \frac{5}{4\times 32}\left(\frac{Y}{R^2}\right)^4 \pm \frac{7}{4\times 64}\left(\frac{Y}{R^2}\right)^5 - \cdots\right] \tag{30}$$

The approximated expression for the square root can be expressed as,

$$X^{1/2} \cong R\left[1 \pm \frac{Y}{2R^2} - \frac{1}{8}\left(\frac{Y}{R^2}\right)^2 \pm \frac{1}{16}\left(\frac{Y}{R^2}\right)^3 - \frac{1}{32}\left(\frac{Y}{R^2}\right)^4 \pm \frac{1}{64}\left(\frac{Y}{R^2}\right)^5 - \cdots\right] \tag{31}$$

So the computational error (ec) in determining the square root can be expressed as,

$$e_c = \left(X_{exact}^{\frac{1}{2}} - X_{approximated}^{\frac{1}{2}}\right) \cong R \times \frac{1}{32}\left(\frac{Y}{R^2}\right)^4 \left[1 \pm \frac{1}{2}\left(\frac{Y}{R^2}\right)\right]^{-1} \tag{32}$$

The percentage of error in calculating the square root is,

$$\begin{aligned} e_c(\%) &\cong \left(\frac{R \times \frac{1}{32}\left(\frac{Y}{R^2}\right)^4 \left\{1 \pm \frac{1}{2}\left(\frac{Y}{R^2}\right)\right\}^{-1}}{R\left(1 \pm \frac{Y}{R^2}\right)^{\frac{1}{2}}}\right) \times 100 \\ &= \left(\frac{\frac{1}{32}\left(\frac{Y}{R^2}\right)^4}{\left\{1 \pm \frac{1}{2}\left(\frac{Y}{R^2}\right)\right\} \times \left(1 \pm \frac{Y}{R^2}\right)^{\frac{1}{2}}}\right) \times 100 < \frac{\frac{1}{32}\left(\frac{Y}{R^2}\right)^4}{\left\{1 \pm \frac{1}{2}\left(\frac{Y}{R^2}\right)\right\}^2} \times 100 \end{aligned} \tag{33}$$

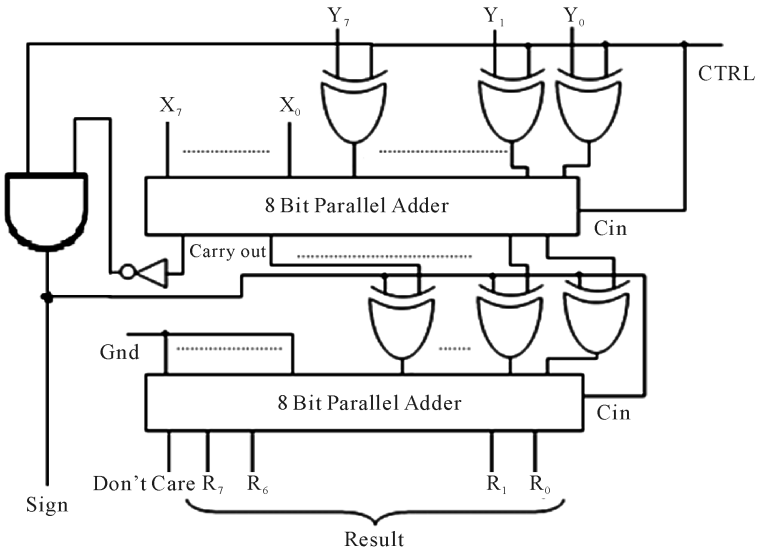

Figure 8. 8 bit binary adder-subtractor architecture.

If $x = \frac{Y}{R^2}$, then can easily be shown that $\frac{x^4}{32\left\{1-\frac{x}{2}\right\}^2} < \frac{1}{4\times 32} = \frac{1}{128} < \frac{1}{100}$.

So, $e_c(\%) \cong \left(\frac{R \times \frac{1}{32}\left(\frac{Y}{R^2}\right)^4 \left\{1 \pm \frac{1}{2}\left(\frac{Y}{R^2}\right)\right\}^{-1}}{R\left(1 \pm \frac{Y}{R^2}\right)^{\frac{1}{2}}} \right) \times 100 < \frac{x^4}{32\left\{1-\frac{x}{2}\right\}^2} \times 100 < \frac{1}{100} \times 100 = 1\%$. It is obvious that the error is minimum if and only if $X^{1/2} = R$ that is if $Y = 0$. Putting $Y = 0$ in Equation (33), the expression becomes $e_c(\%) = \left(\frac{0}{1}\right) \times 100 = 0\%$. **Table 1** exhibits the comparative study of the computational error in proposed algorithm with respect to "Bakshali" algorithm.

5. Result Analysis

The square root architecture has been implemented using VHDL and verified using Modelsim simulator which is shown in **Figure 9** and tested in Xilinx simulator with Xilinx Vertex-5 FPGA using XC5VLX30 device with a package FF676 at a speed grade (−3). **Table 2** shows the comparative study of the N-R method and the proposed method with respect to LUT count, delay and power consumption. The power has been measured using Xilinx Xpower Analyzer tool.

Table 1. Coefficients of the quotient expression.

Digit	Exact square root	Square root by Bakhshali method	Square root by modified Bakhshali (MB) method
194	13.928388277184119338467738928513	13.928427632973087518542063996601	13.928388278388278388278388278387
180	13.416407864998738178455042012388	13.416409521710381309235177430013	13.416413373860182370820668693008
175	13.228756555322952952508078768196	13.228756708407871198568872987468	13.228756708407871198568872987468
190	13.784048752090221767955912552934	13.784068995071780587100921362757	13.784048852701702442635085122128
174	13.190905958272919170936807732722	13.190906032742767436644987665394	13.190906032742767436644987665394
193	13.892443989449804508432547041029	13.892477688057798555036124096891	13.892443995593095850165258905612
183	13.527749258468682897425874817381	13.527753496503496503496503496502	13.527751601960045231813041839423
186	13.638181696985855892758597551 93	13.638190682556879739978331527621	13.638182498130142109199700822 73
188	13.711309200802088249871742898 17	13.711322990734755440637793578961	13.711309523809523809523809523804
192	13.856406460551018348219570732047	13.856435116130406989132750905599	13.856406480117820324005891016197

Table 2. Comparison of LUT count, delay and power of two methods.

Square root technique	4 bit			8 bit			16 bit		
	LUT (no. of slices)	Delay (ns)	Power (mw)	LUT (no. of slices)	Delay (ns)	Power (mw)	LUT (no. of slices)	Delay (ns)	Power (mw)
N-R method	17	3.56	2.78	36	10.03	4.63	252	33.13	5.32
Proposed method	16	3.47	2.56	34	7.47	3.15	103	27.43	3.96

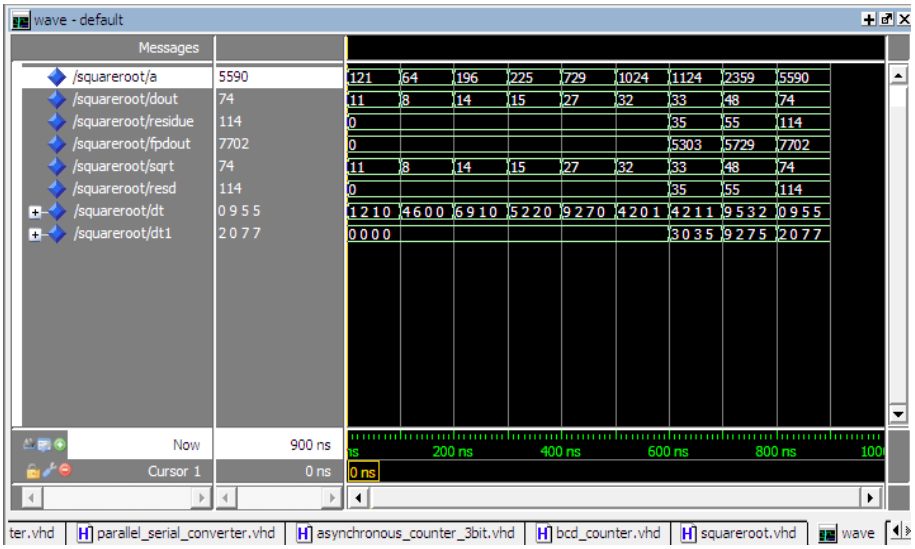

Figure 9. Test bench window of VHDL implementation of square root architecture.

6. Conclusion

From the result analysis, it is obvious that the architecture provides comparatively more accuracy than the well-known N-R approximation. Moreover, the proposed architecture produces less amount of propagation delay than N-R method and the power consumption is also very less as shown in **Table 2**. The area with respect to LUT count is also less than N-R method. It has also less circuit complexity than N-R technique which has been elucidated in the result analysis.

Acknowledgements

We would like to thank our dear colleagues for their precious support to execute the research in the department.

References

[1] Oberman, S.F. and Flynn, M.J. (1997) Design Issues in Division and Other Floating Point Operations. *IEEE Transactions on Computers*, **46**, 154-161.

[2] Pineiro, J.A. and Bruguera, J.D. (2002) High-Speed Double-Precision Computation of Reciprocal, Division, Square Root, and Inverse Square Root. *IEEE Transactions on Computers*, **51**, 1377-1388.

[3] Kwon, T.J. and Draper, J. (2008) Floating-Point Division and Square Root Implementation Using a Taylor-Series Expansion Algorithm with Reduced Look-Up Tables. 51*st Midwest Symposium on Circuits and Systems*, Knoxville, 10-13 August 2008, 954-995.

[4] Park, I. and Kim, T. (2009) Multiplier-Less and Table-Less Linear Approximation for Square and Square-Root. *IEEE International Conference on Computer Design*, Lake Tahoe, 4-7 October 2009, 378-383.

[5] Ramamoorthy, C., Goodman, J. and Kim, K. (1972) Some Properties of Iterative Square-Rooting Methods, Using High-speed Multiplication. *IEEE Transaction on Computers*, **C-21**, 837-847.

[6] Thakkar, A.J. and Ejnioui, A. (2006) Design and Implementation of Double Precision Floating Point Division and Square Root on FPGAs. *IEEE Aerospace Conference*, Big Sky.

[7] Ye, M., Liu, T., Ye, Y., Xu, G. and Xu, T. (2010) FPGA Implementation of CORDIC-Based Square Root Operation for Parameter Extraction of Digital Pre-Distortion for Power Amplifiers. 6*th International Conference on Wireless Communications Networking and Mobile Computing*, Chengdu, 23-25 September 2010, 1-4.

[8] Ercegovac, M.D. and McIlhenny, R. (2009) Design and FPGA Implementation of Radix-10 Algorithm for Square Root with Limited Precision Primitives. *Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, 1-4 November 2009, 935-939.

[9] Wang, X., Zhang, Y., Ye, Q. and Yang, S. (2009) A New Algorithm for Designing Square Root Calculators Based on FPGA with Pipeline Technology. 9*th International Conference on Hybrid Intelligent Systems*, **1**, 99-102.

[10] Maharaja, B.K.T. (1994) Vedic Mathematics. Motilal Banarasidass Publisher, Delhi.

[11] Saha, P., Banerjee, A., Bhattacharyya, P. and Dandapat, A. (2011) High Speed ASIC Implementation of Complex Multiplier using VEDIC Mathematics. *Proceeding of the* 2011 *IEEE Students*' *Technology Symposium*, Kharagpur, 14-16 January 2011, 237-341.