FPGA Implementation of Low Power High Speed Square Root Circuits

K N Vijeyakumar[1], Dr V Sumathy[2], P Vasakipriya[3], A Dinesh Babu[4]

[1,3]Department of Electronics and Communication Engineering, Anna University of Technology, Coimbatore.
[2]Department of Electronics and Communication Engineering, Government College of Technology, Coimbatore.
[4]Department of Electronics and Communication Engineering, SRM University, Vadapalani Campus, Chennai.

vijey.tn@gmail.com, sumi2001_gct@yahoo.co.in

***Abstract*— we present an efficient methodology for square root calculation using non-restoring SQUARE ROOT algorithm. The main principle of the proposed method is similar with conventional non-restoring algorithm, except that it eliminates the need for previous quotient in its final iteration. To reduce gate count we have proposed two area efficient subtract units and used in our proposed design. As an enhancement to our work we evaluated the design by eliminating the need for calculation of partial remainder in the final iteration. The proposed design and its enhanced version are designed using VHDL and simulated using Synopsys design compiler. Experimental results demonstrates better performance of our proposed architecture compared with the prior art in terms of area, power and delay.**

***Index Terms*—FPGA, non-restoring, Gate Level, square root**

I. Introduction

Much of the algorithms in scientific calculation, graphics, multimedia applications resembling digital signal processing (DSP) and image processing algorithms requires square root calculation[1]. The calculation of square root requires a lot of computational effort and is time expensive. To reduce the computational complexity a lot of square root algorithms and techniques have been developed and implemented such as Rough estimation, Babylonian method, exponential identity, Taylor-series expansion algorithm, Newton-Raphson method, Sweeney Robertson Tocher redundant method (SRT redundant method), SRT non redundant method and sequential algorithm (digit-by-digit method) [1][3],[4],[5]. The execution of these algorithms through software requires large time. With the advances in chip design an approach for hardware implementation of the square root algorithm is carried out in[2][3].However it is found that Newton Raphson method is slow for input operands with large bit widths[5].

Yamin Li et al., in [7],[8] discussed about the FPGA implementation of non-restoring square root algorithm. The speed of the circuits are high compared to architecture in [9]-[11] due to elimination of intermediate complex conversions to estimate partial square roots in each iteration along with elimination of seed generators and multiplexers. However the use of CSA and add/sub units in the architecture increases area cost for large bit-widths of input operand.

An unfolded pipelined architecture for square root calculation is proposed by Llamocca- Obregon in [12] . The architecture gives output in each cycle though the initial latency is high. However the delay of last stage increases the overall delay. Tole Sutikno[1] designed a square root circuit using non-restoring square root algorithm by eliminating redundant add/subtract blocks as proposed in [14] by Samavi et al.,. In [13] FPGA implementation of square root algorithm using polynomial approximation is discussed. Though the operating frequency of the architecture is high compared to architectures in [1],[14] the area cost is high due to complex multipliers used. Xiumin et al., [6] proposed a pipelined architecture for square root calculation. The architecture has high speed compared architecture in [13] , however the logic element usage increases rapidly for high bit widths of input operand. In this brief we propose a new strategy to implement a Square root calculator in FPGA using basics of non restoring algorithm. Section II discusses the basics of square root calculation using digit-by-digit algorithm. In section III, the proposed methodology and its implementation are discussed. Section IV describes the enhancements done to the design proposed in section III. Section V reports the experimental results of the proposed design and its enhanced version. In addition the comparison of the proposed and enhanced design with its counterparts are discussed in this section. Section VI gives a brief conclusion of the work done.

II. BASICS OF SQUARE ROOT CIRCUIS

Out of the algorithms present to find square root of a circuit, "digit-by-digit" algorithmic method is fast and requires only basic operators: adders and shifters. Inherent to digit-by-digit algorithm is a search and test step: find a digit, e, when added to the right of a current solution r, such that (r+e). (r+e)<=x, where, x is the value for which a root is desired. Expanding: r.r+2.r.e+e.e<=x.

Digit-by-digit method of can be divided in two classes, i.e. restoring and non-restoring algorithms [14]. In restoring algorithm, the procedure is composed by taking the square root obtained so far, appending 01 to it and subtracting it, properly shifted, from the current remainder.

K N Vijeyakumar is with the Anna University of Technology, Coimbatore, Tamil Nadu 641 047, India, (e-mail:vijey.tn@gmail.com).
Dr V Sumathy, is with Government college of Technology, Coimbatore Tamil Nadu 641 047, India, (email: sumi_2001@gmail.com).
P Vasakipriya is with the Anna University of Technology, Coimbatore, Tamil Nadu 641 047, India, (e-mail:pvasakipriya@gmail.com).
A Dinesh Babu is with the SRM University, vadapalani Campus, Chennai, Tamil Nadu 600 026, India, (e-mail: dbanbumani@gmail.com).

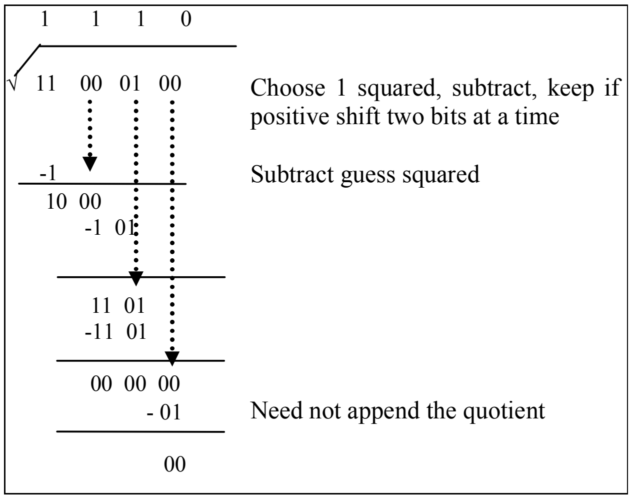

1 1 1 0
√ 11 00 01 00
Choose 1 squared, subtract, keep if positive shift two bits at a time
-1
Subtract guess squared
10 00
-1 01
11 01
-11 01
00 00 00
- 01
Need not append the quotient
00

Fig 1. Implementation of the algorithm for 8 bit number (Integer 196)

```
n - Number of bits in input operand

Step 1: Case(Bit_{n-1} & Bit_{n-2})
        If (Bit_{n-1} & Bit_{n-2} > 1);
            Quotient = 1;
        Else
            Quotient = 0;
        PR = Bit_{n-1} & Bit_{n-2} -1;
Step 2 : For(i = n-3 to 2)

        New PP = PR //  Bit_i Bit_{i-1}
        L = Quotient // 01
        PR =  New PP – L;

        If (PR > 0);
            Quotient = Append 1;
        Else
            Quotient =Append  0;
        Endif;
        i = i + 2;
Step 3 : New PP = Bit_1 Bit0;

        If ( 01 > New PP)
            New Quotient = Quotient//0;
        Else
            New Quotient = Quotient//1;
        Endif ;
```

Fig 2. Pseudo code of the proposed HS-Square root algorithm

The new root bit developed is truly 1, if the resulting remainder is positive, and 0 vice versa, which the remainder must be restored by adding the quantity just subtracted. The non-restoring algorithm does not restore the subtraction if the result was negative; instead, it appends 11 to the root developed so far and performs addition in the next step. If addition causes an overflow, then in the next iteration subtraction has to be performed [1].

III. PROPOSED HIGH SPEED NON -RESTORING (HS - NR) SQUARE ROOT ARCHITECTURE

We implemented the square root calculator with a little modification to the conventional square root algorithm. The basic step in our algorithm are based on digit by digit calculation and is similar to modified square root non-restoring algorithm proposed by Tole Sutikno in [1] except that the need for previous quotient has been eliminated in the final iteration. The non-restoring architecture [1] proposed by Tole Sutikno., is fast and easy to implement as it uses only subtraction operation and appends 01 for the calculation of square root of a number .In our architecture we use two new area efficient subtract units: S1 - performs subtraction using comparator and S2 - performs subtraction using multiplexer for both borrow and difference. The gate counts of proposed subtract units S1, S2 and CSM proposed in [1] is shown in table I. The implementation of the algorithm for an 8 bit example is shown in figure.1.

The pseudo code for the proposed HS square root algorithm is shown in figure.2 and the hardware implementation for an 8 bit example is shown in figure.3.

TABLE I
GATE COUNT OF PROPOSED SUBTRACT UNITS & CSM [1]

DESIGN	*S_1*	*S_2*	*CSM[1]*
AND	2	1	5
OR	2	1	2
XOR	4	4	-
2:1 Multiplexer	5	10	5
Inverter	1	-	7
Total Gate count	14	16	19

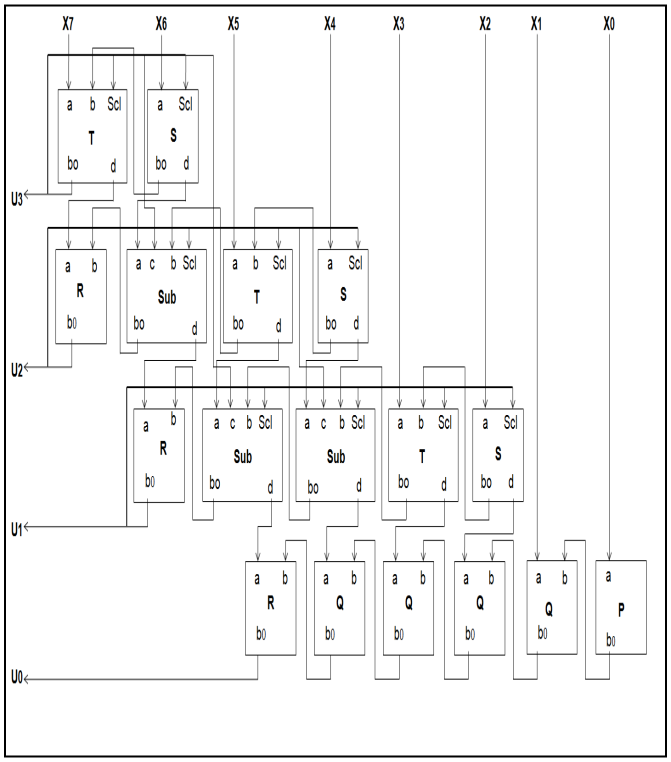


Fig 3. Architecture of unsigned 8-bit HS –NR square root circuit

The architecture shown in fig.3 uses proposed subtract units S_1 and S_2 shown in fig.4 and P,Q,R,S & T blocks . The input operand to the circuit is $X_7X_6X_5X_4X_3X_2X_1X_0$ and the root output is $R_3R_2R_1R_0$. P,Q,R,S & T blocks are the modified subtract units to which the inputs are set as follows: cbscl=100, cscl=00, cscl=00, cb=10 and c=0. It is seen that 8 bit architecture requires 4 iterations or stages to compute root. Thus n bit input operand requires n/2 stages to compute root. The subtract units S1 and S2 has 3 inputs: a, b and c and a select signal scl. The logic that defines the output of subtract units is given by

Subtract unit- S1

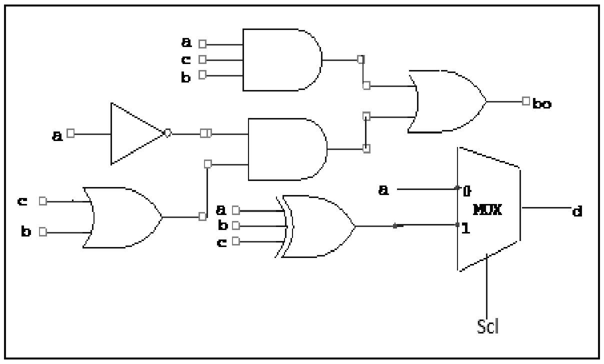
$$d = ascl' + (a\text{-}c\text{-}b)\ scl \quad (1)$$
$$bo = ((c+b)\ a') + abc \quad (2)$$

Subtract unit- S2

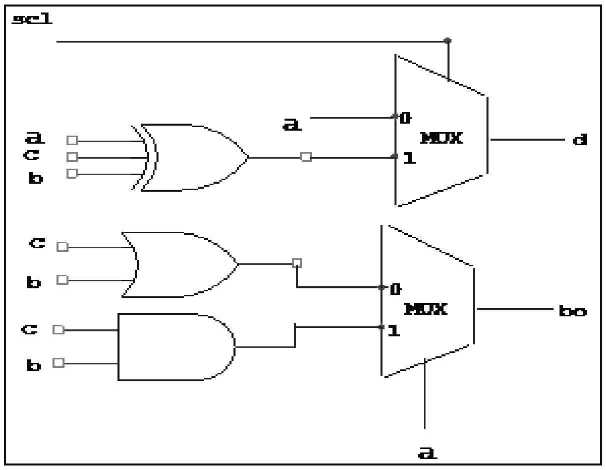
$$d = ascl' + (a\text{-}c\text{-}b)\ scl \quad (3)$$
$$bo = ((c+b)\ a') + abc \quad (4)$$

The equation that defines the output of the P, Q, R, S and T modules are given by

$$b0 = a \quad (5)$$
$$a'b = b0 \quad (6)$$
$$a'b = b0 \quad (7)$$
$$a.Sel' + b0.Sel = d \quad (8)$$
$$xu' + (x \text{ xor } b)u = d \quad (9)$$



(a)



(b)
Fig 4. Block diagram of subtract unit (a) S1 (b) S2

IV. ENHANCED DESIGN

As a further enhancement to our work we do not estimate partial remainder in the final iteration. The implementation of the enhanced algorithm for an 8 bit (integer-196) is shown in figure 5. The hardware implementation for an 8 bit example is shown in figure 6.

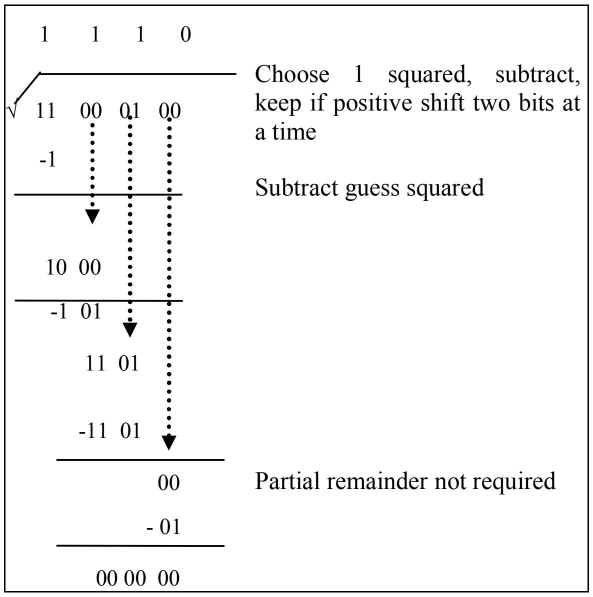

1 1 1 0
√ 11 00 01 00
-1
10 00
-1 01
11 01
-11 01
00
- 01
00 00 00
Choose 1 squared, subtract, keep if positive shift two bits at a time
Subtract guess squared
Partial remainder not required

Fig 5. Implementation of the enhanced algorithm for 8 bit number (Integer 196).

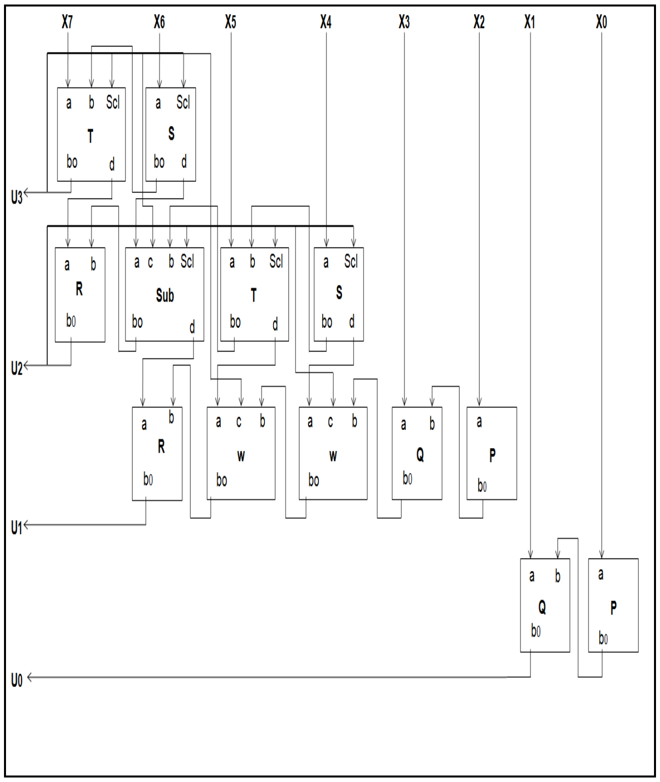


Fig 6. Architecture of Enhanced HS- NR square root calculator for n = 8.

TABLE II
COMPARISON OF POWER, DELAY AND PDP OF THE PROPOSED 8-BIT HS SQUARE ROOT TECHNIQUES WITH CONVENTIONAL METHODS

DESIGN	**DELAY (NS)**	**DYNAMIC POWER (µW)**	**CELL LEAKAGE POWER (µW)**	**TOTAL POWER (µW)**	**□ŕ**
Classical NR [14	14.848	230.700	1.107	231.806	5.8
Reduced NR [14]	13.125	181.69	0.94	182.634	4.05
XY module NR [14]	13.125	185.56	1.002	186.562	4.13
Simple X module NR [14]	12.679	161.183	0.519	161.702	3.46
Simple NR-D/D [1]	14.522	280.378	0.970	281.349	6.88
Optimized NR-D/D [1]	15.242	127.491	0.396	127.887	3.29
Proposed-HS NR (Using S_1)	11.079	105.762	0.375	106.137	1.98
Proposed-HS NR (Using S_2)	9.892	105.762	0.338	106.100	1.77
Enhanced HS NR(Using S_1)	9.892	59.586	0.218	59.804	1
Enhanced HS NR(Using S_2)	9.892	59.586	0.218	59.804	1

TABLE III
COMPARISON OF AREA OF THE PROPOSED HS-NR SQUARE ROOT CALCULATOR WITH PREVIOUS DESIGNS FOR N= 8, 16.

DESIGN	**AREA(μm^2)**	
	***n* = 8**	***n* = 16**
Classical NR [14	198.933	750.112
Reduced NR [14]	169.423	528.756
XY module NR [14]	180.593	534.595
Simple X module NR [14]	96.166	358.019
Simple NR-D/D [1]	211.508	600.95
Optimized NR-D/D [1]	86.346	359.283
Proposed-HS NR (Using S_1)	77.784	336.767
Proposed-HS NR (Using S_2)	74.578	331.228
Enhanced HS NR(Using S_1)	47.225	275.228
Enhanced HS NR(Using S_2)	47.225	275.228

V. RESULTS AND ANALYSIS

The proposed HS NR 8 bit square root calculator and its enhanced version are designed using VHDL code and synthesized using SYNOPSIS Design Compiler. Tole Sutikno design [1] and Samavi et al., design [14] are used for comparison.

The delay and power dissipation extracted from the simulation analysis are shown in table 2. It can be seen that the proposed square root calculator and its enhanced version demonstrates better performance in terms of power dissipation and delay compared to designs in [1] and [14]. This can be realized in better power-delay product ratio(represented as □ŕ calculated based on enhanced version) of our proposed design compared to all other designs mentioned in literature. The leakage power of our proposed HS-NR design and its enhanced version are better when compared to Tole Sutikno [1] and Samavi et al.,[14] designs.

In addition we have calculated the area of our proposed and enhanced designs for n = 8, 16 and reported in table 3. It is seen that the proposed HS-NR design and its enhanced version demonstrates better area reduction compared to the designs in [1] and [14]. This is due to the reduced gate count of the proposed subtractor units S1 and S2 employed in the architecture.

VI. CONCLUSION

The hardware implementation of HS-8 bit square-root calculator based on modified non-restoring algorithm using area efficient subtractor units is done. The main feature of the proposed methodology is the elimination of appending quotient and calculation of partial remainder at the final iteration. This paves the way for extension of the proposed methodology to higher bits with ease. The performance evaluations based on experimental results have shown that the proposed design is efficient in terms of power dissipation and hardware resource utilization compared to the previous designs.

REFERENCES

[1] Tole Sutikno, “An Efficient Implementation of the Non Restoring Square Root Algorithm in Gate Level,” in International Journal of Computer Theory and Engineerin, Vol.3, N0.1, Febrauary 2011, pp.46-51.

[2] L. Yamin and C. Wanming, "Implementation of Single Precision Floating Point Square Root on FPGAs," in IEEE Symposium on FPGA for Custom Computing Machines, Napa, California, USA,1997, pp.226-232.

[3] K. Piromsopa, et al., "An FPGA Implementation of a fixed-point square root operation," presented at the Int. Symp. on Communications and Information Technology (ISCIT 2001), ChiangMai, Thailand, 2001.

[4] Stanislaw Majerski ‘Square –rooting algorithm for High-Speed Digital Circuits” IEEE Transactions on computers Vol.34, issue 8, August 1985, pp.724-733.

[5] C. Ramamoorthy, J. Goodman and K. Kim:”Some properties of iterative Square-rooting methods Using High-speed Multiplication,” IEEE Transaction on Computers, Vol. C-21, No.8, 1972.pp837-847.

[6] W. Xiumin, et al., "A New Algorithm for Designing Square Root Calculators Based on FPGA with Pipeline Technology," in Hybrid Intelligent Systems, 2009. HIS '09. Ninth International Conference on, 2009, pp. 99-102.

[7] Li Yamin, Chu Wanming, "Parallel-Array Implementations of a Non-restoring Square Root Algorithm," Computer Design: VLSI in Computers and Processors, 1997,pp:690-695.

[8] Y. Li and W. Chu, “A New Non-Restoring Square Root Algorithm and Its VLSI Implementations”, Proc. Of 1996 IEEE International Conference on Computer Designs:VLSI in Computers and Processors, Austin, Texas, USA, October 1996,pp538-544.

[9] M. Birman, A. Samuels, G. Chu, T. Chuk, L. Hu, McLeod, and J. Barnes, “Developing the WTL3 170/3,171 Sparc Floating-point Coprocessors”, IEEE MICRO February,1990.pp55-64.

[10] T. Lang and P. Montuschi, “High Radix Square Root with Prescaling”, IEEE Transaction on Computers, Vol. 41,NO.8,1992.pp996-1009.

[11] T. Lang and P. Montuschi, “Very-high Radix Combined Division and Square Root with Prescaling and Selection by Rounding”, Proc. 12th IEEE Symposium on Computer Arithmetic, IEEE Computer Society Press, 1995.pp124-131.

[12] D. R. Llamocca-Obregon, "A Core Design to Obtain Square Root Based on a Non-Restoring Algorithm," presented at the IBERCHIPS Workhsop, Salvador Bahia, Brazil, 2005.

[13] LinZhimou, Lu Guizhu,"A Square Root Algorithm Based on FPGA,". Journal of Xiamen University (natural science edition),.2006, Vo1.45 ,No. 2,pp199-20l.

[14] S. Samavi, et al., "Modular array structure for non-restoring square root circuit," Journal of System Architecture, vol. 54, pp. 957-966, 2000.