FIXED-POINT SQUARE ROOTS

Abhishek Seth[1], *Woon-Seng Gan*[2]

Digital Signal Processing Lab, School of Electrical and Electronic Engineering,
Nanyang Technological University, SINGAPORE
Email: [1]aseth@ntu.edu.sg, [2]ewsgan@ntu.edu.sg.

ABSTRACT

Square root (SQRT) is a common arithmetic operation used in many DSP algorithms. In this paper, we evaluate square rooting methods suitable for implementation on fixed-point (FxP) DSP processors with a fast multiplying unit. The finite wordlength effect on the square rooting methods is highlighted, and it is shown that the theoretically derived convergence rate for the Newton-Raphson (NR) based square rooting methods are not suitable for FxP processor. Also, the most efficient methods for 8-bit and 16-bit FxP processors are identified.

Index Terms—*Fixed-point, square root, bit precision, DSP.*

1. INTRODUCTION

Square root (SQRT) computation is used in applications, like spectrum analysis, audio signal processing, digital communication, 3D graphics, and many others. A number of methods to find the SQRT of a number are described in [1]. Based on their structures, some of them are more suitable for hardware implementation, while others are more suited for software implementation on digital signal processing (DSP) processors with a hardware multiplier. In this paper, we will investigate several square rooting methods suitable for fixed-point (FxP) processors with a fast multiplying unit.

2. SQUARE ROOTING METHODS

There are several square rooting methods described in [1]. They can be broadly classified as (a) direct methods, (b) normalization techniques, (c) approximation by real functions, and (d) algorithms based on Newton-Raphson (NR) formula. Direct methods and normalization techniques are more suitable for hardware implementation, whereas approximation by real functions [2] and NR based algorithms [3] are usually programmed on processors with a fast multiplying unit. As many FxP processors, including DSPs and FPGAs, support fast multiplying unit, we shall investigate performance of (c) and (d) on FxP processors in this paper.

2.1. APPROXIMATION BY REAL FUNCTIONS

The two most popularly used functions for SQRT approximations are Taylor's series approximation (TSA) and Chebyshev polynomial approximation (CPA). The infinite length Taylor's series expansion for SQRT is expressed as

$$\sqrt{1+t} = 1+\frac{1}{2}t-\frac{1}{8}t^2+\ldots+(-1)^{n-1}\frac{1\times 3\ldots(2n-3)}{2\times 4\ldots 2n}t^n+\ldots, \quad (1)$$

and it is valid for $|t|<1$. A general n^{th} order Chebyshev expansion [2] can be derived from (1) by substituting with Chebyshev polynomials.

2.2. APPROXIMATION BY NEWTON-RAPHSON FORMULA

NR formula is used to calculate the SQRT of a number in an iterative manner. The three variants of NR method used to calculate square root of a number are (a) direct Newton-Raphson variant 1 (DNR-1), (b) direct Newton-Raphson variant 2 (DNR-2), and (c) inverse Newton-Raphson (INR). These variants are well studied methods [3], [4] and are described in the following sections. For the present discussion of NR based methods, we will use the following notations: SQRT(x)=$\sqrt{x}$ and ISQRT(x)/2=$1/\left(2\sqrt{x}\right)$.

A. Direct Newton-Raphson variant 1

DNR-1 is an iterative method [3] to compute SQRT of a number and is given by the following equation

$$y_{k+1} = y_k + \left(x-y_k^2\right)/\left(2\sqrt{x}\right), \quad k=0,1..., \quad (2)$$

where y_{k+1} is the estimated value of SQRT(x) obtained after (k+1) iterations. Each iteration of DNR-1 requires 2 multiplications and 2 additions.

B. Direct Newton-Raphson variant 2

The DNR-2 [3] is the second variant and is given by

$$1/\sqrt{x}_{k+1} = 1/\sqrt{x}_k \times 2-\left[\sqrt{x}_k \times 1/\sqrt{x}_k\right], k=0,1..., \quad (3)$$

and

$$\sqrt{x}_{k+1} = \frac{1}{2}\times\left[\sqrt{x}_k + x\ 1/\sqrt{x}_{k+1}\right], k=0,1.... \quad (4)$$

Each iteration of DNR-2 requires 3 multiplications and 2 additions to calculate $\sqrt{x}_{k+1}$ from $\sqrt{x}_k$, and $1/\sqrt{x}_k$ using (3) and (4). Each iteration enhances the approximation of both SQRT(x) and ISQRT(x). The improved approximations of SQRT(x) and ISQRT(x) are used in the next iteration.

Table 1 Initializing strategies for DNR-1, DNR-2 and INR methods to calculate FxP square roots.

NR variant	Initialization strategy		Method name	Remarks & References
	SQRT(x)	ISQRT(x)		
DNR-1	PE	LUT	NLIIRF	[4]
	LUT	LUT	$DNR_T(n)$	[5]
	PE	PE	DNR-1(PE,PE)	The output BP of DNR-1(LUT,PE) is very low, and is not evaluated. DNR-1(PE,PE) is evaluated in this paper.
DNR-2	PE	PE	DNR-2(PE,PE)	The output BP of DNR-2(PE,PE) is very low as compared to DNR-1 methods. The other (remaining 3) initializing strategies for DNR-2 are, therefore, not evaluated.
INR	PE	PE	INR(PE)	The output BP of INR(PE) is more than the DNR-2(PE,PE) but less than the DNR-1 methods. Therefore, the initializing strategy using LUT is not evaluated.

C. Inverse Newton-Raphson

The INR [3] method is the third variant and is given by

$$1/\sqrt{x}_{k+1} = \frac{1}{2} \times 1/\sqrt{x}_k \times \left[3 - x\ 1/\sqrt{x}_k^2\right], \quad k = 0,1..., \quad (5)$$

Each iteration of INR requires 3 multiplications and 1 addition to calculate $1/\sqrt{x}_{k+1}$ from $1/\sqrt{x}_k$. After (k+1) iterations, SQRT(x) is obtained by an additional multiplication, $x\ 1/\sqrt{x}_k = \sqrt{x}_k$.

The initial approximation for SQRT(x) and ISQRT(x) can be performed using polynomial expansions (PEs) or look-up tables (LUTs). While using PEs, we restrict ourselves to the 3rd order polynomials for both SQRT(x) and ISQRT(x) initialization i.e., linear PE (LPE), quadratic PE (QPE) and cubic PE (CPE) are used for SQRT(x) and ISQRT(x) initialization. For example, DNR-1 using LPE for both SQRT(x) and ISQRT(x) initialization is abbreviated as DNR-1(L,L). Among several initialization strategies possible, few strategies are explored previously in [4] and [5] for FxP square roots. In this work, we explore all possible initialization strategies for (FxP) DNR-1, DNR-2 and INR (listed in Table 1) to select the best method and the initializing strategy. We start with a comparison of the theoretical convergence rates of DNR-1, DNR-2 and INR in the next section.

3. CONVERGENCE OF NEWTON-RAPHSON BASED METHODS

Past works [3], [4] have reported the convergence rates of DNR-1, DNR-2 and INR methods. DNR-1 has the property of linear convergence, while the DNR-2 method has the property of convergence rate that lie between linear and quadratic. In addition, the INR method has the property of quadratic convergence. We simulated DNR-1, DNR-2 and INR on MATLAB using double precision (DP) floating-point (FlP) arithmetic for an interval of $0.25 \le x < 1$. LPEs are used for the SQRT(x) and ISQRT(x) initialization. We compare the output accuracy of INR, DNR-1 and DNR-2 against the MATLAB's DP FlP SQRT algorithm, which serves as our golden reference. Denoting the DP FlP result of the MATLAB SQRT algorithm by $y_{DP\ FlP}$ and DP FlP results of NR based methods after k^{th} iteration by $y_{k,DP\ FlP}$, bit precision (BP) of output sample is calculated as

$$\text{Bit precision of } y_{k,DP\ FlP} = -\log_2 | y_{DP\ FlP} - y_{k,DP\ FlP} |. \quad (6)$$

Total number of multiplications (B_M) and additions (B_A) required to generate 100% of the output samples with BP>N ($5 < N < 51$) bits are plotted in Fig. 1. The computational plots in Fig. 1 are in close agreement with the theoretical convergence rates of the NR variant algorithms [3]. Among the three methods, the INR method has the fastest convergence and requires minimum number of additions and multiplications for large values of N (N>8). The DNR-2 method has convergence rate that lie in between DNR-1 and INR, and DNR-1 has the slowest convergence rate. Note that DNR-1 requires the least number of operations (multiplications and additions) for small values of N ($N \le 8$). Similar trends are observed when other combinations of PEs are used for SQRT(x) and ISQRT(x) initialization. For an N-bit FxP processor, it would be useful to find out what percentage of output samples has BP>N−1. As DNR-1 requires minimum operations to achieve BP≤8 for all the output samples, it should be the preferred method for implementation on an 8-bit FxP DSP processor. Similarly, INR should be the preferred method for implementation on 16-bit or higher bits FxP DSP processors. In the next section, we shall investigate the performance of FxP square rooting methods on 8-bit and 16-bit FxP processors.

4. FIXED-POINT SQUARE ROOTS

The results for FxP DNR-1 using LPE and LUT to initialize SQRT(x) and ISQRT(x)/2, respectively have been reported in [4]. The authors called this method NLIIRF method, and evaluated it on a 16-bit FxP DSP processor. In [5], DNR-1 uses LUTs for both SQRT(x) and ISQRT(x)/2 initialization, and is named $DNR_T(n)$ where 'n' is the number of multiplications used by the $DNR_T(n)$ method. A generalized block diagram of the $DNR_T(n)$ method is shown in Fig. 2. It represents $DNR_T(n)$ as a cascade of a non-repetitive and repetitive structures, where the non-repetitive structure (NRS) performs the first iteration and the repetitive structure

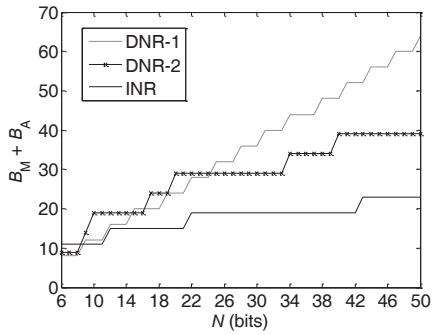

Figure 1 Total number of multiplications and additions required by DNR-1, DNR-2 and INR to generate 100% of the output samples with BP>N (5<N<51).

(RS) performs the subsequent iterations. The NRS can operate alone (i.e. no RS), or combines with one or more RS(s). The number of RSs used for DNR$_T(n)$ method are denoted by B_{RS}, and is shown in Table 2. Depending on the number of RSs used, 'n' takes up odd values (1,3,5 etc.). DNR$_T$(2) is a special case of DNR$_T(n)$, and is shown in Fig. 2. In our work, we use DNR$_T$(1) for an 8-bit FxP processor; and any one of the DNR$_T$(1), DNR$_T$(2) or DNR$_T$(3) for a 16-bit FxP processor. The S values used by DNR$_T$(1), DNR$_T$(2) or DNR$_T$(3) for S-bit truncation are listed in Table 2. To increase computational BP, enhanced precision (EP) LUTs are used [5]. The bit-widths of EP SQRT(x) and ISQRT(x)/2 LUT are denoted by BW_{SQRT} and $BW_{ISQRT/2}$, respectively, and shown in Table 2. Table 2 also shows the number of multiplications (B_M), additions (B_A), and memory bytes required by DNR$_T(n)$ for 8-bit and 16-bit FxP DSP processors. We use the specifications shown in Table 2 to generate the results for 8-bit and 16-bit FxP processors.

The square rooting methods described previously, namely TSA, CPA, NLIIRF, DNR$_T(n)$, DNR-1(PE,PE), DNR-2(PE,PE) and INR(PE), are simulated on 8-bit and 16-bit FxP processors using the MATLAB FxP toolbox [6]. We compare the accuracy of the FxP SQRT algorithm against our golden reference (MATLAB's DP FlP SQRT algorithm). Denoting the MATLAB's DP FlP result by $y_{DP\ FlP}$, and N-bit rounded result after k^{th} iteration (or k^{th} order) by $y_{k,N}$, BP of FxP output sample is calculated as

$$\text{Bit precision of } y_{k,N} = -\log_2|\, y_{DP\ FlP} - y_{k,N}\,|. \qquad (7)$$

Table 3 shows the percentage of output samples with BP>N −1 for N = 8 and 16. To compare the performance of various square rooting methods, number of iterations/order (B_I), additions, multiplications and total memory size required by each square rooting method are also calculated. The last column of Table 3 shows the (maximum) percentage of output samples with BP>N−1, and there is no further improvement in the percentage values even after increasing the number of iterations (or polynomial order). It is expected that the BP increases with increases in order (for real function based methods) or number of iterations (for NR based methods), but simulation results show that the output BP saturates and does not improve after certain order

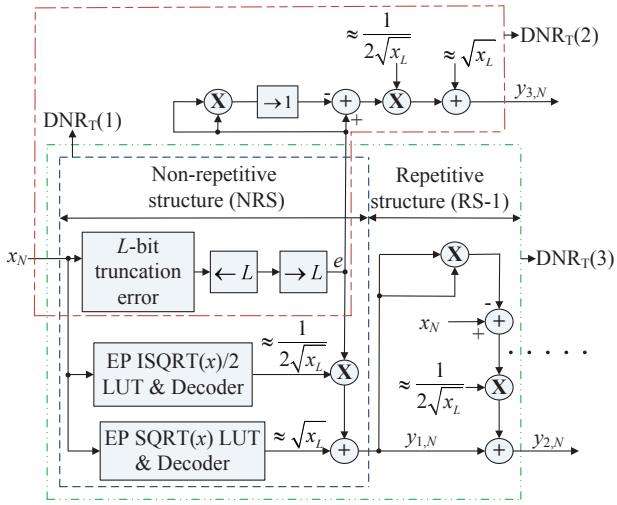

Figure 2 DNR$_T(n)$ as a cascade of repetitive and non-repetitive structures.

or iterations. This is due to the finite wordlength effect. Finite wordlength arithmetic also affects the convergence rates of NR based methods. According to theoretical convergence rates, DNR-1 method should produce the best results for 8-bit FxP processor followed by DNR-2 and INR methods. But due to finite wordlength effect, the output BP of DNR-2 and INR saturates after certain iterations. For 16-bit FxP processor, the INR method should produce the best result followed by DNR-2 and DNR-1 methods. But FxP simulation results show that DNR-1(PE,PE), DNR$_T(n)$ and NLIIRF methods perform better than INR and DNR-2 methods. Similar results are obtained for higher values of N (>16). This shows that DNR-1, DNR$_T(n)$ and NLIIRF methods are less sensitive to finite wordlength effect. In general, it can be said that for FxP processors, the DNR-1(PE,PE), DNR$_T(n)$ and NLIIRF methods generate the best results followed by INR(PE), DNR-2(PE,PE), TSA, and CPA.

The computational workload and BP results of Table 3 are useful in selecting appropriate square rooting method for FxP processor of the given wordlength. For an 8-bit FxP DSP processor, DNR$_T$(1) is the most suitable square rooting method as it provides maximum (100%) BP with minimum memory and computational workload. For a 16-bit FxP processor, DNR$_T$(1), NLIIRF, DNR-1(L,C) and DNR-1(Q,Q) provide the maximum (100%) output bit accuracy. In general, DNR-1(PE,PE), DNR$_T(n)$ and NLIIRF are highly precise square rooting methods. But they differ in the memory and computational workload requirements. Compared to other algorithms, DNR$_T(n)$ trades computational workload with memory without compromising its output BP. In contrast, DNR-1(PE,PE) and NLIIRF are computationally more expensive but require fewer memory. We, therefore, derive a set of highly precise FxP square rooting methods suitable for various implementation requirements. The next section examines the real-time implementation of the DNR$_T(n)$ methods.

Table 2 $DNR_T(n)$ with EP LUTs for 8-bit, and 16-bit FxP processors.

N	B_{RS}, BW_{SQRT}, $BW_{ISQRT/2}$	$DNR_T(n)$	S	B_M, B_A, Memory (Bytes)
8	0, 8, 8	$DNR_T(1)$	4	1, 3, 12
16	0, 16, 8	$DNR_T(1)$	8	1, 3, 288
	0, 16, 8	$DNR_T(2)$	7	2, 4, 144
	1, 16, 8	$DNR_T(3)$	6	3, 5, 72

5. REAL-TIME IMPLEMENTATION

The TMS320VC5505 [7], which is a 16-bit FxP processor from Texas Instruments (TI), is used for the real-time implementation of $DNR_T(1)$, $DNR_T(2)$ and $DNR_T(3)$. This processor provides DSP library to compute SQRT of a 16-bit FxP number [8], which is referred here as the DSPLIB SQRT. We evaluate the performance of the DSPLIB SQRT and compare it against the $DNR_T(n)$ based algorithms, in terms of BP and computational workload performance. The MIPS and memory results for $DNR_T(1)$, $DNR_T(2)$, $DNR_T(3)$ and DSPLIB SQRT are shown in Table 4. The MIPS results are obtained for the sampling frequency of 48 kHz. On-chip memory usage is the summation of the code size and data size for the respective $DNR_T(n)$ implementation. For DSPLIB SQRT, only 66.93% of output samples has BP>15, while this value is more than 99.66% for the $DNR_T(n)$ algorithms. In terms of MIPS requirement, $DNR_T(1)$ has the least computational load among the 4 algorithms, while the rest of the evaluated algorithms (Table 4) have comparable MIPS performance. However, memory requirement for $DNR_T(n)$ algorithms is significantly more than the DSPLIB SQRT algorithm. This is due to the fact that $DNR_T(n)$ algorithms are coded in C, without optimized for memory. However, the BP gain can significantly outweigh the increase in on-chip memory required.

Table 3 Performance comparison of square rooting methods on 8-bit and 16-bit FxP processors.

N	Method	B_I, B_M, B_A, B_M+B_A	Memory (Bytes)	BP (%)
8	$DNR_T(1)$	1,1,3,4	12	100
	NLIIRF	1,3,5.8	14	100
	DNR-1(C,L)	1,8,6,14	6	100
	TSA	6,9,5,14	6	91.75
	INR(C)	2,12,5,17	4	76.04
	CPA	6,9,5,14	6	65.98
	DNR-2(L,Q)	1,7,5,12	5	53.13
16	$DNR_T(1)$	1,1,3,4	288	100
	NLIIRF	3,7,9,16	28	100
	DNR-1(L,C)	2,10,8,18	12	100
	DNR-1(Q,Q)	2,10,8,18	12	100
	$DNR_T(3)$	2,3,5,8	72	99.98
	$DNR_T(2)$	1,2,4,6	144	99.67
	INR(Q)	4,16,6,22	6	85.07
	DNR-2(L,Q)	4,16,11,25	10	49.63
	TSA	6,9,5,14	12	40.48
	CPA	6,9,5,14	12	1.91

BP (%) – percentage of output samples with BP>N−1.

Table 4 Real-time implementation of $DNR_T(1)$, $DNR_T(2)$ and $DNR_T(3)$ on TMS320VC5505.

$DNR_T(n)$	BP (%)	MIPS	On-chip memory (Bytes)
DSPLIB SQRT	66.93	2.35	120
$DNR_T(1)$	100	1.54	1,140
$DNR_T(2)$	99.67	2.21	840
$DNR_T(3)$	99.98	2.88	530

6. CONCLUSION

In this paper, we presented a number of square rooting methods suitable for FxP platforms with a fast multiplying unit. The square rooting algorithms were simulated, and their BP and computational workload performances were compared for 8-bit and 16-bit FxP processors. The finite wordlength effect on each method was highlighted. The $DNR_T(n)$ algorithms outperformed all square rooting methods in terms of BP and computational workload performance for 8-bit and 16-bit FxP processors. In particularly, $DNR_T(1)$, $DNR_T(2)$ and $DNR_T(3)$ algorithms were implemented on the TI C55x DSP processor to validate their performances.

7. ACKNOWLEDGEMENT

This work is supported by the Singapore Ministry of Education Academic Research Fund Tier 2, under research grant MOE2010-T2-2-040.

8. REFERENCES

[1] P. Montuschi, and M. Mezzalama, "Survey of square rooting algorithms," *IEE Proceedings*, vol. 137, pp. 31-40, 1990.

[2] C. Clenshaw, "Polynomial approximations to elementary functions," *Mathematical Tables Aids Computation*, 1954.

[3] C. Ramamoorthy, J. Goodman, and K. Kim, "Some properties of iterative square rooting methods using high-speed multiplication," *IEEE Transactions on Computers*, vol. 21, pp. 837-847, 1972.

[4] N. Mikami, M. Kobayashi, and Y. Yokoyama, "A new DSP-oriented algorithm for calculation of square root using a nonlinear digital filter," *IEEE Transactions on Signal Processing*, vol. 40, pp. 1663-1669, 1992.

[5] A. Seth, and W. S. Gan, "Fixed-point square roots using *L*-bit truncation," to be published in *IEEE Signal Processing Magazine*, vol. 28, issue 6, 2011. [Online Available]: http://www3.ntu.edu.sg/home/aseth.

[6] MATLAB. *Fixed-Point Toolbox*. [Online]. Available: http://www.mathworks.com/help/toolbox/fixedpoint.

[7] TMS320VC5505 Fixed-Point Digital Signal Processor, SPRS503B, 2010.

[8] TMS320C55x DSP Library Programmer's Reference. SPRU422J, 2000.