International Scholarly Research Network
ISRN Signal Processing
Volume 2011, Article ID 615934, 8 pages
doi:10.5402/2011/615934

Research Article

Low-Complexity Inverse Square Root Approximation for Baseband Matrix Operations

Perttu Salmela,[1] Adrian Burian,[2] Tuomas Järvinen,[3] Aki Happonen,[4] and Jarmo Henrik Takala[1]

[1] *Department of Computer Systems, Tampere University of Technology, P.O. Box 553, 33101 Tampere, Finland*
[2] *Nokia Multimedia Imaging, Nokia Devices R&D, Nokia, Visiokatu 1, 33720 Tampere, Finland*
[3] *3GP/DSE, ST-Ericsson, Hermiankatu 1 B, 33720 Tampere, Finland*
[4] *Nokia Devices R&D, Nokia, Elektroniikkatie 3, 90570 Oulu, Finland*

Correspondence should be addressed to Perttu Salmela, perttu.salmela@gmail.com

Received 8 December 2010; Accepted 11 January 2011

Academic Editors: E. Salerno and E. D. Übeyli

Baseband functions like channel estimation and symbol detection of sophisticated telecommunications systems require matrix operations, which apply highly nonlinear operations like division or square root. In this paper, a scalable low-complexity approximation method of the inverse square root is developed and applied in Cholesky and QR decompositions. Computation is derived by exploiting the binary representation of the fixedpoint numbers and by substituting the highly nonlinear inverse square root operation with a more implementation appropriate function. Low complexity is obtained since the proposed method does not use large multipliers or look-up tables (LUT). Due to the scalability, the approximation accuracy can be adjusted according to the targeted application. The method is applied also as an accelerating unit of an application-specific instruction-set processor (ASIP) and as a software routine of a conventional DSP. As a result, the method can accelerate any fixed-point system where cost-efficiency and low power consumption are of high importance, and coarse approximation of inverse square root operation is required.

1. Introduction

Ever higher data rates require sophisticated transmission techniques but, on the other hand, the latest technologies allow use of advanced and more complex receiver algorithms. Such algorithms apply matrix operations which require highly nonlinear division by square root operation. For example, linear minimum mean square error (LMMSE) estimation has been proposed for the receivers of the current 3G, Universal Mobile Telecommunications System [1], and Cholesky decomposition can be used for the inevitable matrix inversion or for solving linear systems. In the upcoming 3G Long-Term Evolution (LTE) systems, multiple-input multiple-output (MIMO) receivers require demanding symbol detection methods like list sphere detector (LSD), which applies QR decomposition of a complex-valued channel matrix. When compared to matrix computation studies targeted for processing large matrices with highly parallel resources [2, 3], there are four notable differences in the baseband processing in the telecommunications field:

(i) the matrices are relatively small,

(ii) fixed-point number system is preferred,

(iii) there are real-time limits,

(iv) low complexity and low power consumption are of high importance.

In this paper, a low-complexity inverse square root approximation method is proposed for baseband matrix operations. The method relies on binary presentation of the fixed-point number system and it avoids large LUTs, large multipliers, and floating-point arithmetic units. The principal idea of the method is to substitute the highly nonlinear inverse square root function with a less nonlinear function with appropriate pre- and postprocessing. The

accuracy and complexity of the method can be adjusted with one design parameter. Thus, the method lends itself to lower-complexity applications where coarse approximations and fixed-point computations are preferred. In addition to comparison of hardware implementations of inverse square root methods, we show how the proposed method can be applied as a software routine or as an accelerating unit of an ASIP. The implementations are applied for Cholesky and QR decompositions required by 3G and 3G LTE receivers.

2. Previous Work

There are several methods to compute the inverse square root function. One of the basic approaches is to use lookup tables (LUT) for obtaining an initial value for iterations, which refine the value to higher accuracy [4, 5]. The main differences among these kinds of methods are in the size and content of LUT and the used iteration algorithm. In [5], a large multiplier was used since it was available in the targeted general purpose processor. In [6], savings were obtained by using a $m \times n$ multiplier, $m \leq n$, and utilizing the fact that less significant bits of intermediate result do not contribute to the accuracy of the final result. A software implementation using LUT initialization followed by iterations was presented in [7]. Another software approximation in [8] relied heavily on the binary representation of floating-point numbers.

LUTs using low-order polynomial approximation were applied in [9]. Polynomial approximation was also used in [10] where a second-degree minimax polynomial approximation was followed by modified Goldschmidt iteration. A comparison considering area costs was also given. Digit recurrence methods were proposed in [11, 12]. The main disadvantage of using digit recurrence when compared to iterative algorithms is their linear convergence. Approximation based on LUT followed by multiplication with operand modification was proposed in [13, 14] and used also in [15]. Argument reduction followed by series expansion was applied in [16]. Another approach is to work in logarithmic domain [17, 18] where the computation of the inverse square root is straightforward [19, 20].

For shorter word lengths (WLs) and for using fixed-point numbers, table addition methods have been proposed. These methods consist of parallel LUTs and multioperand additions. As a benefit, no multipliers are required. In [21], a symmetric table addition method (STAM) was developed as an extension to a simpler bipartite method. Selecting appropriate multipartite method, that is, design space exploration, was considered in [22]. The STAM enhanced with an error correction term and internal presentation in exponent and mantissa form was used in [23].

When compared to the previously mentioned methods, the proposed method in this paper is novel, that is, it is not a derivative of any of the existing methods. The area costs are kept at low as large LUTs and large multipliers are avoided. The proposed method lends itself also to software implementation. Furthermore, the proposed method can be adjusted to work only in subunitary range, which is sufficient for, for example, Cholesky decomposition, and the accuracy of the method can be adjusted along with the complexity up to a certain level while maintaining high area efficiency.

3. Targeted Matrix Decompositions in Baseband Processing

In this section, we describe where the targeted low-complexity inverse square root operations have been applied.

3.1. Baseband Processing with Fixed-Point Number System. As the baseband functions are applied in receivers of, for example, handheld telecommunications devices, low complexity is important for decreasing the area costs and power consumption. Therefore, fixed-point number system is preferred, that is, limited accuracy is applied. In this study, the targeted fractional word length (FWL) is 11 bits and integer word length (IWL) is 5 bits, that is, 16-bit words are assumed.

Targeted matrix operations are illustrative examples of baseband functions for two reasons. Firstly, the computations consist mainly of massive repetitions of a single operation, which is multiply and accumulate in this case. Secondly, an efficient mapping of computations to custom hardware or DSP is prevented by one less frequently used, but demanding, operation, which is inverse square root, $1/\sqrt{x}$, in this case. Thus, there is a realistic need for low-complexity, limited accuracy implementations of $1/\sqrt{x}$ function.

3.2. Cholesky Decomposition for LMMSE. The LMMSE estimation of transmitted data vector applies typically the Cholesky decomposition. Basically, the LMMSE estimate, $\hat{\mathbf{d}}$, can be expressed as a function of received data, $\mathbf{y}$, channel matrix, $\mathbf{M}$, and power of noise, σ^2:

$$\hat{\mathbf{d}} = \left(\mathbf{M}^H\mathbf{M} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{M}^H\mathbf{y}, \tag{1}$$

where it is assumed that autocorrelation $E(\mathbf{d}\mathbf{d}^H) = \mathbf{I}$. The estimation in (1) can be derived into a form, which can be presented as a linear system with positive definite real-valued matrix. With Cholesky decomposition $\mathbf{A} = \mathbf{L}^T\mathbf{L}$, such a linear system, $\mathbf{A}\mathbf{x} = \mathbf{b}$, can be solved with the aid of two triangular systems, that is, $\mathbf{L}^T\mathbf{z} = \mathbf{b}$ and $\mathbf{L}\mathbf{x} = \mathbf{z}$.

Diagonal elements, l_{ii}, of Cholesky factor $\mathbf{L}$ are defined as

$$l_{ij} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2} \tag{2}$$

and nondiagonal elements, l_{ij}, as

$$l_{ij} = \frac{1}{l_{ij}}\left(a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk}\right), \tag{3}$$

where a_{ij} denotes the elements of $\mathbf{A}$. Equations (2) and (3) show that nondiagonal elements require division by square root and diagonals square root operations. Thus, the division by square root can be replaced with multiplication with the

inverse square root, that is, two demanding operations are substituted with one demanding and one less demanding operation. The square root operation of (2) can also be computed with an additional multiplication, as $\sqrt{x} = x/\sqrt{x}$. An important property of Cholesky decomposition is the preservation of the subunitary of matrix elements, which limits the arguments of $1/\sqrt{x}$ operations efficiently.

3.3. QR Decomposition for LSD. The LSD is used in MIMO receivers to estimate the transmitted symbol, $\mathbf{s}$, by approximating maximum likelihood detection:

$$\mathbf{s}' = \arg\min_{\mathbf{s}} ||\mathbf{y} - \mathbf{H}\mathbf{s}||^2, \tag{4}$$

where $\mathbf{y}$ is the received symbol vector and $\mathbf{H}$ is complex-valued channel matrix whose dimensions are equal to the number of transmit and receive antennas of MIMO system. The approximation is based on substitution with QR decomposition $\mathbf{QR} = \mathbf{H}$, that is,

$$\mathbf{s}' = \arg\min_{\mathbf{s}} ||\mathbf{y}' - \mathbf{R}\mathbf{s}||^2, \quad \text{where } \mathbf{y}' = \mathbf{Q}^H\mathbf{y}. \tag{5}$$

The LSD approximates (5) by gradually increasing the dimensions of symbol vector and computing partial Euclidean distances. With this practice, the search space can be limited efficiently.

QR factorization with modified Gram-Schmidt algorithm [24] is presented in Algorithm 1.

It decomposes $\mathbf{H}_{n\times n}$ to the orthogonal $\mathbf{Q}_{n\times n}$ and upper triangular $\mathbf{R}_{n\times n}$. Conjugated transpose is denoted with $(\cdot)^H$. The lines 2 and 3 show that division by square root is required as the elements are divided by diagonals which are norms, $\|\cdot\|$. In a similar way as with Cholesky decomposition, the division can be substituted with multiplication by inverse square root.

```
(1) for k = 1 : n
(2)     R_{k,k} = ||H_{1:n,k}||
(3)     Q_{1:n,k} = H_{1:n,k}/R_{k,k}
(4)     for j = k + 1 : n
(5)         R_{k,j} = Q^H_{1:n,k} H_{1:n,j}
(6)         H_{1:n,j} = H_{1:n,j} − Q_{1:n,k} R_{k,j}
(7)     end
(8) end
```

Algorithm 1

4. Low-Complexity Approximation Method

The main principle of the proposed method is to avoid straightforward approximation of $1/\sqrt{x}$ function which is highly nonlinear in subunitary range $0 < x \leq 1$. Instead, the more softly nonlinear function $1/\sqrt{c+u}$ with $c \geq 1$ and $0 < u \leq 1$ is approximated. The usage of $1/\sqrt{c+u}$ is justified by the following fixed-point representations in two complement formats of x, c, and u. If the positive subunitary x has α leading zeros, c and u can be defined so that

$$x = \underbrace{0.00\cdots0}_{\alpha} c_{N-1}c_{N-2}\cdots c_0 u_{M-1}u_{M-2}\cdots u_0. \tag{6}$$

In other words, the bits of c and u do not overlap and the word lengths of c and u are denoted with N and M, respectively. Positive nonsubunitary range, $x > 1$, is presented similarly, except that the number of leading zeros, α, can have negative values. Since $c_{N-1} = 1$ for all valid values of x, the x can be presented with the aid of shifting by α, that is,

$$\begin{aligned} x \times 2^{\alpha} &= 1 \cdot c_{N-2}\cdots c_0 u_{M-1}u_{M-2}\cdots u_0, \\ x &= 2^{-\alpha} \times 1 \cdot c_{N-2}\cdots c_0 u_{M-1}u_{M-2}\cdots u_0. \end{aligned} \tag{7}$$

Thus, the desired form, $c + u$, can be obtained, and we note that u is a positive subunitary number. The targeted function can be written as

$$\frac{1}{\sqrt{x}} = \frac{1}{\sqrt{2^{-\alpha}(c+u)}} = 2^{\alpha/2}\frac{1}{\sqrt{c+u}}. \tag{8}$$

We can distinguish two cases depending on the value of α, which represents the number of leading zeros of fixed-point binary representation of x (6). This distinct behavior is obtained because the remainder of $\alpha/2$ in (8) can be either zero or one. For even values $\alpha = 2k$,

$$\frac{1}{\sqrt{x}} = 2^k\frac{1}{\sqrt{c+u}} \tag{9}$$

and, for odd values $\alpha = 2k + 1$,

$$\frac{1}{\sqrt{x}} = 2^k\sqrt{2}\frac{1}{\sqrt{c+u}}. \tag{10}$$

In order to approximate (9) or (10), the expression $1/\sqrt{c+u}$ must be considered. A tempting solution is to approximate $1/\sqrt{c+u}$ with binomial series. In principle, the $1/\sqrt{c+u}$ could be approached with arbitrarily high precision, as the binomial series converges. Multipliers are required if polynomial approximation [9, 10] or series expansion [16] is applied. Although the approximation with binomial series has a solid basis, it does not lend itself to low-complexity implementations due to the high-order terms.

4.1. Linear Approximation. We attempt to identify the characteristic of $1/\sqrt{c+u}$ and to determine a first-degree polynomial that will give the smallest approximation error for a low-complexity hardware implementation. So, we will approximate the expression $1/\sqrt{c+u}$ with straight lines, that is,

$$\frac{1}{\sqrt{c+u}} \simeq a_t - b_t(c+u), \tag{11}$$

where $a_t, b_t > 0$ and subscript t is the integer interpretation of the concatenation $c_{N-2}\cdots c_0\alpha_0$. The number of approximating lines, that is, the accuracy of the approximation, depends on the WL of c. Since the MSB of c has always constant value, $c_{N-1} = 1$, the number of approximating lines is 2^N.

The range of the targeted expression is $1 \leq 1/\sqrt{c+u} \leq 1/\sqrt{2}$, since $1 \leq c + u < 2$. The domain of c is defined by the

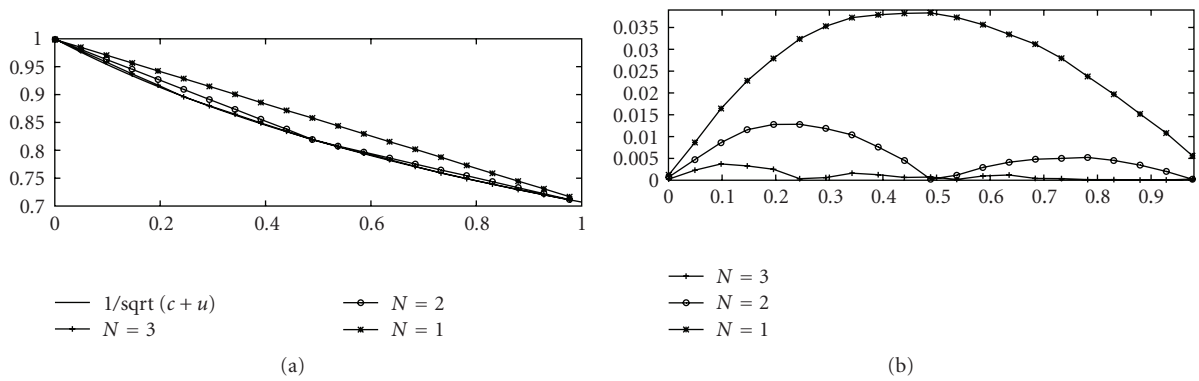

Figure 1: Linear approximation of $1/\sqrt{c+u}$: (a) approximating lines; (b) approximation error decreases as N is increased.

WL, N, that is, $1 \le c \le 2(1-2^{-N})$. Naturally, the domain of u depends on N and M, that is, $0 \le u \le 2^{-(N-1)} - 2^{-(M+N-1)}$. In practice, the approximating lines are formed by dividing the domain of $c + u$ into evenly spaced regions, which are determined by the N highest bits of c. The values in the start and end points are given by $1/\sqrt{c}$ and the value of the last end point is $1/\sqrt{2}$. The linear approximation is illustrated in Figure 1(a) where $1/\sqrt{c+u}$, with even α approximated with $N = 1, 2, 3$. The error of approximation is shown in Figure 1(b). The figures indicate that by increasing the word length N, the accuracy of the approximation can be adjusted conveniently.

For odd values $\alpha = 2k + 1$, the $\sqrt{2}/\sqrt{c+u}$ is approximated in a similar way. The lines used for even values, $\alpha = 2k$, cannot be used without multiplication with $\sqrt{2}$, and, therefore, different approximating lines are preferred. To obtain the final result, that is, the approximation of $1/\sqrt{x}$, the approximating straight lines in must be scaled with 2^k as shown in (9) and (10). The scaling can be carried out easily with shift operation, whose direction depends on the sign of α.

4.2. Coefficients for Hardware Implementation. The linear approximation has the form $a_t - b_t(c + u)$, which includes multiplication. However, for obtaining low complexity, the multiplications should be avoided. Braun multiplier adds shifted values of the multiplicand multiplied with one bit of the multiplier. The principle of adding shifted values can be used to approximate the product $b_t(c + u)$. Since $b_t \le 1/2$, the product can be presented as

$$b_t(c+u) = d_{1,t}\frac{c+u}{2^1} + d_{2,t}\frac{c+u}{2^2} + \cdots + d_{M+N-1,t}\frac{c+u}{2^{M+N-1}}, \tag{12}$$

where $d_{i,t} \in \{-1, 0, 1\}$. As division with powers of two can be implemented with hardwired shifting in hardware, an approximation of the previous form is suitable for low-complexity implementation. Naturally, the accuracy depends on the number of terms included in the sum. In the proposed method, at maximum three terms are included, that is, an approximation,

$$b_t(c+u) \simeq d_{1,t}\frac{c+u}{2^{e_{1,t}}} + d_{2,t}\frac{c+u}{2^{e_{2,t}}} + d_{3,t}\frac{c+u}{2^{e_{3,t}}}, \tag{13}$$

in which $d_{i,t} \in \{-1, 0, 1\}$ and $e_{i,t} \in \{1, \ldots, 8\}$, is used. The coefficients $d_{i,t}$ and $e_{i,t}$ are searched for each approximating line, that is, for each c and α_0, separately. Instead of three shifters with freely variable shift count, three multiplexers can be used to select appropriate terms.

5. Inverse Square Root Unit Implementations

The block diagrams of the hardware implementations of the inverse square root units are shown in Figure 2. Figure 2(a) shows only the linear approximation $a_t - b_t(c + u)$. The top three multiplexers correspond with term $b_t(c + u)$, and the fourth multiplexer outputs a_t. The selections of multiplexer are controlled by parity of α and bits of c excluding the c_{N-1} which has constant value.

In the next block diagram in Figure 2(b) the previous unit is instantiated in the inverse square root unit. The range of the unit in Figure 2(b) is positive subunitary, that is, $0 < x \le 1$, which is sufficient for the Cholesky decomposition. The structure is further extended in Figure 2(c) to allow free range, that is, $x > 0$. Basically, nonsubunitary range of x results also in negative values of α, and, therefore, both left shifting and right shifting are required as indicated in Figure 2(c). In practice, shifters consists of hardwired shift operations from which one is selected with multiplexer. Therefore, a combination of left and right shifters can be assumed to have the same complexity of unidirectional shifter with respectively wider range of shifted bits. As the input signal x has wider WL in Figure 2(c), the negative α is detected by comparing the number of leading zeros and IWL.

Only basic arithmetic and logic units are being used. The key components are priority encoder, adders, multiplexers, and shifters. Part of the functionality, for example, constant scaling, is implemented by hardwiring bits to the new positions. Due to the scaling, WLs of intermediate signals are relatively short. As the targeted accuracy depends on N, different implementations can be generated according to targeted application. Figure 2(a) shows a general case, that is, the number of inputs of multiplexers and N are free variables. In Figures 2(b) and 2(c) $N = 1$ and, therefore, multiplexers are controlled solely by α_0. If $N > 1$, the c is obtained from the output of the first shifter(s) and the control signal is

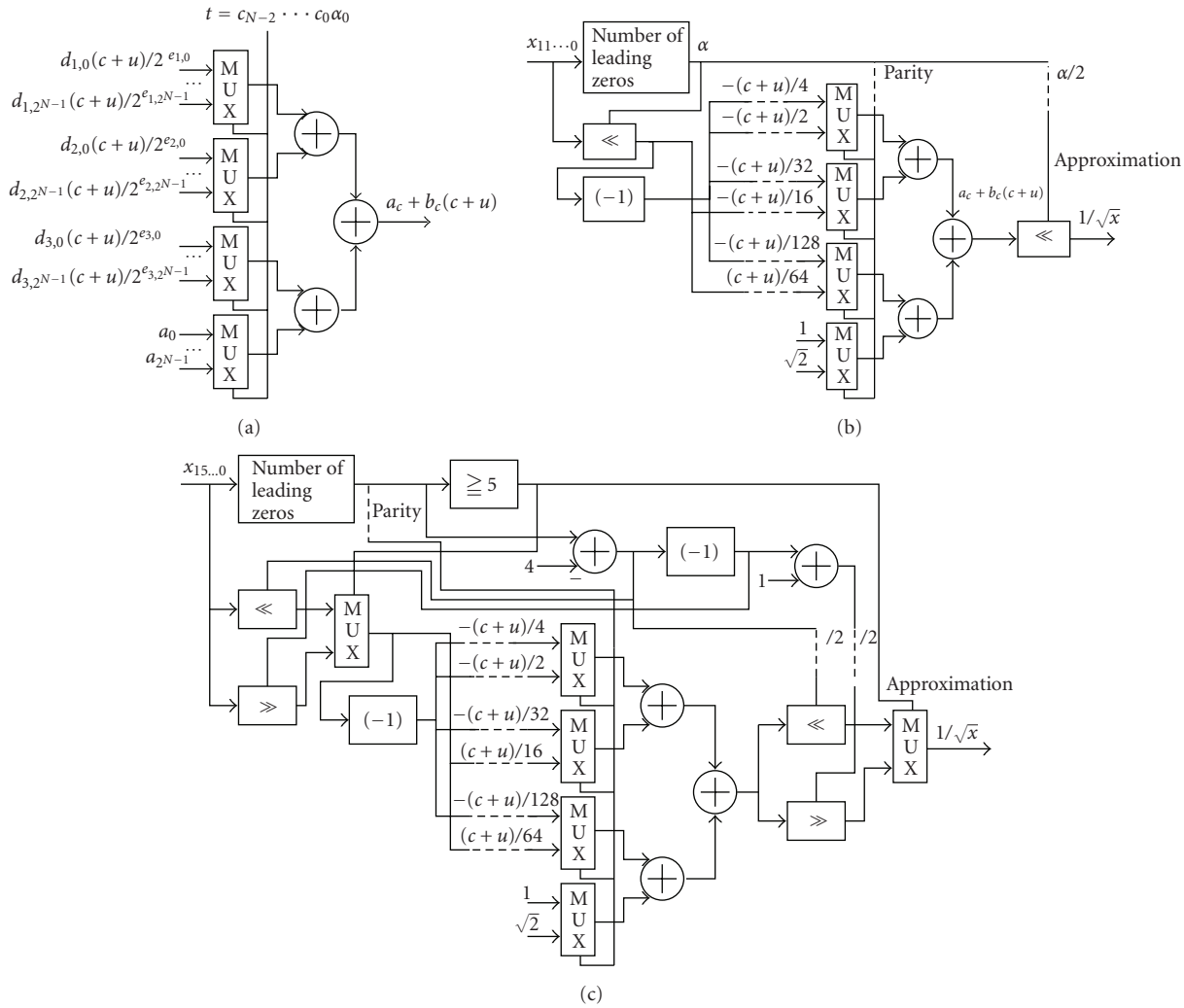

FIGURE 2: Units for (a) linear approximation with $a_t - b_t(c+u)$, (b) approximation of $1/\sqrt{x}$ in subunitary range, and (c) approximation in nonsubunitary range. Left shifting and right shifting are denoted with $\ll$ and $\gg$, respectively. Negation is marked with (-1).

generated by concatenation of $c_{N-2} \cdots c_0$ and α_0. Only the structure of linear approximation depends on N, and the other components in Figures 2(b) and 2(c) remain unaltered if N is increased.

6. Comparisons

Areas of the proposed method and competitive methods are estimated for a suggestive comparison. The proposed method is synthesized with 130 nm technology. The areas of other methods are estimated by considering their most expensive area components, such as multipliers and LUTs, unless more accurate details are clearly specified in the referred design. Only the mantissa of floating-point implementations is considered since its computation is similar in fixed-point number system.

6.1. Estimation of Area. Areas in terms of logic gate equivalents (GEs) of the synthesized arithmetic and logic operations with different WLs are given in Table 1. Since the basic unit of area is one NAND gate, fractions are possible. On the contrary to simple cost estimation of LUTs in [10], we have estimated the area of all LUTs individually. If structures of LUTs are not specified in detail, fair assumptions are made for the referred works. The synthesized LUTs are filled with random bits. The main reason for accurate modeling of LUT complexity is that the relative area of LUT depends both on the address line width and data WL. Estimated areas of all the LUTs are given in Table 1.

6.2. Compared Implementations. Since low area is emphasized in the targeted application domain of baseband processing, the methods are compared using the area efficiency as the ratio of accuracy versus area. The metric is defined as

$$\text{area efficiency} = \frac{\text{accuracy in bits}}{\text{area in GEs}}. \tag{14}$$

For single precision (SP) methods the accuracy is 23 bits and for dual precision (DP) methods 52 bits. The area efficiency

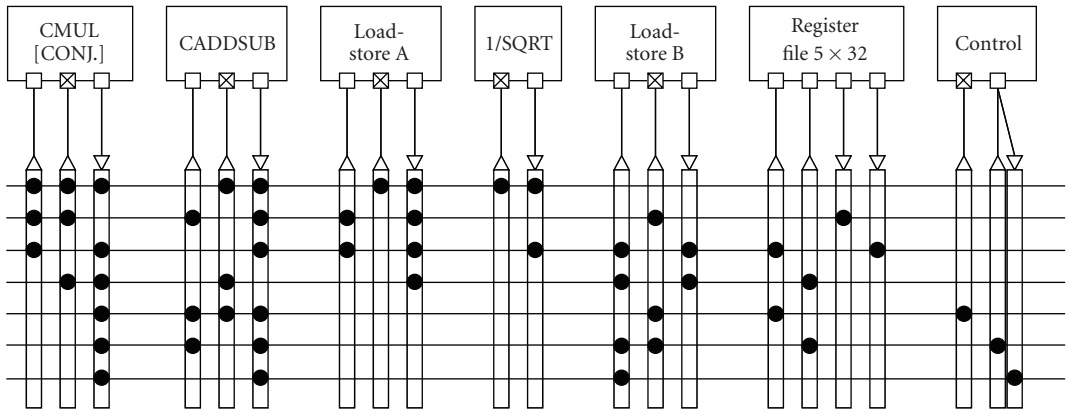

Figure 3: TTA processor for QR decomposition. Filled circles denote valid connections between resources and internal buses. The processor has special function units for complex-valued arithmetic and $1/\sqrt{x}$ approximation.

Table 1: Estimated area in terms of GEs of arithmetic and logic units and LUTs in compared implementations.

(a)

Unit	Operands	GE
Inverter	1	0.75
XOR	1×1	2.25
Full adder	$1 \times 1 \times 1$	6.50
Adder	8×6	34.75
Adder	19×10	76.50
Adder	20×9	78.75
Multiplier	7×7	247.00
Multiplier	20×20	2047.00
Multiplier	22×23	2542.00
Multiplier	16×56	4254.50
Multiplier	56×56	14647.25
Multiplier	76×76	26590.00

(b)

LUT size	GE
108×16	488.50
$2^9 \times 6$	505.50
216×8	593.75
$2^9 \times 8$	670.25
$2^9 \times 10$	872.00
$2^7 \times 36$	943.50
$2^9 \times 19$	2258.75
821×22	3720.50
$2^{10} \times 23$	4740.50
$2^{10} \times 25$	5065.50
$2^{11} \times 23$	9334.75

results for all the methods are shown in Table 2. The average accuracy of the proposed method in Table 2 is obtained in the subunitary range. Nonsubunitary range would increase the average accuracy even further. There are four versions of the proposed method with design parameter $N = 1, 2, 3, 4$. The results show that the proposed method has the lowest area, and even if the accuracy is adjusted with N, the area efficiency remains the highest except with $N = 1$. Naturally, the accuracy is relatively modest, as we have preferred the lowest area instead of high accuracy.

The first method in Table 2 was targeted to DP general purpose processor [5]. It required LUTs of sizes $2^{10} \times 23$ and $2^{11} \times 23$ and multiplier for 76×76 operands. Since the implementation was targeted to the general purpose processor, the hardware resources were not dedicated only to the inverse square root function. In [6] two 16×56 and one 56×56 multipliers were required. The total memory size was 72192 bits divided into four tables. For smaller gate count, we have assumed uniform division to four tables of 18048 bits with WL 22 bits, which is the widest word fetched from the tables. High speed was emphasized in [10]. Therefore, we compare with the method with single multiply and accumulate unit [10], which had better area efficiency. The authors also reported the complexity of 5030 full adders and, therefore, their value is used instead of our own estimates. In [13], SP floating-point numbers were targeted. A $2^{10} \times 25$ LUT was required and a 20×20 multiplier. In addition, a requirement of 15 inverters was reported. Symmetric table addition method (STAM) was used in [21]. The smallest total LUT size was obtained with four LUTs of sizes $2^9 \times 19$, $2^8 \times 10$, $2^8 \times 8$, and $2^8 \times 6$. In addition, a sum of all the data read from LUTs must be generated, which requires adders with operand sizes 19×10, 8×6, and 20×9. Also a requirement of 45 XOR gates was reported. Both SP and DP were targeted in [16] but the method for SP gave better area efficiency. The SP method required $2^7 \times 36$ LUT, four 4×4 multipliers, and one 22×23 multiplier. Fixed-point number systems were targeted in [23]. The method applied STAM enhanced with added correction value. Estimated complexity of 625 GE and LUT size of 3456 bits were given in [23]. Since the structures of LUTs were not reported, we have assumed that, due to the STAM, the memory is divided at least to two LUTs. We also assume 16-bit WL. Several smaller LUTs or shorter WL would degrade the area efficiency. The estimated complexity of LUTs is added to the reported gate count.

Table 2: Suggestive comparison of inverse square root methods. The proposed method has the highest area efficiency with $N = 2, 3, 4$.

method	Accuracy (FWL of result)	Area (GE)	Area efficiency (accuracy/area)
[5]	52	40665.25	0.0012787
[6]	52	38038.05	0.0013671
[10]	52	32695.50	0.0015905
[13]	23	7123.80	0.0032286
[21]	16	4597.80	0.0034800
[16]	23	4383.50	0.0052469
[23]	16	1602.00	0.0099875
Proposed $N = 1$	4	405.75	0.0098583
Proposed $N = 2$	6	433.75	0.0138329
Proposed $N = 3$	8	514.00	0.0155642
Proposed $N = 4$	10	622.00	0.0160772

6.3. Power Consumption. The power consumption of the largest proposed unit ($N = 4$) with 100 MHz is 0.339 mW, which includes the power required by input and output registers. Naturally, the static power consumption is proportional to the area, and, therefore, low complexity has been targeted. The dynamic power is proportional both to the area and average switching activity of the gates. Even if the average switching activity of competitive methods cannot be estimated sufficiently accurately, the differences in the area are significant. For example, [23] has the smallest area, 1602 GE, of the referred methods and the average switching activity of [23] should get as low as $622/1602 \times 100\% = 39\%$ of the average switching activity of the largest proposed unit (622 GE, $N = 4$) to achieve roughly the same dynamic power consumption.

7. Matrix Decomposition Implementation Case Studies

In this section, the proposed method for $1/\sqrt{x}$ approximation is applied in QR and Cholesky decomposition implementations. Many matrix decompositions are implemented with systolic arrays [25] applying inherent regularity of the algorithm, and the computations are alleviated with CORDIC [26] algorithm. However, such structures are not as flexible as programmable processors and such high parallelism can easily result in so fast processing that the array processor must idle most of the time if applied, for example, in QR decomposition of MIMO receiver.

7.1. QR Decomposition with ASIP. Complex-valued QR decomposition was implemented with transport triggered architecture (TTA) [27] processor in [28]. The TTA is an ASIP template where parallel computing resources can be tailored according to the application. The proposed $1/\sqrt{x}$ unit is instantiated in the processor as shown in Figure 3. In addition, there are units for complex addition and subtraction and complex multiplication with optional conjugation.

Typically, MIMO systems have only a couple of transmit and receive antennas, and, therefore, a 4×4 matrix decomposition is targeted. The modified Gram-Schmidt algorithm requires $2n^3$ operations for $n \times n$ matrix [24]. The processor implementation takes 139 clock cycles for 4×4 matrix. If 2048 subcarriers must be processed within a coherence time of the channel, 160 MHz clock frequency is adequate. The processor is synthesized with 130 nm technology and it takes 16.3 kGE with 160 MHz and 23.2 kGE with 269 MHz. The power consumption with 160 and 269 MHz clock frequencies is 6.91 mW and 16.79 mW, respectively.

7.2. Cholesky Decomposition with DSP. Cholesky decomposition was implemented as a software routine on TI's C55x DSP in [29]. Equations (2) and (3) show that the algorithm lends itself to the multiply and accumulate instruction, for which DSPs are typically optimized. Furthermore, an efficient hardware looping can be applied in the innermost loops as testing within the loop can be avoided. With the simple $1/\sqrt{x}$ approximation the developed program decomposes 64×64 matrix in 85070 clock cycles.

Maintenance of a continuous flow of computations is more important for an efficient software implementation than focusing on avoidance of multiplications. In other words, pipeline should be kept full by avoiding conditional branching when possible. For example, describing computation of α as defined in (6) in C language would result in a cumbersome loop testing bits of x. However, it can be avoided as the instruction set of the applied DSP has adequate assembly instruction for obtaining the number of leading zeros. Furthermore, short branches according to the parity of α can be avoided with guarded instructions, that is, the computations proceed uninterrupted by both branches but only the other branch affects the state. Thus, the proposed method lends itself to an efficient software implementation on a DSP with adequate instructions for obtaining the number of leading zeros and for guarded execution.

8. Conclusions

The inverse square root function is highly nonlinear function and, therefore, approximated usually with high-complexity implementation. The proposed approximation method targets moderate precision fixed-point numbers. The computation has been based on an appropriate substitute, which allowed approximation without large LUTs and large multipliers. The method has one design parameter which allows scaling of the accuracy and hardware complexity. The area efficiency of the proposed method has been given in terms of approximation accuracy per area. Comparisons with previously reported methods show that the proposed method achieves low complexity and high area efficiency. Finally, the method has been applied on the targeted baseband functions as a function unit of an ASIP and as a software routine on a DSP.

References

[1] P. Darwood, P. Alexander, and I. Oppermann, "LMMSE chip equalization for 3GPP WCDMA downlink receivers with channel coding," in *Proceedings of the IEEE International Conference on Communications (ICC '01)*, vol. 5, pp. 1421–1425, Helsinki, Finland, June 2001.

[2] G. Baker, J. Gunnels, G. Morrow, B. Riviere, and R. van de Geijn, "PLAPACK: high performance through high-level abstraction," in *Proceedings of the International Conference on Parallel Processing (ICPP '98)*, pp. 414–422, Minneapolis, Minn, USA, August 1998.

[3] J. Demmel, "LAPACK: a portable linear algebra library for supercomputers," in *Proceedings of the IEEE Control Systems Society Workshop on Computer-Aided Control System Design (CACSD '89)*, pp. 1–7, Tampa, Fla, USA, December 1989.

[4] H. Kwan, R. L. Nelson, and E. E. Swartzlander, "Cascade implementation of an iterative inverse-square-root algorithm, with overflow lookahead," in *Proceedings of the IEEE 12th Symposium on Computer Arithmetic*, pp. 115–122, Bath, UK, July 1995.

[5] S. F. Oberman, "Floating point division and square root algorithms and implementation in the AMD-K7 microsprocessor," in *Proceedings of the 14th IEEE Symposium on Computer Arithmetic*, pp. 106–115, Adelaide, Australia, April 1999.

[6] W. F. Wong and E. Goto, "Fast hardware-based algorithms for elementary function computations using rectangular multipliers," *IEEE Transactions on Computers*, vol. 43, no. 3, pp. 278–294, 1994.

[7] K. Turkowski, "Computing the inverse square root," Tech. Rep. 95, Media Technologies: Computer Graphics Advanced Technology Group Apple Computer, Inc., October 1994.

[8] C. Lomont, "Fast inverse square root," Tech. Rep., Department of Mathematics, Purdue University, West Lafayette, Ind, USA, February 2003.

[9] V. K. Jain, S. Shrivastava, A. D. Snider, D. Damerow, and D. Chester, "Hardware implementation of a nonlinear processor," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '99)*, vol. 6, pp. I-509–I-514, Orlando, Fla, USA, June 1999.

[10] J. A. Piñeiro and J. D. Bruguera, "High-speed double-precision computation of reciprocal, division, square root, and inverse square root," *IEEE Transactions on Computers*, vol. 51, no. 12, pp. 1377–1388, 2002.

[11] N. Takagi, "A hardware algorithm for computing reciprocal square root," in *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, pp. 94–100, Vail, Colo, USA, June 2001.

[12] T. Lang and E. Antelo, "Radix-4 reciprocal square-root and its combination with division and square root," *IEEE Transactions on Computers*, vol. 52, no. 9, pp. 1100–1114, 2003.

[13] N. Takagi, "Generating a power of an operand by a table look-up and a multiplication," in *Proceedings of the 13th IEEE Symposium on Computer Arithmetic*, pp. 126–131, Asilomar, Calif, USA, July 1997.

[14] N. Takagi, "Powering by a table look-up and a multiplication with operand modification," *IEEE Transactions on Computers*, vol. 47, no. 11, pp. 1216–1222, 1998.

[15] M. J. Schulte and K. E. Wires, "High-speed inverse square roots," in *Proceedings of the 14th IEEE Symposium on Computer Arithmetic*, pp. 124–131, Adelaide, Australia, April 1999.

[16] M. D. Ercegovac, T. Lang, J. M. Muller, and A. Tisserand, "Reciprocation, square root, inverse square root, and some elementary functions using small multipliers," *IEEE Transactions on Computers*, vol. 49, no. 7, pp. 628–637, 2000.

[17] J. N. Coleman and E. I. Chester, "A 32-bit logarithmic arithmetic unit and its performance compared to floating-point," in *Proceedings of the 14th IEEE Symposium on Computer Arithmetic*, pp. 142–151, Adelaide, Australia, April 1999.

[18] M. Haselman, M. Beauchamp, A. Wood, S. Hauck, K. Underwood, and K. S. Hemmert, "A comparison of floating point and logarithmic number systems for FPGAs," in *Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '05)*, pp. 181–190, Napa, Calif, USA, April 2005.

[19] C. H. Chen and C. Y. Lee, "Cost effective lighting processor for 3D graphics application," in *Proceedings of the International Conference on Image Processing (ICIP '99)*, vol. 2, pp. 792–796, Kobe, Japan, October 1999.

[20] A. Happonen, E. Hemming, and M. Juntti, "A novel coarse grain reconfigurable processing element architecture," in *Proceedings of the IEEE International Midwest Symposium on Circuits and Systems*, vol. 3, pp. 827–830, Cairo, Egypt, December 2003.

[21] J. E. Stine and M. J. Schulte, "Symmetric table addition method for accurate function approximation," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 21, no. 2, pp. 167–177, 1999.

[22] F. de Dinechin and A. Tisserand, "Some improvements on multipartite table methods," in *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, pp. 128–135, Vail, Colo, USA, June 2001.

[23] K. Rounioja and J. A. Parviainen, "Arithmetic processing unit for reciprocal operations," in *Proceedings of the International Symposium on System-on-Chip (SoC '03)*, pp. 109–112, Tampere, Finland, November 2003.

[24] G. H. Golub, *Matrix Computations*, John Hopkins University Press, Baltimore, Md, USA, 1989.

[25] S. Y. Kung, *VLSI Array Processors*, Prentice-Hall, Upper Saddle River, NJ, USA, 1987.

[26] R. Andraka, "Survey of CORDIC algorithms for FPGA based computers," in *Proceedings of the ACM/SIGDA 6th International Symposium on Field Programmable Gate Arrays (FPGA '98)*, pp. 191–200, Monterey, Calif, USA,, February 1998.

[27] H. Corporaal, *Microprocessor Architectures from VLIW to TTA*, John Wiley & Sons, New York, NY, USA, 1998.

[28] P. Salmela, A. Burian, H. Sorokin, and J. Takala, "Complex-valued QR decomposition implementation for MIMO receivers," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '08)*, pp. 1433–1436, Las Vegas, Nev, USA, April 2008.

[29] P. Salmela, A. Happonen, T. Järvinen, A. Burian, and J. Takala, "DSP implementation of Cholesky decomposition," in *Proceedings of the Joint 1st Workshop on Sensor Networks and Symposium on Trends in Communications*, pp. 6–9, Bratislava, Slovakia, June 2006.