INTEGRATION, the VLSI journal xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

On-chip implementation of a low-latency bit-accurate reciprocal square root unit

Cuauhtémoc R. Aguilera-Galicia[a,*], Omar Longoria-Gandara[a], Luis Pizano-Escalante[a], Javier Vázquez-Castillo[b], Manuel Salim-Maza[c]

[a] Department of Electronics, Systems and Informatics, ITESO-The Jesuit University of Guadalajara, Tlaquepaque, Mexico
[b] Department of Engineering, Universidad de Quintana Roo, Chetumal, Mexico
[c] NXP Semiconductors, Mexico

ARTICLE INFO

Keywords:
Digital CMOS integrated circuits
Reciprocal square root
Digital signal processing
Fixed-point arithmetic
Newton method
Piecewise-polynomial approximation

ABSTRACT

Many applications such as gaming, digital signal processing and communications systems, require computation of the reciprocal square root operation (RSR). Although several architectures have been reported for computing the RSR operation, these are mainly focused on accelerating high-precision floating-point units. In mobile-device implementations, fixed-point (FxP) units are preferred due to their low computational cost and power consumption. This article presents an on-chip implementation of a bit-accurate, FxP-RSR unit using a 130 nm CMOS process. The proposed architecture is based on a piecewise-polynomial approximation in a reduced range of the RSR function and the Newton-Raphson method. Experimental results show that the manufactured chip exhibits lower latency and less power consumption than existing standard-cell-based implementations. These characteristics make the proposed chip a useful silicon intellectual property suitable for embedded applications where low power, low latency, and low hardware cost is required.

1. Introduction

Fast computation of the square root and reciprocal square root (RSR) operations is required in diverse modern applications. For example, in multiple-input multiple-output (MIMO) wireless communication to perform tasks such as digital modulation [1], channel estimation [2,3], singular-value decomposition [4], and matrix inversion [5]. Likewise, in the area of digital signal processing these nonlinear operations are required for matrix decomposition, [6–9], and gaming for 3D image rendering [10,11].

Physical IPs can be utilized to improve the performance of electronics applications implemented in low-power embedded systems and mobile devices with limited computational resources, for example, the NXP microcontroller based on ARM Cortex-M4 [12]. Where the processing unit could present bottlenecks produced by complex operations such as the following elementary functions: exponential, logarithms, trig, hyperbolic trig, roots, RSR, among others. In these applications it is of paramount importance to reduce the microprocessor load, by implementing the complex operations in silicon IPs instead of executing them by software instructions. These customized blocks improve overall system performance in terms of speed and power consumption [8,9,10,11,13,4]. Due to its ever-expanding presence, having an off-the-shelf RSR intellectual property (IP) reduces time-to-market cycles and increases resource utilization.

The aforementioned implementations ([2,8,10,11]) have shown the benefits of using dedicated modules to compute the RSR. However, the performance of on-chip implementation has not been yet reported in the open literature, to the authors' best knowledge.

Several double-precision floating-point (FP) architectures for computing the RSR operation have been proposed. In Refs. [14] and [15] a modified digit-recurrence algorithm is used leading to high-latency (28 cycles). Initial works [16] used an architecture based on rectangular multipliers. Later [17], showed improved performance when using smaller multipliers and Taylor series evaluations. The proposal in Ref. [18] presents the best estimated cost-delay tradeoff among those mentioned here. It is based on look-up tables (LUTs), polynomial approximation and one Goldschmidt iteration. These architectures focus mainly on accelerating high-precision FP units. Hence, they are not suitable for mobile devices due to the hardware cost and power consumption. For instance [19], reports an double-precision FP unit that computes the $1/x$, $\sqrt{x}$, and $1/\sqrt{x}$ operations. It is synthesized using 180 nm CMOS standard-cells library, requires 0.524437 mm^2 of area

* Corresponding author.
E-mail addresses: cuauhtemoc@iteso.mx (C.R. Aguilera-Galicia), olongoria@iteso.mx (O. Longoria-Gandara), luispizano@iteso.mx (L. Pizano-Escalante), jvazquez@uqroo.edu.mx (J. Vázquez-Castillo), manuel.salim@nxp.com (M. Salim-Maza).

https://doi.org/10.1016/j.vlsi.2018.04.016
Received 18 December 2017; Received in revised form 16 March 2018; Accepted 29 April 2018

and has a power consumption of 40.8691 mW. These values are too high considering that [20] reports a single-precision FP unit for embedded applications, which computes the addition, subtraction, multiplication, and division operations and it consumes 71.2 mW with an area of 0.6 mm^2 using 180 nm CMOS process.

In addition, FP single-precision designs for computing the square-root operation have been reported. In Refs. [21] and [22], shared divider/square-root-circuit designs are reported, the integrated-circuit layouts are shown, the area and delay are specified, however measurements of the manufactured chips are not reported. Moreover, the technologies (1.2 µm) and design methodologies used in Refs. [21] and [22] are far from state-of-the-art. A standard-cell implementation of the RSR based on LUT and a modified NR iteration is presented in Ref. [23]. An improved version of [23] was later proposed in Ref. [24]. Alternatively [25], reports a standard-cell implementation of the square root based on LUTs and Taylor series. Synthesis results from a digit-recurrence square-root circuit for two standard-cell technologies (40 nm, and 60 nm) are presented in Ref. [26], which reports an estimated power consumption for each technology. A digit-recurrence implementation for computing the $1/x$, $\sqrt{x}$, and $1/\sqrt{x}$ operations is presented in Ref. [27]; it is based on radix-8 for determining the next digit and shows a latency of eight cycles.

In real mobile applications, the high-demand computing tasks are implemented in specialized fixed-point (FxP) units. This leverages lower hardware cost and reduces the power consumption of the FxP implementations [28,29,3]. Examples of this trend are the applications presented in Refs. [2,6,8,9], all of which use 16-bit FxP units to compute either the square-root or the RSR operation. Similarly [7], and [10] documented the use of 23-bit and 32-bit FxP units to perform the same operations, respectively.

Despite the advantages of the FxP arithmetic for real applications on mobile devices, few papers have reported an FxP implementation either of the square-root [29–33], or the RSR [34,3,35].

In this paper, we present the on-chip implementation of a low-latency, bit accurate, FxP RSR unit. The integrated circuit design is based on a piecewise-polynomial approximation and NR method [34]. The implementation includes the logical and physical synthesis using 130 nm CMOS (8RF-DM) ASIC technology. The physical chip design and its verification are performed and the post-silicon verification of the manufactured chip is reported. The proposed IP delivers 16-bit results in only two clock cycles. Hereafter the proposed unit is named 2C-RSR. The experimental results show that the power consumption of the proposed implementation is lower than previously reported designs [24,26]. The low latency of the 2C-RSR chip contributes to higher throughput for low clock frequencies, which is desired in low power embedded implementations [35]. Hence, the features of the reported IP are highly relevant for low-power mobile-device applications, such as [7–9,11].

2. 2C-RSR algorithm

The proposed RSR unit computes the operation

$$y = 1/\sqrt{x} \tag{1}$$

where $x, y \in \mathbb{R} \mid x, y = \sum_{i=-f}^{k-1} b_i 2^i$ with $b_i \in \{0,1\}$, and $k, f \in \mathbb{Z}$ are the number of bits for representing the integer and fractional parts respectively of x and y in FxP format.

In this work, an FxP format is represented by $Q(w, f, sign)$ notation, where $w = k + f$ is the word-length and $sign \in \{s, u\}$ indicates signed or unsigned format, respectively. Due to the finite size of w in real implementations, the result computed by (1) is an approximation of the exact value, i.e., $1/\sqrt{x}$ computed using infinite precision. Nevertheless, the proposed design is able to provide a result with a maximum error of $2^{-f}/2$ for the selected FxP format $Q(16,11,u)$, which makes the result bit-accurate with respect to the result computed by a double-precision FP unit (IEEE-754-2008 standard) [36] when this is represented in the $Q(16,11,u)$ FxP format. We selected this format because it allows to represent the magnitude of standard-Gaussian random variables, which is useful to study real-valued random variables whose distributions are unknown [37].

2.1. Bit-accurate property

Since bit-accurate is not a standardized concept, we define it below as used in this paper. Let (2) be the conversion operation of v, from decimal to binary FxP format

$$Q_w^f\{v\} = v^{w,f}. \tag{2}$$

In (2), v is the decimal representation of the result obtained from any arithmetic operation Θ performed by an FxP arithmetic unit. The expression $v^{w,f}$ stands for the binary representation of v in FxP format considering w and f parameters. Likewise, let (3) be the conversion operation of v_{FP}, from decimal to binary FxP format

$$Q_w^f\{v_{FP}\} = v_{FP}^{w,f}. \tag{3}$$

In (3), v_{FP} denotes the decimal representation of the result performed by a double-precision FP arithmetic unit, and $v_{FP}^{w,f}$ is the binary representation of v_{FP} in FxP format. Therefore, the bit-accurate property holds for v when (4) is met

$$v^{w,f} = v_{FP}^{w,f}. \tag{4}$$

To illustrate the bit-accurate property, Table 1 shows the comparison of two numerical results. The first row shows the result of the $1/\sqrt{9}$ operation performed by an FP arithmetic unit and its equivalent value when this is represented in FxP format (which can be obtained by using the *fi*(v,0,w,f) Matlab function). The second row shows the result of the same operation performed by a bit-accurate FxP arithmetic unit using $w = 16$ and $f = 11$. When this result is represented in $Q(16,11,u)$ format, all the bits are equal to the corresponding $v_{FP}^{16,11}$ value, and it can be said that the obtained result is bit-accurate. It must be noted that this does not necessarily holds for a different format, $Q(20,15,u)$ in this example. The advantage of a bit-accurate result computed by an FxP unit is that the result can be shared with a more-precise FP unit without introducing a conversion error.

2.2. 2C-RSR algorithm background

The 2C-RSR chip implements the algorithm reported in Ref. [34]. This is based on the Newton-Raphson (NR) method. The seed for the NR iteration is computed by a piecewise-polynomial approximation. Due to the nonlinearity of the RSR function, the polynomials are evaluated in a limited range of x, namely the working range (wr). This condition improves the polynomial fit and results in a better approximation. For computing the RSR of x when x is outside wr, a scaling and a de-scaling step are required. At the end, a rounding step is applied to obtain a bit-accurate result with a maximum error of ½ unit in the last place (ulp), with ulp = 2^{-f} for the $Q(w, f, sign)$ format. Each step of the algorithm is summarized below.

Table 1
Bit-accurate assertion.

$\Theta = 1/\sqrt{9}$		
Value v_x	$Q_{16}^{11}\{v_x\}$	$Q_{20}^{15}\{v_x\}$
v_{FP} = 0.333333333333333	0.01010101011	0.010101010101011
v = 0.33349609375	0.01010101011	0.010101010110000
Is v bit-accurate?	yes	no