A novel approximation scheme for floating-point square root and inverse square root for FPGAs

Pietro Pennestrì
Faculty of EEMCS
University of Twente
7522 NB Enschede, NL
Email: p.pennestri@student.utwente.nl

Yanqiu Huang
Faculty of EEMCS
University of Twente
7522 NB Enschede, NL
Email: yanqiu.huang@utwente.nl

Nikolaos Alachiotis
Faculty of EEMCS
University of Twente
7522 NB Enschede, NL
Email: n.alachiotis@utwente.nl

***Abstract*—Jointly computing the square root (SQRT) and the inverse square root (ISQRT) of floating-point numbers is common in many algorithms, e.g., in image or time series data processing when computing norms or vector normalization. Existing designs suffer from high latency and inefficient resource utilization due to the separate architectures that carry out these two operations. In this paper, we first propose a non-iterative approximation method for computing SQRT and ISQRT based on the Chebyshev min-max criterion to reduce the latency while meeting the accuracy requirements of various applications; thereafter a shared architecture of these two operations is designed and implemented in FPGA with less logic units. In contrast with other approximation solutions, our method does not need to perform any iterations and the accuracy can be mathematically estimated. A comparison with vendor-provided IP cores for FPGAs revealed that our proposed SQRT/ISQRT floating-point IP core utilizes less resources while reducing the clock-cycle latency by nearly four times.**

I. Introduction

The SQRT and ISQRT are two common operations in algorithms that require both the computation of vector norms and vector normalization. These tasks frequently occur in image processing, machine learning, and sensor fusion applications [1].

Studies have previously mapped these two operations onto reconfigurable hardware in FPGA [2]. Due to the resource-demanding nature of these operations on FPGAs, a common approach is to approximate these functions and then iteratively refine them, using, for instance, heuristically determined first-order approximation [1, 3]. This approach, however, relies on an initial guess, while the required number of iterations to reach the desired accuracy depends on the closeness of this initial guess to the actual function value, which cannot be quantified mathematically. Moreover, most existing FPGA implementations compute either the SQRT or the ISQRT but not both, thereby requiring to instantiate two floating-point cores, which results in inefficient resource usage for computing both operations. Hasnat et al. [4] presents an FPGA-based architecture that computes both the SQRT and the ISQRT, which, however, implements an iterative approximation method, thus its accuracy cannot be quantified mathematically.

To address the aforementioned limitations, we propose a novel non-iterative approximation method and describe a resource-efficient FPGA-based architecture that computes both the square root and the inverse square root. The main contributions of this work are:

- We propose a novel polynomial approximation scheme for $\frac{1}{\sqrt{x}}$ and $\sqrt{x}$ based on the Chebyshev criterion. It is non-iterative without an inital guess and the accuracy can be mathematically estimated.
- We present a shared architecture for the proposed approximated calculation of these two operations, while the computation of $1/\sqrt{x}$ and $\sqrt{x}$ is independent of each other.

The paper is organized as follows: Section II introduces the Chebyshev criterion and Sec. III presents the approximated calculation of (I)SQRT and the FPGA implementation. The evaluation results are summerized in Sec. IV followed by the conclusion in Sec. V.

II. Chebyshev approximation

Any continuous function can be approximated by using Chebyshev polynomials at the desired degree with the mathematically estimated approximation errors. This section introduces this approximation theory.

Let $f(x)$ be a given real-valued continuous function. It can be approximated as

$$F(P,x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_n x^n \tag{1}$$

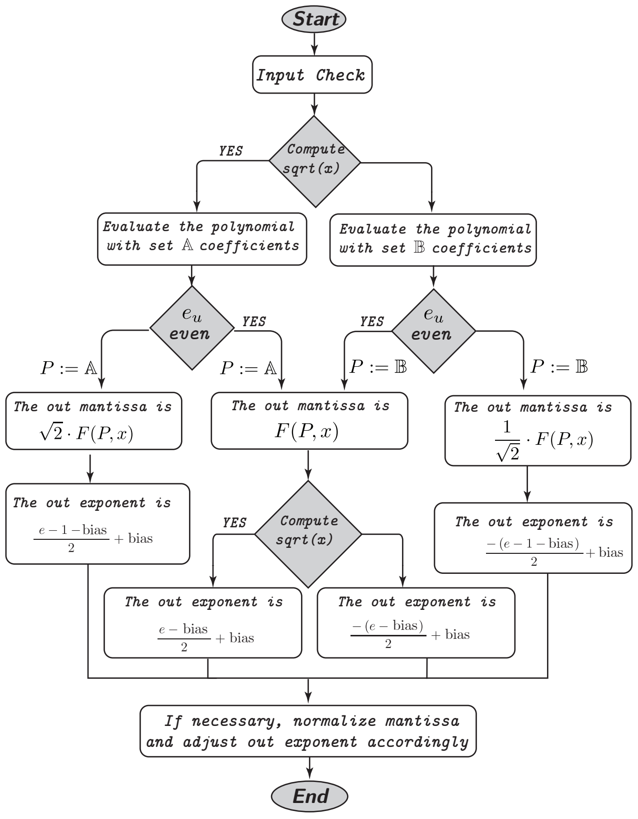
with n orders of polynomials and coefficients $p \in P$ over an interval of $[x_0, x_{n+1}]$ using the Chebyshev min-max polynomial approximation, such that the distance function $\varepsilon(x)$ is minimized, where

$$\varepsilon(x) = \max_{x_0 \le x \le x_{n+1}} \left| f(x) - F(P,x) \right| \tag{2}$$

It is proved that $F(P,x)$ is the Chebyshev approximation of a differentiable[1] function $f(x)$ on $[x_0, x_{n+1}]$ if and only if there exists a set of points

$$x_0 \le x_1 \le x_2 \le \ldots \le x_{n+1}$$

[1]Recent proofs (*e.g.* [5]) of the Chebyshev equioscillation theorem require continuity only. However, the original theorem statement from Chebyshev[6] includes the differentiability of $f(x)$


Start
Input Check
Compute sqrt(x)
YES
Evaluate the polynomial with set $\mathbb{A}$ coefficients
Evaluate the polynomial with set $\mathbb{B}$ coefficients
e_u even
YES
$P := \mathbb{A}$
$P := \mathbb{B}$
The out mantissa is $\sqrt{2}\cdot F(P,x)$
The out mantissa is $F(P,x)$
The out mantissa is $\frac{1}{\sqrt{2}}\cdot F(P,x)$
The out exponent is $\frac{e-1-\text{bias}}{2}+\text{bias}$
The out exponent is $\frac{-(e-1-\text{bias})}{2}+\text{bias}$
The out exponent is $\frac{e-\text{bias}}{2}+\text{bias}$
The out exponent is $\frac{-(e-\text{bias})}{2}+\text{bias}$
If necessary, normalize mantissa and adjust out exponent accordingly
End
Fig. 1: Overall computation scheme of the ISQRT & SQRT architecture

such that:

$$\varepsilon(x_i) = (-1)^i L \qquad (i = 0,1,\ldots,n+1) \tag{3a}$$

$$\left.\frac{\mathrm{d}\varepsilon(x)}{\mathrm{d}x}\right|_{x=x_j} = 0 \qquad (j = 1,2,\ldots,n) \tag{3b}$$

where L denotes the maximum value of ε in $[x_0, x_{n+1}]$.

These conditions ensure that the error function $\varepsilon(x)$, within the approximation interval, has $(n+2)$ maximas and minimas with alternating sign, and n stationary points, respectively. The previous $(2n+2)$ conditions, expressed by equations (3), are those required by the so-called Chebyshev's equioscillation theorem [6, 7, 5]. In particular, the resulting equations form the following nonlinear system

$$(-1)^{j+1} L + \sum_{i=0}^{n} p_i x_j^i - f(x) = 0 \quad (j = 0,1,2,\ldots,n+1) \tag{4a}$$

$$\sum_{i=1}^{n} i p_i x_k^{i-1} - \left.\frac{\mathrm{d}f(x)}{\mathrm{d}x}\right|_{x=x_k} = 0 \quad (k = 1,2,\ldots,n) \tag{4b}$$

After solving it using Chebyshev approximation, we can obtain not only the coefficients of the polynomials, p_0, p_1, ..., p_n, but also the maximum error L, as well as the location $x_1, x_2, \ldots, x_n$ where maxima and minima of $\varepsilon(x)$ occur. This makes our method different from others with a strong mathematical support of the approximation errors.

III. Proposed Computation Scheme of (I)SQRT and FPGA Implementation

This section introduces the proposed computation scheme of SQRT and ISQRT based on Chebyshev approximation, as well as the corresponding FPGA implementation.

A. Approximated SQRT/ISQRT based on Chebyshev theory

Let us denote with $x \in \mathbb{R}^+$ a number on which to apply the operations of $\sqrt{x}$ or $\frac{1}{\sqrt{x}}$. We assume that this number is expressed according to IEEE754 standard [8]:

$$x = (-1)^s 2^{e_u} \cdot m \tag{5}$$

where $m \in [1\,,2)$ is the mantissa and e_u is the *unbiased* exponent ($e = e_u + \text{bias}$). The following four cases can be distinguished:

- isqrt even exponent: $\frac{1}{\sqrt{x}} = 2^{\frac{-e_u}{2}} \cdot \frac{1}{\sqrt{m}}$;
- isqrt odd exponent: $\frac{1}{\sqrt{x}} = 2^{\frac{-e_u+1}{2}} \cdot \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{m}}$;
- sqrt even exponent: $\sqrt{x} = 2^{\frac{e_u}{2}} \cdot \sqrt{m}$;
- sqrt odd exponent: $\sqrt{x} = 2^{\frac{e_u-1}{2}} \cdot \sqrt{2} \cdot \sqrt{m}$.

On these premises, the task is limited to the computation of either the inverse square root or the square root of the mantissa. Hence, the proposed computation scheme is aimed toward a polynomial approximation of the two previous operations within the range $[1\,,2)$.

In order to meet different accuracy requirements, we have tried different polynomial orders, n, with different number of equally spaced sub-intervals, N. The maximum relative error is reported in Table I as n and N vary [2]. It is obvious that lower approximation error can be achieved by using higher polynomial degree or increasing the number of sub-intervals at the desired degree.

Based on Table I, users can select the proper n and N according to the accuracy requirement, and store the coefficients $\mathbb{A}$ for the SQRT and $\mathbb{B}$ for ISQRT. Depending on the operation SQRT/ISQRT and the even/odd exponent, the approximated mantissa can be obtained. Then sqrt(x) and/or isqrt(x) can be further calculated using Eqs.(6-9). The overall computation scheme is summarized in Fig. 1.

B. FPGA implementation

In this section, we present the FPGA implementation of the shared SQRT/ISQRT for the case $n = 2$ and $N = 8$. The proposed architecture is depicted in Fig. 4. The arithmetic core consists of two basic components, the polyroot, which computes the polynomial approximation of the output mantissa, and the exponent, which computes the output exponent as illustrated in Fig. 1. The exponent can be adapted to implement half-/single-/double-precisiong floating-point arithmetic.

Fig. 5 shows the block diagram of the polyroot block. It is based on Horner's polynomial evaluation scheme

[2]To replicate the proposed result see https://gitlab.com/mocast-isqrt-sqrt/matlab-code

TABLE I: Proposed method: Computation of maximum relative error (MRE) within the range [1 , 2), n: Polynomial degree , N: number of sub-intervals of $\log_2(N)$ amplitude

$1/\sqrt{x}$			$\sqrt{x}$		
n	N	MRE (this inv.)	n	N	MRE (this inv.)
2	1	3.8335e-03	2	1	7.6384e-04
2	2	7.1823e-04	2	2	1.4346e-04
2	4	1.1463e-04	2	4	2.2916e-05
2	8	1.6430e-05	2	8	3.2855e-06
2	16	2.2090e-06	2	16	4.4179e-07
2	32	2.8674e-07	2	32	5.7347e-08
2	64	3.6537e-08	2	64	7.3073e-09
3	1	5.7648e-04	3	1	8.2059e-05
3	2	6.3523e-05	3	2	9.0636e-06
3	4	5.5903e-06	3	4	7.9832e-07
3	8	4.2321e-07	3	8	6.0452e-08
3	16	2.9293e-08	3	16	4.1847e-09
3	32	1.9301e-09	3	32	2.7573e-10
4	1	8.9120e-05	4	1	9.8694e-06
4	2	5.7777e-06	4	2	6.4124e-07
4	4	2.8042e-07	4	4	3.1147e-08
4	8	1.1213e-08	4	8	1.2457e-09

and consists of an adder, a multiplier, and an on-chip ROM memory block that stores the polynomial coefficients. The arithmetic core was mapped to a `Cyclone V 5CEFA2F23I7` FPGA.

IV. Evaluation

This sections evaluates the accuracy and resource utilization of our approximation scheme by comparing it with related literature.

A. Accuracy comparison

As introduced in Sec. 1, there are different ways of approximating SQRT/ISQRT. Here we compare the accuracy of our method with other three methods [1, 3, 4] in terms of relative error. Fig. 2 depicts the comparison of our method vs. [1]. It can be observed that our method achieves two orders of magnitude lower error. The worst case relative error reported by [3] is 0.0333, which is one order of magnitude greater than ours. In order to compare with [4], we use the same intervals. The following linear approximations are derived based on Chebyshev theory:

$$P_1^{\mathrm{m2}}(x) = \begin{cases} 1.50971358702244 - 0.500000000000000 \cdot x \\ 1.34377614452159 - 0.358217995563320 \cdot x \\ 1.28092920145555 - 0.310455290827024 \cdot x \\ 1.22617183547953 - 0.272442417019179 \cdot x \\ 1.17789649241698 - 0.241599852241761 \cdot x \\ 1.13491593196515 - 0.216167221275863 \cdot x \\ 1.09632697261222 - 0.194903334931431 \cdot x \end{cases} \tag{6}$$

Fig. 3 shows the error comparison between the results reported in [4] and our method using Eq. (6).

B. Resource utilization and latency comparison

Table II shows resource utilization and a performance comparison with the vendor-provided IP cores `ALTFP_INV_SQRT` and `ALTFP_SQRT`. It can be observed that the second-order polynomial approximation

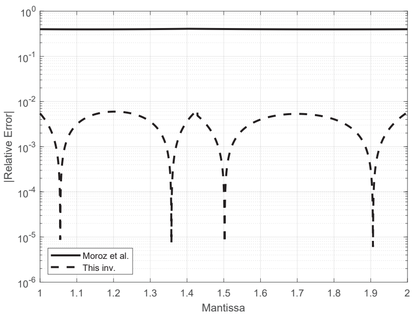


Fig. 2: comparison of relative error between moroz *et al.* [1] and our method

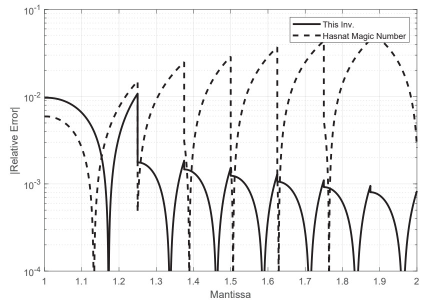


Fig. 3: $1/\sqrt{x}$ approximation: relative error comparison between the magic constants proposed by hasnat *et al.* [4] and our method using eq. (6)

only needs one DSP block and has a latency of 7 clock cycles. Furthermore, less hardware resources are used (shared between the SQRT and the ISQRT). More specifically, up to 3.7x less logic resources and up to 9x less registers are utilized at the cost of a 9x increase in block RAMs, which allows for higher flexibility in efficiently using the target hardware since ALMs and registers can, unlike block RAMs, be efficiently used to implement any functionality.

Table III provides the maximum operating clock frequency for `Slow 1100mV 100C`, `Slow 1100mV -40C`, `Fast 1100mV 100C`, `Fast 1100mV -40C` models.

Approximation accuracy was measured using `Modelsim` simulations. The maximum relative error both for $1/\sqrt{x}$ and $\sqrt{x}$, is lower than 1/2 ulp of the half precision format.

V. Conclusion

Novel polynomial approximations, based on the Chebyshev min-max criterion, for the considered operations,have been proposed. The accuracy achieved compares positively with that presented in recently published literature. An architecture capable of independently computing, in floating point notation, both $\sqrt{x}$ and $\frac{1}{\sqrt{x}}$ was designed and implemented into `FPGA`. The result was compared with existing vendor-provided IP cores, achieving 3 and 4 times less logic utilization for `ALTFP_INV_SQRT` and `ALTFP_SQRT`, respectively. The proposed architecture uses 6 times less DSP blocks than the `ALTFP_INV_SQRT`. The latency in clock

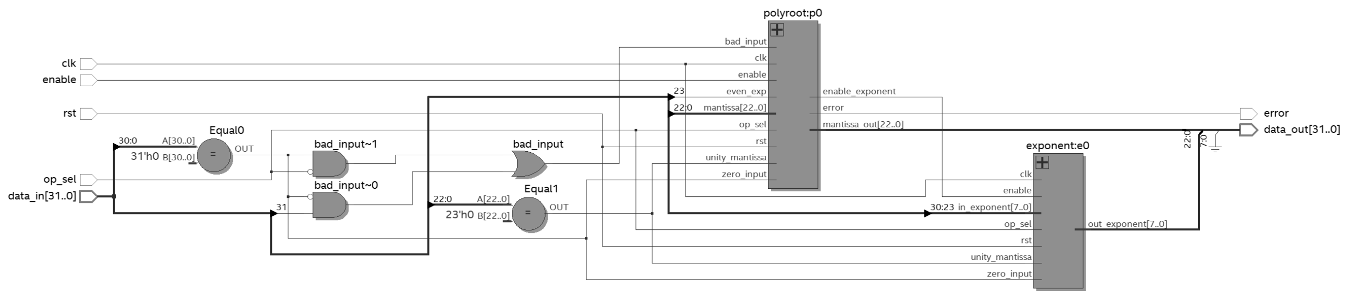
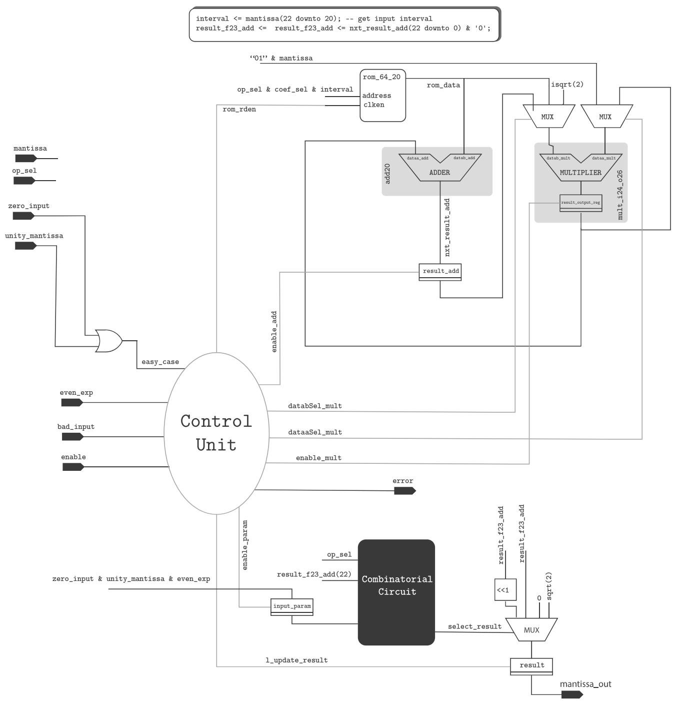


Fig. 4: Top View of the implemented architecture



```
interval <= mantissa(22 downto 20); -- get input interval
result_f23_add <=  result_f23_add <= nxt_result_add(22 downto 0) & '0';
```

Fig. 5: Proposed simplified schematic of the mantissa computation block (`polyroot` architecture)

TABLE II: Resource utilization and performance on a Cyclone V FPGA, and comparison with vendor IP cores.

	This work	ALTFP_INV_SQRT	ALTFP_SQRT
Logic Utilization (ALMs)	66/9430 (< 1%)	278/9430 (3%)	245/9430 (3%)
Registers	60	925	542
Block Memories	1280/1802240 (<1%)	565/1802240 (<1%)	143/1802240
DSP Blocks	1/25 (4%)	6/25 (24%)	0/25 (0%)
Latency (clock cycles)	7	26	16

cycles is 3 and 2 times less than the `ALTFP_INV_SQRT` and the `ALTFP_SQRT`, respectively.

TABLE III: Timing Analysis: Maximum Frequency

Model	Max. Frequency [MHz]
Slow 1100mV 100C	127.28
Slow 1100mV -40C	123.87
Fast 1100mV 100C	230.47
Fast 1100mV -40C	260.28

References

[1] L. V. Moroz, V. V. Samotyy, and O. Y. Horyachyy, “Modified fast inverse square root and square root approximation algorithms: the method of switching magic constants,” p. 21.

[2] Y. Li and W. Chu, “Implementation of single precision floating point square root on FPGAs,” in *Proceedings. The 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines Cat. No.97TB100186)*. IEEE Comput. Soc, 1997, pp. 226–232.

[3] L. Parrilla, A. Lloris, E. Castillo, and A. Garcia, “Table-free seed generation for hardware newton–raphson square root and inverse square root implementations in IoT devices,” pp. 1–1.

[4] A. Hasnat, T. Bhattacharyya, A. Dey, S. Halder, and D. Bhattacharjee, “A fast FPGA based architecture for computation of square root and inverse square root.” IEEE, pp. 383–387.

[5] R. Mayans, “The Chebyshev equioscillation theorem,” *Convergence*, 2007.

[6] P. Chebyshev, “Sur les parallélogrammes les plus simples symétriques autour d’un axe,” in *Oeuvres de P.L. Tchebichev*. Académie Impériale del Sciences, 1907, vol. II, pp. 709–715, published in 1878.

[7] E. Remez, *General computational methods of Chebyshev approximation: The problems with linear real parameters* , ser. Translation series. U.S. Atomic Energy Commission, Division of Technical Information, 1962.

[8] “IEEE standard for binary floating-point arithmetic,” *ANSI/IEEE Std 754-1985*, pp. 1–20, 1985.