

Article

A Low-Cost High Radix Floating-Point Square-Root Circuit

Yuheng Yang ¹, Qing Yuan ^{1,2} and Jian Liu ^{1,*}

¹ Institute of Microelectronics of the Chinese Academy of Sciences, Beijing 100029, China; yangyuheng@ime.ac.cn (Y.Y.); yuanqin@ime.ac.cn (Q.Y.)

² University of Chinese Academy of Sciences, Beijing 100049, China

* Correspondence: liujian@ime.ac.cn

Abstract: In this paper, we propose an efficient architecture of floating-point square-root circuit with low area cost, which is in accordance with the IEEE-754 standard. We extend the principle of the standard SRT algorithm so that the latency and area cost of the proposed circuit are linear with the radix. In addition, no extra computation cycles are required. With 65 nm technology, the area cost of the single-precision floating-point square-root circuit based on proposed architecture is only 6450.84 μm^2 , and the dynamic power consumption is only 0.764 mW at 300 MHz. The implementation results show that the proposed square-root circuit can reduce the area cost by 60%–90% compared with other designs in the literature.

Keywords: high-radix; square-root; SRT; floating-point



Citation: Yang, Y.; Yuan, Q.; Liu, J. A Low-Cost High Radix Floating-Point Square-Root Circuit. *Electronics* **2021**, *10*, 1988. <https://doi.org/10.3390/electronics10161988>

Academic Editors: Fabian Khateb and Spiros Nikolaidis

Received: 28 June 2021

Accepted: 16 August 2021

Published: 18 August 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Although square-root operation is not commonly used compared with other arithmetic operation, many instruction set architectures (ISA) include square-root instruction, such as ARM, x86, or RISC-V ISAs. Compared with addition or multiplication units, square-root circuit usually has higher complexity and longer latency. The common algorithms for square-root operation are SRT, Goldschmidt, Taylor-series, or Newton-Raphson algorithm [1–4], which can be divided into two categories: multiplication-based approximation algorithms and digital recursive algorithms. It is a challenge to implement an efficient floating-point square-root operation on hardware, which needs to balance the computing performance, area cost, and power consumption etc.

The multiplication-based square-root algorithms (e.g., Newton–Raphson algorithm) are usually approximated by inverse operation. The results of these algorithms are not obtained digit-by-digit. Instead, the calculation accuracies are improved step-by-step through the multiplication and addition operations to get the final results. The convergence rate of these algorithms is quadratic [5], and the algorithms usually have higher computational efficiency. In order to support the iterative calculations, we need more independent multiplier and adder hardware resources. Hence, the timing performance of the implemented circuit is limited by the latency of multiplier. In addition, the mainstream processors usually adopt the IEEE-754 standard, which makes the rounding operation of multiplication-based algorithms more complicated and more difficult to obtain the remainder.

Compared with the multiplication-based algorithms, the digital recursive algorithms need more iteration cycles, and the convergence is linear [6]. In each iteration cycle, the partial square-root digits with fixed bit-width can be obtained. At present, the most widely used digital recursive algorithm is SRT, in Intel or IBM [7,8] processor cores, the SRT algorithm with lower radix is used to implement square-root circuit. In the standard SRT algorithm, although the higher radix can improve the computational performance, the area cost of the lookup table increases in quadratic with the radix [9]. For instance, Synopsys Design-Ware can provide a single-precision square-root circuit based on SRT-16, and the area cost is about 29 K equivalent gates. In [2], the square-root circuit is implemented based

on SRT-8 algorithm, and the main contribution of the work is to optimize the area cost of the lookup table. However, the area cost of the optimized lookup table still accounts for about 30% of the total square-root circuit. Even if SRT algorithm with lower radix is adopted, it still needs a large area size. Taking IBM-z990 as an example, the SRT-16 square-root circuit implemented by SRT-4 cascading structure still needs about $2.5 \text{ W}\mu\text{m}^2$ even if it adopts 40 nm technology [10].

In order to tackle the above-mentioned design challenges, we extend the principle of standard SRT algorithm and optimized the iterative process. Specifically, we get the partial square-root digits with fixed bit-width by an estimation circuit in each iteration cycle instead of the lookup table, and the errors of estimated square-root digits can be detected and corrected in current cycle to ensure the error will not propagate in next iteration cycle and obtain accurate calculation results. Through the optimized iterative process, the error detection and correction do not need additional cycles. Compared with the standard SRT algorithm with the same radix, the proposed circuit has the same calculation cycles and can effectively reduce the area cost.

The rest of the paper is organized as follows. In Section 2, we discuss the standard SRT algorithm to achieve floating-point square-root operation process. Section 3 explains the proposed algorithm and provides a mathematical analysis. In Section 4, we describe the architecture of the proposed square-root circuit. Section 5 provides the implementation results and comparisons. Finally, concluding remarks are given in Section 6.

2. SRT Algorithm Analysis

According to IEEE-754 standard, any single-precision floating-point number $X = Y \times 2^e$, where $Y \in [1, 2)$ is the 23-bit mantissa code, and $e \in [-128, 127]$ is exponent code of 8-bit. The square-root operation result of X is $\sqrt{X} = \sqrt{Y \times 2^e} = \sqrt{Y} \times 2^{e/2}$, where $e/2$ can be realized by 1-bit right shift operation. If e is an odd number, it is necessary to shift Y by 1-bit to obtain Y^* for mantissa square-root operation. At this time, the mantissa $Y^* \in [2, 4)$ and the result is less than 2, it also complies with IEEE-754 standard. The square-root operation of X is converted into the square-root calculation of Y , and Y can be expressed further as (1), where S is the square-root digits and P is the remainder after the finite precision square-root operation.

$$Y = S^2 + P \quad (1)$$

In the standard SRT algorithm with radix- r , (1) can be calculated in an iterative manner by shift and subtraction operation. In each iteration cycle, the $L = \log_2 r$ bit-width partial square-root digits can be achieved, after n iteration times, the square-root result S can be expressed as (2), and the remainder P can be expressed as (3), where w_i is the partial square-root digits generated in the i -th iteration with L bit-width.

Combining (2) and (3), the iterative Formula (4) of partial remainder P_{n+1} can be obtained, in the standard SRT algorithm, $w_{n+1} = \text{select}(S_n, P_n)$ is lookup table function. Generally, it is necessary to construct with the P-D graph [11].

$$S_n = \sum_{i=0}^n w_i \times r^{-i} \quad (2)$$

$$P_n = r^n (Y - S_n^2) \quad (3)$$

Formula (4) is the basic iterative process of standard SRT algorithm, in which $w_{n+1} = \text{select}(S_n, P_n)$ is usually implemented in ROM. It can be seen from (4) that the radix- r is proportional to the performance of the algorithm. With the increase of r , the bit-width of the partial square-roots digits obtained increases in each iteration, and the cycles of iteration required decrease. The calculation accuracy of SRT algorithm is 1 ULP (unit in last place). The latency of data path in square-root circuit based on standard SRT algorithm

increases linearly with r , while the area cost of the lookup table increases quadratically with r [5].

$$\begin{aligned} P_{n+1} &= r^{n+1} (Y - S_{n+1}^2) = r^{n+1} [Y - (S_n + w_{n+1} r^{-n-1})^2] \\ &= r^{n+1} \left[(Y - S_n^2) - 2S_n w_{n+1} r^{-n-1} - w_{n+1}^2 r^{-2(n+1)} \right] \\ &= rP_n - [w_{n+1} \times (2S_n + w_{n+1} r^{-n-1})] \end{aligned} \quad (4)$$

The area cost of the lookup table increases about four times with the increase of one bit-width of the partial square-root digits [11,12]. Table 1 shows the area cost of lookup table (implemented by ROM) in standard SRT algorithm with different radices. The area cost, as given in Table 1, adopts 65 nm technology, and under the same technology, the area of a NAND2 cell is $1.8 \mu\text{m} \times 1.4 \mu\text{m}$. It can be seen that with the increase of radix, the area cost of lookup table increases greatly, which limits the application of high radix standard SRT algorithm.

Table 1. Synthesize evaluation of area cost of the lookup table circuit (ROM) with different radices.

Radix	16	32	64	128	256
Area (μm^2)	4216.0	8028.7	20,220.5	94,179.0	376,719.8

3. The Proposed Square-Root Algorithm

In order to solve the problem of the large area cost of lookup table in standard high radix SRT algorithm, we adopt the cascade non-recovery remainder division with a short bit-width to replace the lookup table which is the standard SRT algorithm.

In standard SRT algorithm with radix- r , the partial square-root digits of $L = \log_2 r$ bits are achieved in each iteration. The proposed partial square-root digits estimation algorithm can be expressed as (5) and (6), all parameters are expressed in binary, where p^* is the highest $2L$ digits of the partial remainder generated in the previous iteration cycle, p_{2L-i-1}^* is the $2L - i - 1$ -th digit of p^* , u_0^* is the highest L digits of P_n , s^* is the highest L digits of S_n , and w_{n+1}^* represents the estimated value of partial square-root digits with L bit-width.

$$w_{n+1}^* = \sum_{i=0}^{L-1} 2^{-i} \text{sign}(u_i^* - s^*) \quad (5)$$

$$u_i^* = \begin{cases} 2(u_{i-1}^* - s^*) + p_{2L-i-1}^*, & u_{i-1} \geq 0 \\ 2(u_{i-1}^* + s^*) + p_{2L-i-1}^*, & u_{i-1} < 0 \end{cases} \quad (6)$$

In (5), the L -bit partial square-root digits can be obtained by the cascaded non-recoverable division. In addition, in (6) only the addition or subtraction operation with L -bit is needed, compared with standard SRT algorithm, only the full-adder with $2L$ bit-width is needed.

However, it should be pointed out that errors may occur due to the lack of full-precision operands in (5) and (6). Therefore, it is necessary to extend the iterative process of the standard SRT algorithm and correct the errors in time to avoid the errors propagation in the iterative process.

$\Delta w = w_n - w_n^*$ is the errors between the estimated value and the true value of partial square-root digits in the proposed algorithm, the true value of the partial remainder is shown in (7), the estimated value is shown in (8), and the errors of the partial remainder ΔP can be expressed by (9):

$$P_n = rP_{n-1} - [w_n \times (2S_{n-1} + w_n r^{-n})] \quad (7)$$

$$P_n^* = rP_{n-1} - [w_n^* \times (2S_{n-1} + w_n^* r^{-n})] \quad (8)$$

$$\begin{aligned}
\Delta P &= P_n - P_n^* \\
&= w_n^*(2S_{n-1} + w_n^*r^{-n}) - w_n(2S_{n-1} + w_nr^{-n}) \\
&= w_n^*2S_{n-1} - w_n2S_{n-1} + w_n^*2r^{-n} - w_n^2r^{-n} \\
&= 2S_{n-1}(w_n^* - w_n) + (w_n^* - w_n)(w_n^* + w_n)r^{-n} \\
&= (w_n^* - w_n)(2S_{n-1} + w_n^*r^{-n} + w_nr^{-n}) \\
&= -\Delta w(2S_{n-1} + w_n^*r^{-n} + w_nr^{-n})
\end{aligned} \tag{9}$$

Substituting (9) into the basic recursive Formula (4) of the standard SRT algorithm, the relationship between the estimated value P_n^* and the real value P_{n+1} generated in the next iteration is shown in (10):

$$\begin{aligned}
P_{n+1} &= rP_n - [w_{n+1}(2S_n + w_{n+1}r^{-n-1})] \\
&= r[P_n^* - \Delta w(2S_{n-1} + w_n^*r^{-n} + w_nr^{-n})] - [w_{n+1}(2S_n + w_{n+1}r^{-n-1})] \\
&= r[P_n^* - \Delta w(2S_{n-1} + 2w_nr^{-n} + w_n^*r^{-n} - w_nr^{-n})] - [w_{n+1}(2S_n + w_{n+1}r^{-n-1})] \\
&= r[P_n^* - \Delta w(2S_{n-1} + 2w_nr^{-n} - \Delta wr^{-n})] - [w_{n+1}(2S_n + w_{n+1}r^{-n-1})] \\
&= r[P_n^* - \Delta w(2S_n - \Delta wr^{-n})] - [w_{n+1}(2S_n + w_{n+1}r^{-n-1})] \\
&= rP_n^* - \Delta w(2S_n - \Delta wr^{-n})r - w_{n+1}(2S_n + w_{n+1}r^{-n-1})
\end{aligned} \tag{10}$$

Considering the general iteration process of digital recursion algorithm to analyze the error conditions of the w_n^* . Assuming that m represents the full-precision bit-width of P and S , the highest L digits of P and S can be represented as $d_p = \sum_{i=0}^{L-1} 2^{-i}p_{m-1-i}$,

$d_s = 1 + \sum_{i=1}^{L-1} 2^{-i}s_{m-1-i}$ respectively, and the remaining digits can be represented as $\Delta p = \sum_{i=L}^{m-1} 2^{-i}p_{m-1-i}$ and $\Delta s = \sum_{i=L}^{m-1} 2^{-i}s_{m-1-i}$ respectively, where $\Delta p, \Delta s \in [0, 2^{-m+L} - 1]$.

Therefore, the real value of the P and S can be represented as $P = d_p + \Delta p$ and $S = d_s + \Delta s$. According to (6), only the highest L digits of the operands are used for calculation in each estimation cycle, when $d_p > d_s$ or $d_p < d_s$, Equation (5) can obtain the real value of the partial square-root digits by the highest L digits of the two operands, while the remaining digits Δp and Δs do not affect the results. When $d_p = d_s$, the result of the true value depends on the digits of the remaining digits. If $\Delta p \geq \Delta s$, then the estimated result of (5) and the real result are both "1", and there is no error in the estimated result. When $\Delta p < \Delta s$, the estimated result of (5) is "1", but the real result of square-root digit is "0". In this case, because the digits of residual value are not included in the estimation process of (6), the calculation error is generated, and in the worst case, when $i = 0$, the generated error in (6) accumulates in the calculation of the next stage non-recovery remainder division and the maximum error accumulation is caused.

In order to achieve the results in accordance with IEEE-754 standard, it is necessary to analyze the maximum errors of the estimated partial square-root digits quantitatively. Assuming that in the worst case, P and S satisfy the following conditions: $d_p = d_s = d$, and $\Delta p < \Delta s$, we can achieve $d > 2^{L-1}\Delta s > 2^{L-1}\Delta p$, and $P < S$. In the calculation process of full-precision, the partial square-root digits w with L bit-width can be expressed as (11):

$$w = \sum_{i=0}^{L-1} 2^{-i} \text{sign}(U_i - S) \tag{11}$$

In (11), U_i is the partial remainder generated in the iterative calculation process, and U_i can be expressed by (12), where $U_0 = P$.

$$U_i = \begin{cases} 2U_{i-1} - S, & 2U_{i-1} \geq S \\ 2U_i, & 2U_{i-1} < S \end{cases}, i \in [1, L-1] \tag{12}$$

Substituting P , S , and d into (12), after L times of iterative calculation, the partial remainder corresponding to the square-root digits of the real value is obtained: $U_{L-1} = d + 2^{L-1}\Delta p - (2^{L-1} - 1)\Delta s$.

In the proposed algorithm the estimated partial square-root digits w^* can be expressed as (13), where $S^* = d$ and U_i^* is the estimated partial remainder with L bit-width, and the calculation process of U_i^* can be expressed as (14), where $U_0^* = P - S^* = \Delta p$.

$$w^* = \sum_{i=0}^{L-1} 2^{-i} \text{sign}(U_i^* - S^*) \quad (13)$$

$$U_i^* = \begin{cases} 2U_{i-1}^* - S^*, & U_{i-1}^* \geq 0 \\ 2U_{i-1}^* + S^*, & U_{i-1}^* < 0 \end{cases}, i \in [1, L-1] \quad (14)$$

Substituting P , S^* and d into (14), after L times of iteration, we can get that the remainder with maximum error corresponding to the estimated partial square-root digits w^* is $U_{L-1}^* = 2^{L-1}\Delta p - d$.

We can get the errors between the real value remainder U_{L-1} and the estimated remainder U_{L-1}^* can be expressed as: $\Delta U = U_{L-1} - U_{L-1}^* = 2d - (2^{L-1} - 1)\Delta s$. From the constraints $\Delta s > 0$ and $d > 2^{L-1}\Delta s$, we can get the following conclusions: $\Delta U > 0$, $\Delta U \in (d, 2d)$, $w^* > w$, $\Delta w = w - w^* \in [-1, 0]$.

Through the above quantitative analysis, we can get the relationship between the estimated value and the real value of the partial square-root digits with L bit-width, which can be expressed as (15):

$$w = \begin{cases} w^* - 1, & U_{L-1}^* < 0 \\ w^*, & U_{L-1}^* \geq 0 \end{cases} \quad (15)$$

In (15), the error of the estimated partial square-root digits can be corrected by a L -bit subtracter, and when the error occurs, $\Delta w = -1$, the errors of estimated partial remainder can be obtained, and the correction process is shown in (16):

$$\begin{aligned} P_{n+1} &= rP_n^* + (2S_n + r^{-n})r - w_{n+1}2S_n - w_{n+1}^2r^{-n-1} \\ &= rP_n^* + (r - w_{n+1}) \times [2S_n + (r + w_{n+1})r^{-n-1}] \end{aligned} \quad (16)$$

According to the constraint condition $w_n \in [0, r)$ of SRT algorithm, in the correction process described in (16), $r - w_{n+1}$ can be realized by a simple subtracter with L bit-width, $2S_n + (r + w_{n+1})r^{-n-1}$ can be realized directly by bit splicing operation. Therefore, compared with the standard SRT square-root algorithm in (4), Only one subtracter with L bit-width is added in (16). The proposed square-root algorithm can be summarized as (17)–(19):

$$k_{n+1} = \begin{cases} -w_{n+1}^*, & P_n \geq 0 \\ r - w_{n+1}^*, & P_n < 0 \end{cases} \quad (17)$$

$$H_n = \begin{cases} -2S_n - w_{n+1}^*r^{-n-1}, & P_n \geq 0 \\ 2S_n + (r + w_{n+1}^*)r^{-n-1}, & P_n < 0 \end{cases} \quad (18)$$

$$P_{n+1} = rP_n + H_n \times k_{n+1} \quad (19)$$

Compared with the standard SRT algorithm, the proposed square-root algorithm avoids the use of lookup table, and has a general expression, which can be extended to the design of square-root circuit with any radix.

4. Proposed Square-Root Architecture

According to Equations (17)–(19), a single-precision floating-point square-root circuit with radix-16 is designed. The structure of mantissa iteration is shown in Figure 1. The structure is similar to the square-root circuit base on standard SRT algorithm. In Figure 1, a partial square-root digits estimation circuit is used to replace the lookup table in the standard SRT algorithm. The partial square-root digits correction circuit and the k_n , H_n correction circuit corresponding to (17) and (18) are added. The necessary adders and multipliers in the standard SRT algorithm are also included.

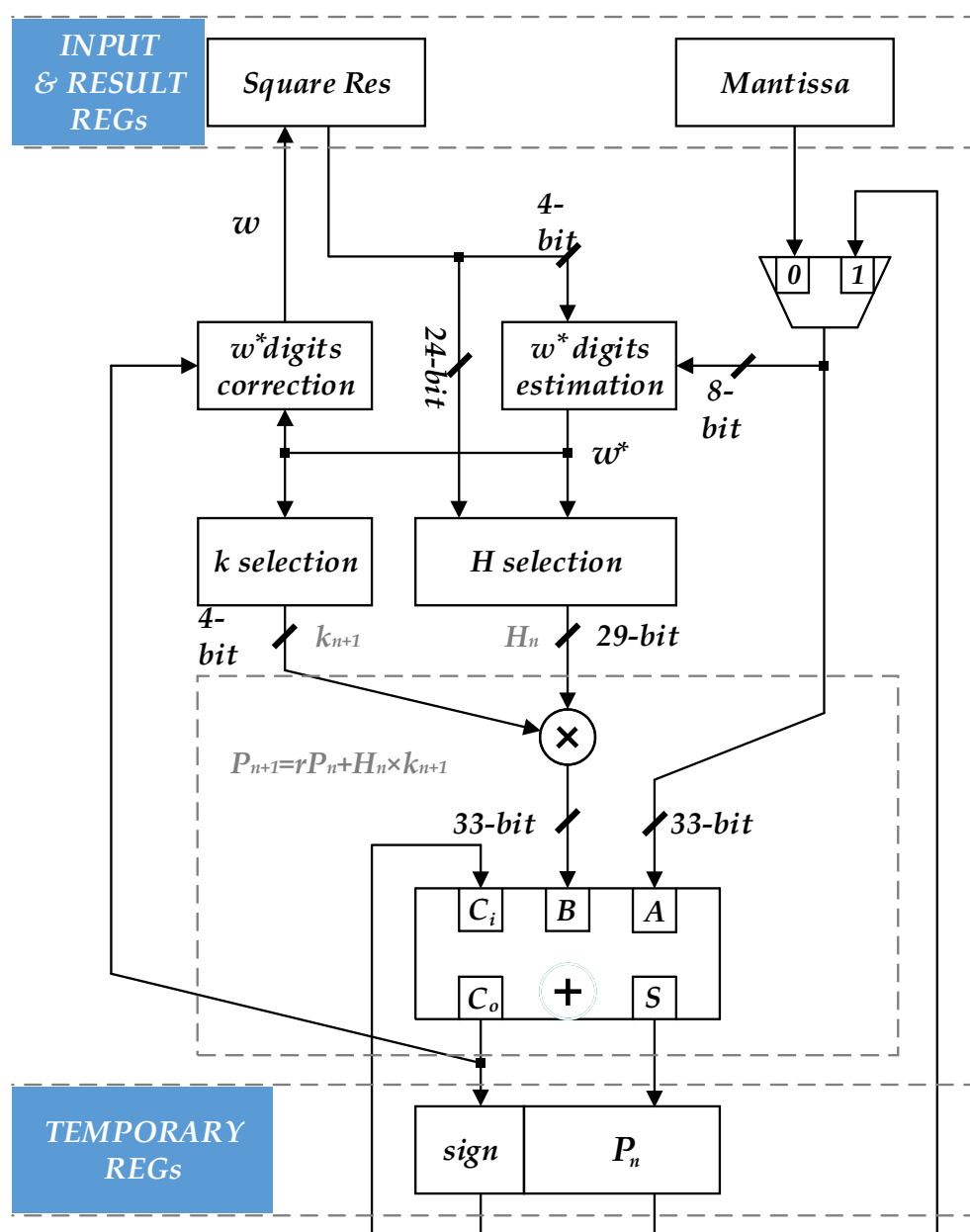


Figure 1. The structure of the proposed mantissa square-root.

The mantissa iterative circuit shown in Figure 1 can generate 4-bit partial square-root digits in each iteration cycle. In order to support the 4 rounding modes specified in IEEE-754 standard, it will takes 7 cycles to perform a single-precision square-root operation to obtain the complete mantissa, while the rounding operation requires an additional cycle to be calculated separately.

Combining (17) and taking into account the rounding mode specified in IEEE-754 standard, the one input of multiplier in Figure 1 is 4-bit, while the other bit-width of the input needs 33 to ensure the correctness of the result, and the bit-width of the adder also needs 33 to complete the calculation result in the last iteration cycle.

According to (13) and (14), the proposed partial square-root digits estimation circuit is shown in Figure 2. Where U is the highest 8-bit of the partial remainder, and S^* is the highest 4-bit of the current square-root result. In each iteration cycle, the circuit can achieve 4-bit partial square-root digits.

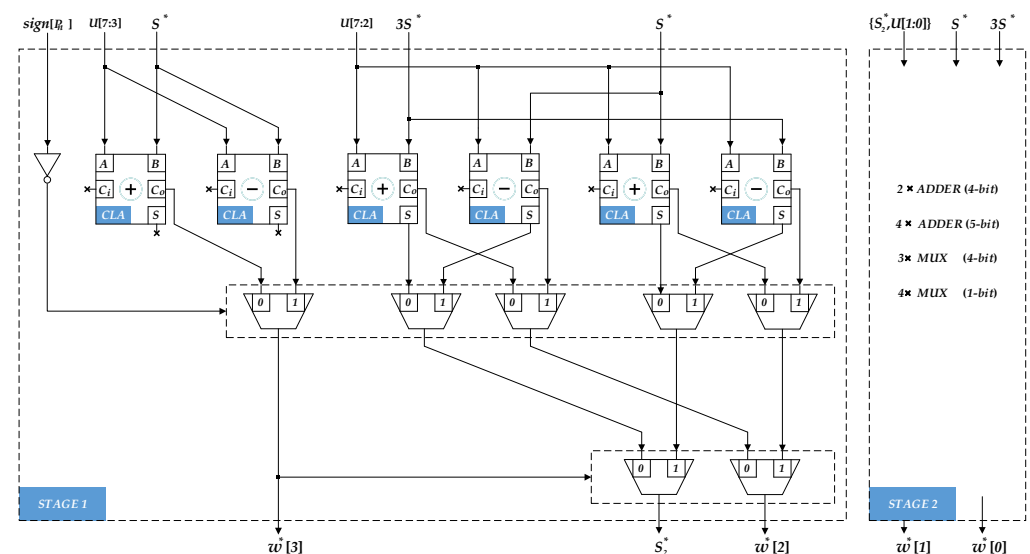


Figure 2. The structure of the proposed partial square-root digits estimation circuit.

It can be seen from (14): the estimation process of partial can be composed of 4-stage cascaded full-adders, but even if the carry look-ahead adder is used, it still needs 4-stage full-adder latency to get 4-bit partial square-root digits. In order to obtain a better timing performance, the structure of estimation circuit is improved in this paper. First, for the estimation process of each square-root digit in (14), the composite adder is used instead of the full-adder, so that the addition and subtraction are carried out independently, and the result of the addition/subtraction operation is selected according to the sign of the previous stage. The carry-in delay of full-adder is reduced. In addition, the secondary operation is expanded from (14) to (20):

$$\begin{aligned}
 U_n^* &= 2U_{n-1}^* \pm S^* \\
 U_{n+1}^* &= 2U_n^* \pm S^* \Rightarrow \begin{cases} 4U_{n-1}^* \pm 3S^*, U_n^* \geq 0 \\ 4U_{n-1}^* \pm S^*, U_n^* < 0 \end{cases} \quad (20)
 \end{aligned}$$

It can be seen from (20) that by decomposing the secondary adder into two independent adders, the execution process of the secondary adder can be carried out simultaneously with the former adder. When the current stage adder obtains the final result, the next result will be obtained only after the latency of the one level 2-input multiplexer. Each cascade structure can reduce the latency of about one level adder, and the advantage of this circuit will be more obvious in higher radix structures.

Table 2 shows the latency and area cost evaluation results of partial square-root digits estimation circuits with different radices. The evaluation conditions are the worst process angle (voltage in 1.08 V, temperature in 125 °C) with 65 nm technology. From the data given in Table 2, it can be seen that the critical path delay and area cost of the proposed partial square-root digits estimation circuit both almost increase linearly with the radix. Compared with the standard algorithm, the radix-256 partial square-root digits estimation circuit is only 1.8 K gates.

Table 2. Synthesize evaluation of the proposed partial square-root digits estimation circuit with different radices.

Radix	16	32	64	128	256
Crit. Path (ns)	1.268	1.608	2.196	2.557	3.151
Leaf Cell Count	317	583	744	1123	1796
Area (μm ²)	715.6	1297.8	1736.3	2736.7	3832.9

In order to achieve accurate results, it is necessary to detect and correct the error of the partial square-root digits and the partial remainder in iteration process. According to (15), the correction circuit of partial square-root digits can be realized by a subtractor with 4-bit. Based on (16), the coefficient k correction circuit output estimated value w^* or $r - w^*$ according to the sign of partial remainder in the previous iteration, and the result is still 4-bit.

According to (18), it can be seen that in the iterative process, the coefficient H can be realized by bit splicing, which is composed of S_n 5-bit after left shift and 4-bit w^* . When $P_n \geq 0$, the 5th digit of H is fixed to "0", while when $P_n < 0$, the 5th digit is fixed to "1". The structure of the coefficient H correction circuit is shown in Figure 3. Since the bit-width of the partial square-root digits is increased by 4 in each cycle, the splicing operation of the coefficient H needs to go through 7 cycles, adding a total of 3 levels latency of 2-input multiplexer. Since the correction operation of the coefficient k and H is carried out in parallel. The latency caused by the correction circuit is about the latency of 1 level 4-bit adder or 3 levels latency of 2-input multiplexer.

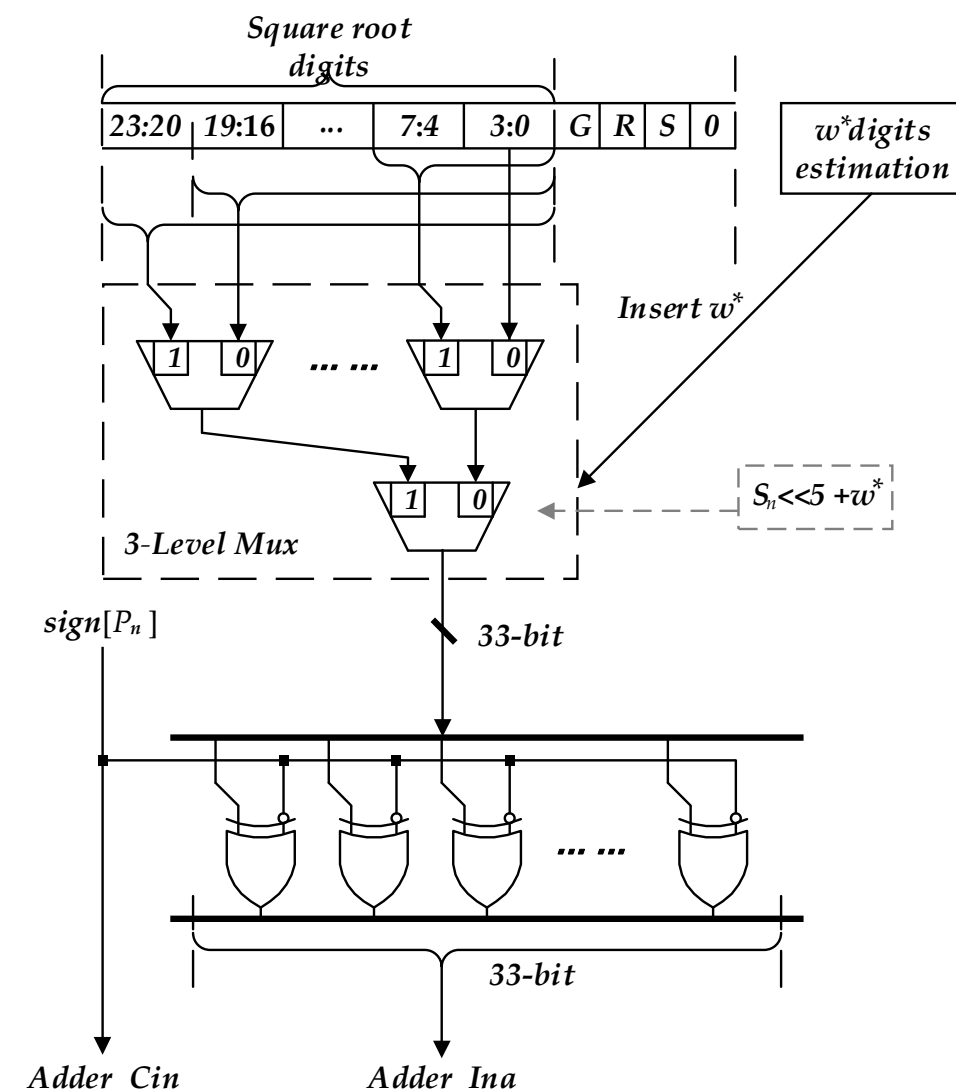


Figure 3. The structure of the multiplier correction circuit.

According to (18) and (19), the sign of the partial remainder is generated according to the previous iteration, and the partial remainder correction circuit will output $-k \times H$ or $k \times H$. Therefore, in the above two cases, the addition or subtraction operations need to be performed respectively. It means the independent adder and subtractor are implemented. As shown in Figure 3, we use the characteristics of the full-adder to solve the above problem.

When the addition operation is performed, the input operands of the adder is $k \times H$, and the carry-in is 0. When the subtraction operation is performed, the input operands is $\sim (k \times H) + 1$, where $\sim (k \times Y)$ is an inversion operation, which can be implemented by parallel XOR gates. The operation of the additional “1” to the complement code can be used as the carry-in of the full-adder. The structure in Figure 3 can preform the partial remainder correction operation without increasing the adder resources, and the latency only increased by one level XOR gate.

5. Implement Results and Comparison

In order to get accurate evaluation results, we use Synopsys Design-Compiler to get the synthesis results of the proposed square-root circuit in 65 nm technology. Table 3 shows the synthesis results under the worst process angle (1.08 V, 125 °C), clock frequency is 300 MHz.

Table 3. Synthesis results of the proposed square-root circuit with different radices.

Radix	Precision	Area (μm^2)	Leaf Cell Count	Crit. Path (ns)	Cycles	Power (mW)
16	32	6450.84	2850	3.1899	8	0.764
	64	9437.42	4270	3.3693	15	1.224
64	32	9199.08	4065	5.0232	6	1.522
256	32	12,017.88	5309	6.1370	5	1.716

The calculation period given in Table 3 depends on the bit-width of the operand. In order to support the 4 rounding modes specified in IEEE-754 standard, sufficient square-root digits must be obtained in the iteration process. For the single-precision floating-point operand, at least 27 bits of square-root digits should be obtained, including 24 bits of standard square-root digits and 3 bits of rounding digits (guard bit, rounding bit and stick bit). Double-precision floating-point operand requires at least 56 bits of square-root digits, including 53 bits of standard square-root digits and 3 bits of rounding digits.

In addition, in order to provide a fair comparison with the results in other reports, the implementation results with different calculation precisions and different radices based on the proposed architecture are given in Table 3. For the area cost of the square-root circuit, Table 3 gives two expressions: the leaf cell count and the cell areas.

Combining the area cost of lookup table shown in Table 1 with the area data shown in Table 3, the advantages of the proposed design in area cost can be illustrated. When the radix is 64, only the lookup table (ROM) will cost $20,220.5 \mu\text{m}^2$, while the overall area cost of the proposed square-root circuit is $9199.08 \mu\text{m}^2$, which is only about 45% of the lookup table in the standard SRT algorithm. When the radix is 256, the area cost of the lookup table is $376,719.8 \mu\text{m}^2$ while the area cost of the proposed square-root circuit is $12,017.88 \mu\text{m}^2$, which is only about 3% of the lookup table circuit.

Through the comparison of the area data in Tables 1 and 3, it can be seen that the huge area cost of the lookup table limits the application of the standard SRT algorithm in high radix square-root circuits. Therefore, in the design of high radix square-root circuit, the proposed architecture will have more obvious advantages in area cost. In addition, the bottleneck of high radix SRT square-root circuit is also solved.

Figure 4 shows the detailed function waveform of the square-root circuit with radix-16 and 32 bits based on the proposed architecture. The meanings of the signals are summarized as follows: “i_div_a” is the input data; “o_div_r” is the result output; “o_div_hskd” is the valid indication signal of result; “man_sub_o” is the estimated value of partial square-root digits, which corresponds to w^* in Equation (13); “man_qds_o” is the correction value of partial square-root digits, which corresponds to w_n in Equation (15); “rem_add_a” is the partial remainder generated in the previous iteration, which corresponds to rP_n in Equation (19); “rem_add_o” is the partial remainder generated in current iteration, which corresponds to P_{n+1} in Equation (19); “rem_mul_o” corresponds to $H_n \times k_{n+1}$ in Equation (19) and “div_cnt_r” is a counter, which displays the calculation cycle and is used to control the iteration process.

<i>i_div_a</i>	32'h4956c102	4956c102				
<i>i_div_en</i>	1'h1					
<i>i_div_rm</i>	2'h0	0				
<i>o_div_hskd</i>	1'h0					
<i>o_div_r</i>	32'h12c10200	f94cf000	0a0b7000	0e2b8670	0e2b8677	446a78c5
<i>div_cnt_r</i>	3'd0	3	4	5	6	7
<i>exp_a</i>	8'd19	19				
<i>exp_sqrt_o</i>	8'd9	9				
<i>div_res_r</i>	25'h0000000	0000ea7	100ea78	00ea78c	0ea78c5	
<i>man_sub_o</i>	4'he	9	c	5	8	
<i>man_qds_o</i>	4'he	8	c	5	7	8
<i>rem_add_a</i>	33'h0d6c10200	101100000	194cf0000	0a0b70000	0e2b86700	000ea78c5
<i>rem_mul_o</i>	33'h0c4000000	107c31000	0753c7000	0928b7990	0ea78c540	000000000
<i>rem_add_o</i>	33'h012c10200	1f94cf000	00a0b7000	00e2b8670	1f83fa1c0	000ea78c5
<i>i_clk</i>	1'h0					

Figure 4. The simulation waveform of the proposed square-root circuit.

In Figure 4, the decimal floating point number input is 879,632.125, the hexadecimal representation is $0 \times 4956_C102$, the result of square-root value is 937.887, and the hexadecimal floating point number is $0 \times 446A_78C5$. As can be seen in Figure 4, after 8 cycles of iterative calculation, the proposed square-root circuit can obtain correct results.

Moreover, when the partial remainder generated in current iteration cycle is negative, it indicates that there is an error in the w^* , and w can be corrected in current cycle, and the errors of partial remainder can be corrected in the next iteration through the circuit corresponding to Equation (18). The correction positions of the partial remainder and square-root digits are marked in the waveform of Figure 4.

Figure 5 shows the comparison of the calculation cycle between the proposed square-root circuit and other mainstream processor. It must be pointed out that the comparison of calculation performance in Figure 5 is only limited to the calculation cycles, without considering the technology, frequency, area cost, or power consumption in different processors.

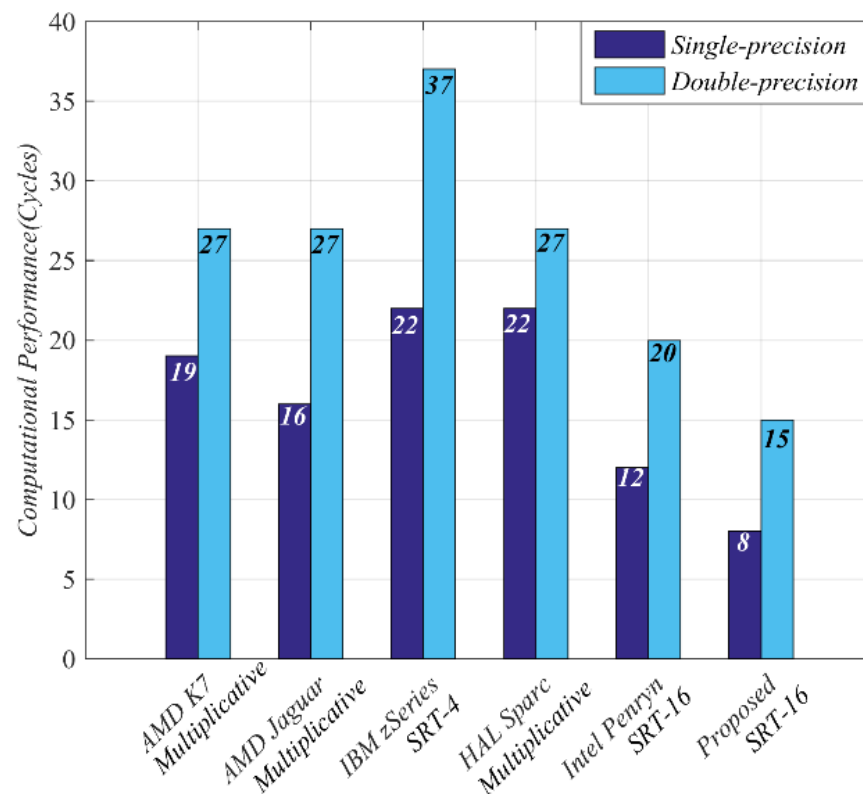
As shown in Figure 5, The performance of the square-root circuits based on multiplication operation is slightly higher than that of SRT-4 algorithm, and the performance of circuits is mainly limited by the latency of multiplication and accumulation units. Compared with SRT-4 algorithm, SRT-16 algorithm can get double bit-width of the square-root digits in each cycle, and the computational performance can be greatly improved. However, it can be seen from the algorithm implemented based on SRT in Figure 5 that in the standard SRT algorithm, the area cost of lookup table of higher radix also limits the application in processor design. Even in Intel Penryn processor, the structure of SRT-4 cascade is used to implement SRT-16 algorithm.

Although the square-root circuit based on multiplication can improve the circuit performance, it increases the throughput of square-root unit to 1 cycle by the pipelined structure. However, the mainstream processors in Figure 5 all adopt the iterative structure to reduce the penalty of pipeline clearing caused by missed branch prediction. It also shows that the proposed structure of square-root circuit is more suitable for the low-speed processor design based on RSIC-V instruction architecture set. In addition, in the comparison of computational performance in Figure 5, the proposed structure proposed achieves lesser computational cycles.

Table 4 lists the comparison with other square-root circuits based on SRT algorithm, including the comparison of area cost (cell area and leaf cell count), operand precision, and power consumption. Considering the different technologies and frequencies used between different designs, in order to provide more fair comparison, the ratio of power consumption to frequency is provided as a reference for the comparison on power consumption.

Table 4. Comparison with others square-root circuit based on SRT algorithms.

	[13]	[14]	[15]	[16]	[17]	Proposed	
Radix	16	8	16	16	4	16	
Area (μm^2)	-	18,338	15,147	-	45,446	6451	9437
Leaf Cell Count	11,250	-	-	59,685	3450	2850	4270
Power ($\mu\text{W}/\text{MHz}$)	32.4	4.605	4.206	30.0	139.692	2.547	4.081
Precision	64	64	64	64	32	32	64
Cycles	16	-	16	16	14	8	15
Technology	90 nm	40 nm	40 nm	90 nm	180 nm	65 nm	
Critical Path (ns)	1.2	1.04	0.656	1.08	4.0	3.19	3.37
Frequency (MHz)	833.3	961.5	1524	925.9	250	300	296.7
Throughput (MB/s)	416.65	-	762	462.95	71.6	150	158.2
Latency (ns)	19.2	-	10.5	17.3	56	25.5	50.6

**Figure 5.** Comparison with others square-root circuit [18].

Through the comparison of calculation performance in Table 4, it can be seen that the area cost of this paper is reduced by 37.69% compared with [15]. It should be noted that [15] uses the 40-nm technology with smaller size, while this paper uses the 65-nm technology, if the shrinkage of technology size is considered, more area reduction will be achieved. In comparison with [13], the number of equivalent gates is reduced by 66.71%, even the area cost of the proposed circuit is only 6.27% of [16]. Compared with reference [17], the circuit area of this paper is smaller, but the calculation performance can be nearly doubled.

Even if the same circuit is implemented in different technology, the power consumption is obviously different. In Table 4, the power consumption of [13,16] implemented in 90 nm technology is about nine times than [14,15]. However, even compared with [14,15],

which are implemented in 40 nm technology, the proposed structure also achieves lower dynamic power under the same calculation cycles and precision. Therefore, the proposed square-root structure is also suitable for power sensitive processor design.

Latency in Table 4 represents the time required for the square-root circuit to complete calculation. It can be seen from Table 4 that the performance of the square-root circuit based on the proposed algorithm is only higher than [17] in terms of the maximum frequency and latency of the circuit. However, it should be pointed out that different process parameters (e.g., technology size, voltage, temperature, etc.) have a significant impact on the maximum frequency of the circuit.

In order to avoid the impact of different process parameters, the combinational logic depth of the critical path in the circuit is generally used to evaluate the performance of the circuit. In Table 4, the maximum logic depth of the square-root circuit with radix-16 and precisions of 32 and 64 are 33 and 41 levels, respectively. However, the data of the maximum logic depth is not given in other references. Therefore, the maximum frequency or performance cannot be directly compared across technology.

However, the maximum frequency and performance of the circuit can be indirectly compared according to the implementation structure of the algorithm. For the square-root circuits given in Table 4, the performance of the circuit is determined by two parameters: the maximum frequency and the calculation cycle. According to the principle of SRT algorithm, the higher the radix r , the lesser number of the iteration cycles required to complete the calculation, and the square-root circuit can achieve higher performance under the same frequency. Both [16,17] adopt the standard SRT algorithm and the lookup table structure. However, from the area comparison data, it can be seen that compared with [17], the radix of [16] is increased by four times, the circuit area cost is increased by 17 times, and the calculation performance is improved by only two cycles. Neither [14] nor [15] adopts the lookup table structure. Instead, the cascade structure of lower radix SRT square-root circuit is adopted to obtain a higher radix. Although the significant increase of circuit area is avoided, when the radix is doubled, the critical path delay of the corresponding circuit will also double.

According to the data in Table 3, when the radix of the square-root circuit based on the proposed algorithm is increased from 16 to 64, the circuit area cost is only increased by about 1.4 times, and the critical path delay is only increased by 1.5 times. Even when the radix is 256, the circuit area increases by only 1.8 times, and the critical path delay increases by only 1.9 times. Therefore, it can be seen from the data in Tables 3 and 4 that although there is a gap in frequency compared with other reports, the proposed square-root structure has better tradeoff between the area cost and frequency, and is more suitable for applications that are sensitive to power consumption and area cost.

6. Conclusions

In this paper, a novel architecture of floating-point square-root circuit based on SRT algorithm was proposed, in which the computational performance and the area cost are linear with the radix. In the proposed architecture, a partial square-root digits estimation circuit is applied to replace the lookup table in the standard SRT algorithm, which solves the design bottleneck of area cost in high radix SRT algorithm. The recursive process of standard SRT algorithm is extended, the estimation error of partial square-root digits and remainder can be corrected in time, and the error accumulation can be eliminated by using the non-recovery remainder division and full-adder. Compared with the standard SRT algorithm, the proposed algorithm does not need additional calculation cycles. Finally, we designed a floating-point square-root circuit with radix-16 in accordance with the IEEE-754 standard, and deploy it into the FPU of RISC-V processor core. Compared with other designs in the literature, the proposed floating-point square-root circuit can reduce the area cost significantly under the same operand precision and computational performance.

Author Contributions: Conceptualization, Y.Y.; Data curation, Q.Y.; Project administration, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Construction of High-Level Innovation Research Institute of Integrated Circuit and System Application in DaWan District of Guangdong Province under Grants Y9SN01K001 and 2019B090909006.

Acknowledgments: The authors would like to thank the reviewers and editors for their insightful comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ercegovic, M.D.; Matula, D.W.; Muller, J.M. Improving Goldschmidt Division, Square Root, and Square Root Reciprocal. *IEEE Trans. Comput.* **2000**, *49*, 759–763. [\[CrossRef\]](#)
2. Aguilera-Galicia, C.R.; Longoria-Gandara, O. IEEE-754 Half-Precision Floating-Point Low-Latency Reciprocal Square Root IP-Core. In Proceedings of the 2018 IEEE 10th Latin-American Conference on Communications (LATINCOM), Guadalajara, Mexico, 14–16 November 2018.
3. Soderquist, P.; Leiser, M. Division and square root choosing the right implementation. *IEEE Micro* **1997**, *17*, 56–66. [\[CrossRef\]](#)
4. Kwon, T.-J.; Draper, J. Floating-point division and square root using a taylor-series expansion algorithm. *Microelectron. J.* **2009**, *40*, 1601–1605. [\[CrossRef\]](#)
5. Ramamoorthy, C.V.; Goodman, J.R.; Kim, K.H. Some Properties of Iterative Square-Rooting Methods Using High-Speed Multiplication. *IEEE Trans. Comput.* **1972**, *21*, 837–847. [\[CrossRef\]](#)
6. Hobson, R.F.; Fraser, M.W. An Efficient Maximum-Redundancy Radix-8 SRT Division and Square-Root Method. *IEEE J. Solid-State Circuits* **1995**, *30*, 29–38. [\[CrossRef\]](#)
7. Baliga, H.; Cooray, N.; Gamsaragan, E.; Smith, P. Improvements in the Intel Core2 Penryn Processor Family Architecture and Microarchitecture. *Int. Technol. J.* **2008**, *12*, 179–192.
8. Lichtenau, C.; Carlough, S.; Mueller, S.M. Quad precision floating point on the IBM z13TM. In Proceedings of the 23rd Symposium on Computer Arithmetic (ARITH), Silicon Valley, CA, USA, 10–13 July 2016; pp. 87–94.
9. Oberman, S.F.; Flynn, M.J. Minimizing the complexity of SRT tables. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **1998**, *6*, 141–149. [\[CrossRef\]](#)
10. Wetter, H.; Schwarz, E.M.; Haess, J. The IBM eServer z990 floating-point unit. *IBM J. Res. Dev.* **2004**, *48*, 311–322.
11. Russinoff, D.M. Computation and formal verification of SRT quotient and square root digit selection tables. *IEEE Trans. Comput.* **2013**, *62*, 900–913. [\[CrossRef\]](#)
12. Savas, S.; Atwa, Y.; Nordström, T. Using Harmonized Parabolic Synthesis to Implement a Single-Precision Floating-Point Square Root Unit. In Proceedings of the 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Miami, FL, USA, 15–17 July 2019.
13. Liu, W.; Nannarelli, A. Power Efficient Division and Square Root Unit. *IEEE Trans. Comput.* **2012**, *61*, 1059–1070. [\[CrossRef\]](#)
14. Rust, I.; Noll, T.G. A Digit-Set-Interleaved Radix-8 Division/Square Root Kernel for Double-Precision Floating Point. In Proceedings of the 2010 International Symposium on System on Chip, Tampere, Finland, 29–30 September 2010.
15. Yuanxi, P.; Jiyang, C.; Yuanwu, L. Low-Latency SRT Division and Square Root Based on Remainder and Quotient Prediction. *Chin. J. Electron.* **2017**, *26*, 58–64.
16. Nannarelli, A. Radix-16 Combined Division and Square Root Unit. In Proceedings of the 2011 IEEE 20th Symposium on Computer Arithmetic, Tuebingen, Germany, 25–27 July 2011.
17. Raveendran, A.; Jean, S.; Mervin, J. A Novel Parametrized Fused Division and Square-Root POSIT Arithmetic Architecture. In Proceedings of the 2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID), Bangalore, India, 4–8 January 2020.
18. Bruguera, J.D. Low Latency Floating-Point Division and Square Root Unit. *IEEE Trans. Comput.* **2019**, *69*, 274–287. [\[CrossRef\]](#)