SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V10I10P115

Volume 10 Issue 10, 151-164, October 2023

Original Article

Area and Delay Efficient RNS-Based FIR Filter Design Using Fast Adders and Multipliers

M. Balaji[1,2], N. Padmaja[2]

[1]*Department of ECE, Jawaharlal Nehru Technological University, Ananthapuramu, Andhra Pradesh, India.*
[2]*Department of ECE, Sree Vidyanikethan Engineering College, Andhra Pradesh, India.*

[1]*Corresponding Author : balajichaitra3@gmail.com*

Received: 15 August 2023 Revised: 17 September 2023 Accepted: 14 October 2023 Published: 31 October 2023

Abstract *- Speed and area are the primary design concerns in today's digital age. Increasing the rate at which multiplications and additions are performed has always been necessary for developing cutting-edge technologies. Wallace and Dadda multipliers are among the fastest multipliers used in many processors to accomplish fast arithmetic operations. A novel approach to design a Lookup Table (LUT) multiplier and adder was proposed and implemented in the Finite Impulse Response (FIR) filter. To improve the Residue Number System (RNS) based FIR filter's performance, several adders like Carry Look Ahead (CLA) adder, Kogge Stone Adder (KSA) and proposed adder architectures have also been implemented. Compared with the 16 taps with 32-bit proposed adder with LUT multiplier, the hardware resource utilization (Logic Elements) is decreased by 5.97% and in 32 taps with 16-bit combination, it reduces by 7.60%. Compared with 32 taps with 4-bit word length, the proposed adder with LUT multiplier in the highlighted combinations, the Fmax is increased by 19.28% and in 32 taps with 16-bit, it increases by 29.74%. The Low-pass RNS FIR filter is designed for a cutoff frequency of 50 Hz, generated filter coefficients in MATLAB, and implemented to denoise the ECG signal.*

Keywords *- FIR filter, Dadda multiplier, Lookup Table, Logic Elements, ECG.*

1. Introduction

The optimized multipliers and adders architectures are introduced in the RNS filter to reduce the filter's area and delay. Using optimized adders and multipliers in RNS FIR filters can provide several advantages, such as reduced hardware complexity, improved performance, and lower power consumption. In an FIR filter, the primary operation is multiplying the input samples with filter coefficients and accumulating these products. Using optimized multipliers, such as Wallace multipliers, Dadda multipliers and LUT multiplier, can reduce the number of partial products required and thus reduce the overall hardware complexity of the filter. This, in turn, can result in lower power consumption and reduced circuit area [1].

Similarly, using optimized adders, like Carry-Look Ahead adders, KSA adders, and proposed adders, can improve the filter's performance by reducing the propagation delay of carry signals and minimizing the number of levels of logic gates in the adder circuit. This will result in faster filter operation and reduced power consumption. Overall, using optimized adders and multipliers in FIR filters can significantly improve performance, power consumption, and circuit area, making them a valuable design consideration for efficient digital signal processing applications [2].

There is a critical need for Digital Signal Processing (DSP) expertise due to the decisive nature of its tasks. Multiple adders and multipliers are often seen in sophisticated DSP systems. Better results may be achieved by complicated signal processing techniques with well-designed adders and multipliers.

In many scenarios, such as controllers and processing chips, adders are one of the essential components. Adders may be found in a variety of networks in a variety of blocks. The time needed for a carry to propagate through a digital adder limits its adding speed. In a traditional ripple adder, the total for each bit location is generated sequentially following the addition of the preceding bit position and carry transfer into the next bit position [3].

Multiple Input Multiple Output (MIMO) applications extensively use the various parallel solutions developed for high throughput systems. The parallel architecture improves the system's throughput, but the L-parallel filter design exponentially increases hardware cost and power consumption. With this shortcoming of parallel architecture, different fast multipliers were implemented with the combination of adders [4]. This technique is far more space-efficient than the standard parallel design, cutting hardware

needs in half. Without the multiplier and adder, there wouldn't be much FIR filter. The filter as a whole can only achieve its desired processing speed and power dissipation if its multiplier and adder block perform as expected.

Several effective multiplier and adder architectures have been created [5]. The research found that the Wallace multiplier and Dadda multiplier had excellent results in terms of latency compared to other designs. The implementation results demonstrate the benefits of the suggested design over the standard one, namely more incredible speed and reduced power dissipation [6].

The following sections of the paper are organized: Section 2 offers a comprehensive description of the methodologies utilized in prior designs. Section 3 describes the different adders and multipliers implemented to design the RNS FIR filter, and section 4 represents the results and performance analysis of the filter. Section 6 delivers the conclusion.

2. Literature Review

The Partial Product (PP) reduction phase in multiplication operations is known for its substantial energy consumption and significant silicon area usage. Hence, three primary strategies are typically utilized to design approximate multipliers. The first technique involves approximations when generating Partial Products (PP).

Next, truncation is applied within the PP tree. The last method approximates the adders and compressors accumulating the partial derivatives. Therefore, approximation computing was developed to reduce power consumption. In this study, they use probabilistic pruning, an approximation approach proposed by [7].

An OR-based error-compensated approximation multiplier with input reordered 4:2 compressors was presented by [8] for low-energy design. By switching the information order, the compressor may function with only two of the four inputs, making it more straightforward and requiring fewer gates. The suggested method achieves 99.3% precision, using just 44.7% of the energy and 31.7% of the space of the best existing practices.

Eight-by-eight approximation multipliers based on high-order approximation compressors were proposed by [9]. Accumulating product terms and decreasing energy consumption with few mistakes is achieved using different compressors for different weights. To streamline the 'carry chain's' logic, higher-order approximation compressors are used for the intermediate significance weights, such as 8-to-2 compressors. To create an error-efficient system, [10] proposed the rounding method-based approximation multiplier, which involves rounding up the input operands to the next power of 2. The transformed inputs are run via an arithmetic module that contains subtractor, adder, and shifter units. Input operand sizes might be anything from 8 bits up to 32 bits. The simulation results show that the delay is around 22% and the power use is approximately 57%, both of which are improvements over comparable approximation multipliers.

To resolve this problem, [11] presented a rounding method that may be adjusted on the fly to serve as an approximate multiplier. The suggested multipliers are attractive because they reduce implementation complexity while improving power efficiency. The proposed technique uses 32.5% less energy, has a 50.8% smaller footprint than filters using existing multipliers, and has 54.7% less latency. To create an approximation multiplier, [12] suggested adding an approximate compressor with a single gate. The proposed technology consumes 61% less energy and occupies 52% less space than current methods.

An approximation multiplier for unsigned integers was suggested by [13] due to its great configurability; it aims to reduce all hardware metrics while retaining outstanding accuracy. It provides various options for reducing energy use by 35–85%, so it may be used in multiple contexts without breaking the bank. An approximation multiplier through the truncating approach was presented by [14]. The approximation multiplier worked by computing the final result using scientific-binary representations of the operands and truncating the intermediate results.

Compared to the same multiplier, this one is 89.2 percent more efficient at conserving energy while taking up just 74.9 percent less room on average. This study presents a novel approximation adder to be used with an energy-efficient, high-performance approximate PP accumulation tree for a multiplier.

To avoid 'carry propagation,' the proposed approximation adder generates an error vector and a rough total. The OR gates and approximation adder-based error reduction methodologies [15] provide two distinct designs for approximate 8X8 multipliers, M1 and M2. It has been shown that the power consumption of the suggested approximation multipliers is lower than that of a speed-optimized accurate Wallace multiplier.

Minimal error margins allow the proposed multipliers to achieve high precision. As a bonus, simulations have shown that M2, although having a longer delay and using more power, is more accurate than M1. Multipliers for the proposed approximation are more precise than those used in prior approximation models. Compared to older designs that prioritized delay and energy savings but had inconsistent accuracy, the suggested alternatives provide considerable savings while maintaining a high level of precision.

3. Proposed Work

In this work, the RNS FIR filter is designed to enhance the filter's performance using optimised adders and multipliers. In an FIR filter, the primary operation is multiplying and accumulating the input samples with filter coefficients.

Using optimized multipliers, such as Wallace multipliers, Dadda multipliers, and LUT Multiplier, can reduce the number of partial products required and thus reduce the overall hardware complexity of the filter. This, in turn, can result in lower power consumption and reduced circuit area. Similarly, using adders like Carry Look-Ahead adder, KSA adder and the proposed adder results in high speed by reducing the propagation delay of the carry. These optimized adders and multipliers are introduced into the RNS FIR filter.

3.1. Operation of RNS-Based FIR Filter

The RNS is a method of representing integers using a set of residue classes. In RNS, an integer is characterised by its remainder modulo, a group of pairwise coprime integers. The RNS provides a way to perform modular arithmetic on the residues in parallel, which can be advantageous for certain digital signal processing operations [18].

Modular arithmetic is a crucial technique to convert the input word into a series of residues (x1, x2, x3) within an FIR filter based on the Residue Number System (RNS). The moduli values are consistently represented in the form of 2n-1, 2n, and 2n+1. When n equals 3, the modulo set takes the form of 7, 8, and 9. Applying modular operations on the filter coefficients and input signals results in the generation of residues. Subsequently, the output of the forward converter undergoes a sequence of FIR filters before being transmitted to the RNS decoder, as depicted in Figure 1.

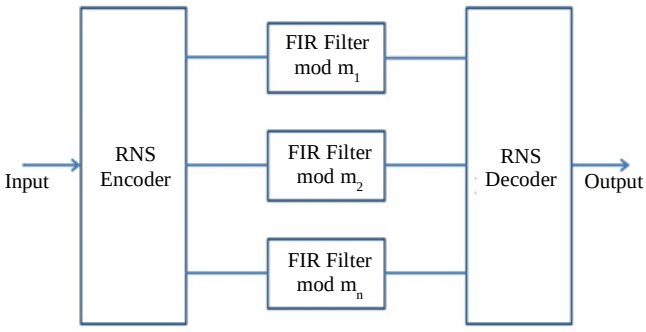

Fig. 1 Block diagram of RNS-based FIR filter

3.2. RNS Encoder Working

The RNS encoder has three modulus operators, each designed for one of the three sets. These operators execute modulus operations on the input sequence, creating a residue set. This residue set is then fed into the FIR filter, and the output of the FIR filters is connected to the RNS decoder.

The logical diagram of the RNS encoder is visually represented in Figure 2.

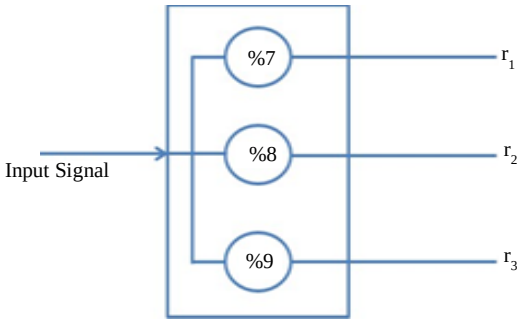

Fig. 2 RNS encoder implementation

For input x[n]=01100100 and moduli set (m_1, m_2, m_3)= {7, 8, 9} the residues are given as

$$r_1 = x[n] \% 7 = (01100100) \% 7 = 0010$$

$$r_2 = x[n] \% 8 = (01100100) \% 8 = 0100$$

$$r_3 = x[n] \% 9 = (01100100) \% 9 = 0001$$

The modulus output of residues is (r_1, r_2, r_3) given to the FIR circuit shown in Figure 3. The FIR circuit can be a basic FIR circuit modified binary DA or a partitioned LUT-based FIR filter. The outputs of appropriate residues are given to the RNS decoder [17].

3.3. FIR Filter Implementation

The coefficients of the FIR filter are d0, d1, d2, and d3. X[n] is the input sequence or a word. The sequence gets delayed for each clock cycle, enters the multiplier, and gets multiplied with the appropriate coefficients. The output of each multiplier is added, resulting in the production. The output for 4-tap FIR filter with coefficients d0, d1, d2 and d3 is given as,

$$Y[n] = d_0 \times X[n] + d_1 \times X[n-1] + d_2 \times X[n-2] + d_3 \times X[n-3] \quad (1)$$

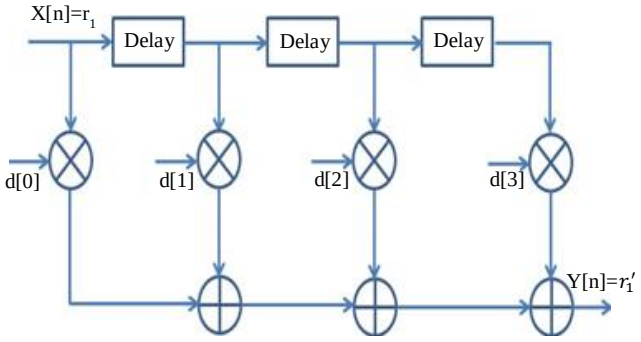

Fig. 3 Structure of 4-tap FIR filter

3.3.1. Operation of FIR Filter

Consider an FIR filter with coefficients as d_0= 0010, d_1= 0100, d_2=0110, d_3=1000 (in binary). Consider the input sequence or input word to be X[n] = 0010 (in binary). The output of the FIR filter after substituting the filter coefficients in equation (1) is,

Y[n] = 0010* X[n] +0100 * X[n-1] + 0110 * X[n-2] + 1000 * x[n-3]

Where X[n-1], X[n-2], X[n-3] are delayed input sequence.

First Clock Cycle

X[n]= 0010, X[n-1] = 0000, X[n-2] = 0000, X[n-3] =0000. Output: Y[n] = 0010*0010 +0000*0100 + 0000*0110 + 0000*1000 = 00100 (4 in decimal).

Second Clock Cycle

X[n]= 0010, X[n-1] = 0010, X[n-2] = 0000, X[n-3] =0000. Output: Y[n] = 0010*0010 +0010*0100 + 0000*0110 + 0000*1000 = 00100 + 01000 = 01100 (12 in decimal).

Third Clock Cycle

X[n]=0010, X[n-1] = 0010, X[n-2] = 0010, X[n-3] =0000. Output: Y[n] = 0010*0010 +0010*0100 + 0010*0110 + 0000*1000 = 00100 + 01000 + 01100 = 11000 (24 in decimal).

Fourth Clock Cycle

X[n]=0010, X[n-1] = 0010, X[n-2] = 0010, X[n-3] = 0010. Output: Y[n] = 0010*0010 + 0010*0100 + 0010*0110 + 0010*1000= 00100 + 001000 + 01100 + 10000 = 101000 (40 in decimal).

3.4. RNS Decoder Working

The Chinese remainder theorem [19] establishes that when the divisors are mutually coprime, it becomes possible to uniquely determine the remainder of the division of n by the product of these integers if one has the remainders resulting from the Euclidean division of an integer by multiple integers. This theorem provides a solution to systems of linear congruences with several moduli, offering an alternative approach to finding a unique solution for a set of equations where the variables are integers, and each equation is based on a different modulo.

Let us consider the output of the FIR filter $(r_1', r_2', \ldots . r_t')$ with modulo $(m_1, m_2, \ldots . m_t)$ where all m_t are mutually prime.

Let M = $(m_1 * m_2 * \ldots . m_t)$

$$M_i = {}^{M}/_{m_i} \quad (2)$$

Let K_i be the result that $(M_i * K_i)\%m_i = 1$, then the corresponding numbers as

$$y = (\sum_{i=1}^{t}(M_i K_i r_i'))\%M \quad (3)$$

RNS output from equation (3) is

$$= \left(\sum (M_i \times K_i \times r_i)\right)\%M$$

=((200*272*8)+(80*255*15)+(300*240*9))%4080
=2000

3.5. Proposed Adder and LUT Multiplier

The proposed half adder circuit in Figure 4 uses basic gates, and the full adder in Figure 5 uses two proposed half-adders with an OR gate. By using these proposed adders, ripple carry adder is designed.

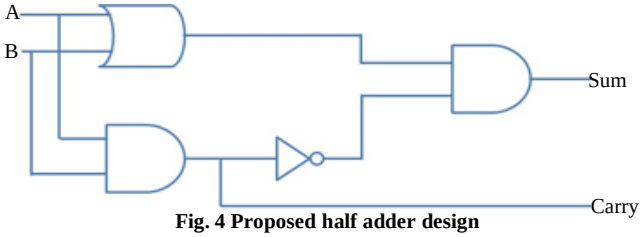

Fig. 4 Proposed half adder design

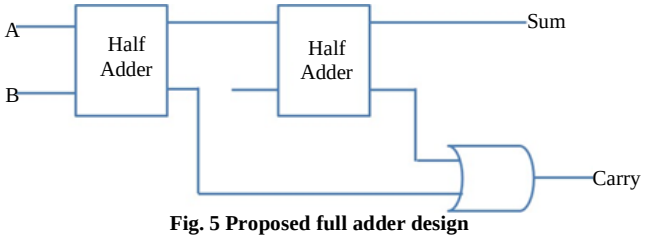

Fig. 5 Proposed full adder design

An LUT (Lookup Table) multiplier is a digital multiplier that uses a table of precomputed values to perform multiplication operations. The LUT multiplier is a popular multiplication method in digital circuits because it can be implemented using simple combinational logic and is area-efficient.

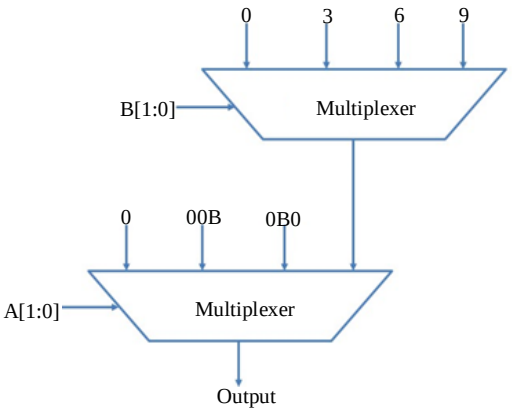

Fig. 6 Proposed 2*2 LUT multiplier design

When A is assigned a value of 11, the resulting output is contingent upon the value of B. The construction of a 4-bit multiplier involves the utilization of four 2-bit multipliers, which rely on Lookup Tables (LUTs). This 4-bit multiplier takes as input two 4-bit values, one for each input operand. These 4-bit input values are partitioned into four sets of 2-bit segments, each undergoing multiplication via 2-bit multipliers based on LUTs. To compute the ultimate output of the 4-bit multiplier, the outcomes of these four distinct multipliers are fed through a combination of full and half adders.

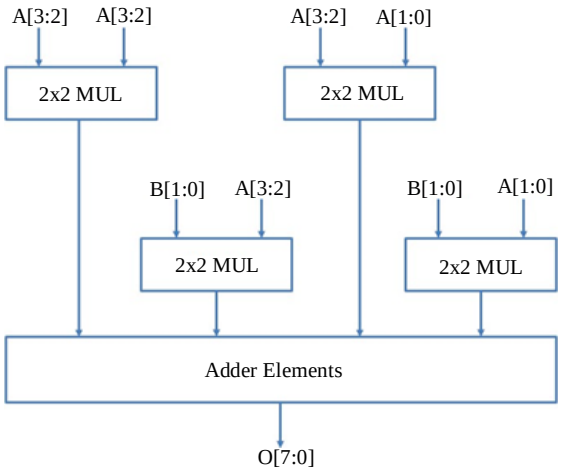

Fig. 7 Proposed 4*4 LUT multiplier design

Finally, when the input value of A is set to 11, the output value depends on the input value of B. A 4-bit multiplier is created by employing four 2-bit multipliers based on Lookup Tables (LUTs). This 4-bit multiplier takes two 4-bit inputs, one for each operand. These 4-bit inputs are divided into four pairs of 2-bit segments, and each pair is multiplied using LUT-based 2-bit multipliers. To obtain the final output of the 4-bit multiplier, the results from these four individual multipliers are passed through full and half adders.

3.6. Wallace Multiplier

A binary multiplier, a digital circuit that multiplies two numbers, is realized in hardware as a Wallace multiplier. It gradually adds up partial products using a variety of full and half adders (the Wallace tree or Wallace reduction) until only two numbers remain. Dadda multipliers aim to limit the number of gates needed by delaying reduction to higher levels, while Wallace multipliers strive to do as much reduction as possible on each layer. There are three levels to the Wallace tree [20].

The bits of one argument are multiplied by the bits of the second argument. Stacks of full and half adders may be used to reduce the number of partial products to two, and then the wires can be grouped into two numbers and added using a standard adder. Long multiplication has a somewhat different form in the Wallace tree. The first thing is multiplying each digit (bit) of the first component by each number (bit) of the second factor. The weight of each of these partial products is the product of its elements. The whole is equal to the total of the parts, weighted according to their relative importance [21].

3.7. Dadda Multiplier

The Dadda tree or Dadda reduction is a set of full and half adders that add up partial products in steps until only two integers remain. Design-wise, it's not too dissimilar from the Wallace multiplier. Still, the new reduction tree makes it quicker and uses fewer gates (except the smallest operand sizes) (for all operand sizes). Dadda multipliers aim to minimize input/output latency and the number of gates needed, unlike Wallace multipliers, which aim to eliminate as much as feasible on each layer. This makes the reduction step of Dadda multipliers cheaper, but the resulting numbers may be a few bits longer, necessitating somewhat larger adders [22].

A lot less time and effort has been put into developing approximate multipliers. A multiplier consists of a PP generator, a PP accumulator, and a carry propagation adder. The approximation PP is computed using faulty 22 multiplier blocks and then accumulated with correct adders in an adder tree. Approximate 44, 88, and 1616 Wallace multipliers [23] may be generated using a carry-in-prediction strategy. The multiplier's last addition step is an excellent place to use the speculative approximation adders. The error-tolerant nature of specific applications informs the design of approximate multipliers. The multiplier is segmented into the multiplying segment (MSB) and the non-multiplying segment by the static-segment multiplier (LSB). These multipliers are designed for usage with unsigned data [24]. Most often, a Booth algorithm is employed to implement signed multiplication. For fixed-width Booth multipliers, approximate methods have been proposed using conditional probability approaches; these methods may be expanded to huge Booth multipliers with widths of more than 32 bits [25].

The technique outperforms conventional approaches in terms of accuracy and space efficiency. By making horizontal and vertical cuts in a carry-save adder, an exact array multiplier may be transformed into an approximation array multiplier. Booth methods in signed 32-bit and 16-bit radix-8 and approximation computing using a 2-bit adder are used to construct an FIR adaptive filter with low PP and minimal accumulation circuitry. Approximation computing has become more important to reduce overhead in embedded and high-performance systems. Data collection is a crucial feature of Wireless Sensor Networks [26-28].

3.8. Carry Look-Ahead Adder

The Carry-Look-Ahead adder's fundamental principle is to produce the carry-in bit for each whole adder circuit based on the carry-out bit of the preceding adder circuit, as opposed

to waiting for the generation of the carry-out bit before using it as the carry-in bit for the following adder circuit. This method makes adding large binary integers much faster. The Carry-Look-Ahead adder speeds up the addition process by concurrently creating the carry bits for each bit position rather than sequentially. The Carry Look-Ahead adder creates the carry-in bit for each complete adder circuit by combining logic gates like AND and OR gates.

The logic gates produce a 'create' signal to determine whether the carry bit needs to be generated for the current adder circuit and a "propagate" signal to determine whether the carry bit is being propagated from the previous adder circuit. The carry-in bit for each adder circuit is then calculated using extra logic gates to propagate and produce signals. Large binary numbers can be added quickly and effectively by repeating this procedure for each bit location in the binary numbers being added [16].

4. Results and Discussion

The High-speed, low-power design of the FIR filter is of utmost importance, which makes to propose implementing the FIR filter using the Wallace multiplier, Dadda multiplier, and LUT multiplier with CLA adder, KSA adder and proposed adders in the suggested work.

The multiplier will produce high-speed, low-area semiconductors by comparing the improved data with other multipliers using simulation and synthesis correspondences in the Quartus tool. In Table 1, the Logic Elements and maximum power dissipation for different multipliers and adders for different combinations of input bit sizes were implemented and compared. In the proposed adder, the logic elements are reduced by 55.86% compared to the KSA adder and 59.49% by the CLA adder. The change in power consumption was nominal when compared with bit by bit individually.

Table 1. Logic Elements (LEs) and power dissipation comparison for different sizes of adders

Adder	Parameters	4 Bit	8 Bit	16 Bit	32 Bit	64 Bit
CLA Adder	LE's	8	18	38	78	158
	Pdmax (mW)	65.18	66.31	68.56	73.07	82.09
KSA Adder	LE's	8	17	36	74	145
	Pdmax (mW)	65.21	66.28	68.51	72.90	82.05
Proposed Adder	LE's	6	8	16	32	64
	Pdmax (mW)	65.09	66.22	68.47	72.98	81.99

Table 2. Logic Elements (LEs) and power dissipation comparison for different sizes of multipliers

Multiplier	Parameters	4×4	8×8	16×16	32×32
Wallace Multiplier	LE's	37	176	758	3126
	Pdmax (mW)	65.47	66.97	70	76.08
Proposed LUT Multiplier	LE's	31	146	623	2567
	Pdmax (mW)	65.47	66.97	69.98	76.06
Dadda Multiplier	LE's	29	144	624	2586
	Pdmax (mW)	65.47	66.97	69.99	76.06

In Table 2, Wallace, Dadda and proposed multipliers were compared with 4×4, 8×8, 16×16, and 32×32 bit sizes. The Logic Elements and maximum power dissipation for a 4×4 Wallace multiplier, Dadda multiplier, and proposed LUT multiplier are 37, 31, and 29, respectively, whereas for 32×32 size, they are 3126, 2586, 2567. The power dissipation for 4×4 size is the same for Wallace Multiplier, Dadda multiplier, and Proposed LUT Multiplier is 65.47mW, and for 32×32 size multipliers, it is 76.06mW. In the proposed work, the RNS FIR filter was designed with CLA adder, KSA adder, and proposed adder with different combinations of multipliers like Wallace, Dadda, and LUT multiplier. The performance analysis parameters, like area (LEs), delay, and Fmax, were compared for different combinations of tapping (4, 8, 16, 32, and 64) with varying lengths of bit (4, 8, 16, and 32).

In Table 3, Logic Elements for different combinations of adders and multipliers were compared to 64 taps with 32-bit word length. RNS FIR filter was designed using a proposed adder with LUT multiplier, and the proposed adder with vedic multiplier gives better performance in terms of the area when compared with other combinations of adders and multipliers. The proposed adder with LUT multiplier with 8-tap saves 6.29% of logic elements for 4-bit RNS FIR filter, 14.05% of Logic Elements for 8-bit, 28.88% of logic elements for 16-bit, and 22.07% of logic elements for 32-bit RNS FIR filter.

Table 3. Logic Elements (LE's) comparison of a different combination of fast adders and fast multipliers

Parameter		CLA Add –Wal Mul	CLA Add– Dadda Mul	CLA Add - LUT Mul	KSA Add-Vedic Mul	Proposed Add-LUT Mul	Proposed Add-Vedic Mul
4 Tap	4 Bit	552	552	546	592	524	554
	8 Bit	785	785	783	834	775	778
	16 Bit	1034	1034	1031	1080	1022	1026
	32 Bit	1525	1525	1522	1572	1517	1521
8 Tap	4 Bit	648	654	657	667	625	640
	8 Bit	1031	1031	999	1138	978	993
	16 Bit	1280	1280	1255	1724	1226	1387
	32 Bit	1768	1769	1735	2211	1723	1870
16 Tap	4 Bit	1008	1016	1065	1219	1059	1032
	8 Bit	2226	2226	2291	2817	2186	2084
	16 Bit	2539	2539	2475	3019	2427	2257
	32 Bit	3111	3111	2972	3560	2925	2756
32 Tap	4 Bit	1616	1647	1741	2006	1785	1644
	8 Bit	4218	4216	4433	5219	4328	3559
	16 Bit	5111	5109	5178	6261	4784	3985
	32 Bit	6470	6470	6218	7082	5482	4883
64 Tap	4 Bit	2835	2894	3076	3751	3232	2948
	8 Bit	8542	8136	5475	11752	5251	7609
	16 Bit	11893	11407	12389	11994	11338	9770
	32 Bit	12644	12563	15728	13890	13830	11460

Table 4. Overall comparison of area (Logic Elements utilization) of proposed results with existing results for 8-tap with an 8-bit combination of input word length

Design/ Parameters	Area (Logic Elements)
G. Reddy Hemantha et al. [3]	4281
Burhan Khurshid et al. [4]	2789
Pavel Lyakhov [30]	1388
D. Kaplun [31]	2456
C.W. Tung [33]	2637
KSA Adder, Vedic Multiplier	**1138**
CLA Adder – Dadda Multiplier	**1031**
CLA Adder –Wallace Multiplier	**1031**
CLA Adder, LUT Multiplier	**999**
Proposed Adder, Vedic Multiplier	**993**
Proposed Adder, LUT Multiplier	**978**

The overall results for the area in terms of Logic Elements utilization produced by the proposed works when compared with the existing works are shown in Table 4.

When compared with the proposed adder with LUT multiplier with Reference [3], the area utilization is decreased by 77.15% and when compared with the proposed

methods, the area utilization is saved by 14.05% in proposed adder with LUT multiplier when compared with the KSA adder with Vedic multiplier combination.

In Table 5, the critical path delay for different combinations of adders and multipliers was compared up to 64-tap with 32-bit word length. The RNS FIR filter using CLA adder with LUT multiplier takes less critical path delay. The CLA adder with LUT multiplier with 8-tap saves 9.92% of logic elements for 4-bit RNS FIR filter, 5.25% of Logic Elements for 8-bit, 8.99% of Logic Elements for 16-bit, and 8.11% of Logic Elements for 32-bit RNS FIR filter.

Table 5. Delay comparison of a different combination of fast adders and fast multipliers

Parameter		CLA Add –Wal Mul	CLA Add – Dadda Mul	CLA Add - LUT Mul	KSA Add-Vedic Mul	Proposed Add -LUT Mul	Proposed Add-Vedic Mul
4 Tap	4 Bit	0.543	0.543	0.501	0.511	0.545	0.507
	8 Bit	0.541	0.541	0.492	0.499	0.491	0.549
	16 Bit	0.514	0.501	0.483	0.551	0.488	0.555
	32 Bit	0.546	0.544	0.491	0.536	0.495	0.517
8 Tap	4 Bit	0.544	0.513	0.490	0.506	0.509	0.514
	8 Bit	**0.514**	**0.544**	**0.487**	**0.488**	**0.491**	**0.498**
	16 Bit	0.556	0.549	0.506	0.509	0.505	0.506
	32 Bit	0.542	0.465	0.498	0.475	0.499	0.495
16 Tap	4 Bit	0.504	0.500	0.502	0.501	0.493	0.498
	8 Bit	0.496	0.500	0.501	0.498	0.492	0.499
	16 Bit	0.500	0.500	0.505	0.499	0.501	0.499
	32 Bit	0.500	0.500	0.504	0.503	0.494	0.500
32 Tap	4 Bit	0.153	0.302	0.488	0.136	0.327	0.183
	8 Bit	0.163	0.149	0.127	0.131	0.197	0.118
	16 Bit	0.186	0.114	0.305	0.344	0.123	0.410
	32 Bit	0.136	0.165	0.202	0.226	0.127	0.113
64 Tap	4 Bit	0.147	0.131	0.148	0.124	0.275	0.159
	8 Bit	0.276	0.272	0.354	0.181	0.188	0.174
	16 Bit	0.183	0.130	0.119	0.299	0.184	0.243
	32 Bit	0.191	0.231	0.115	0.456	0.347	0.399

Table 6. Overall comparison of performance analysis of proposed results with existing results for 8-tap with an 8-bit combination of input word length

Design/ Parameters	Critical Path Delay (ns)
Patronik et. al [7]	7.29
Patronik et. al [7]	7.28
Patronik et. al [7]	7.07
Shaheen Khan et.al [2]	6.00
M Balaji et. al [22]	5.39
R Kamal et. al [9]	5.23
Patronik et. al [7]	4.67
T. K. Shahana et. al [15]	2.55
CLA Adder – Dadda Multiplier	**0.544**
CLA Adder –Wallace Multiplier	**0.514**
Proposed Adder- Vedic Multiplier	**0.498**
Proposed Adder, LUT Multiplier	**0.491**
KSA Adder – Vedic Multiplier	**0.488**
CLA Adder- LUT Multiplier	**0.487**

The overall results for the critical path delay produced by the proposed works compared to the existing assignments for 8-tap with 8-bit combinations are shown in Table 6. Compared with the memory-less DA I, CLA Adder with LUT Multiplier outfits the critical path delay by 93.83% and when compared with T. K. Shahana [15], it decreases by 80.90%.

In Table 7, the Fmax for different combinations of adders and multipliers was compared up to 64-tap with 32-bit word length. When comparing the different combinations for 16-tap, the CLA adder with LUT multiplier will be the appropriate selection to produce the maximum frequency for different bit sizes. The Fmax was increased by 11.77% for the 4-tap, and the 32-tap increased by 11.59%.

Table 7. Fmax comparison of a different combination of fast adders and fast multipliers

Parameter		CLA Add – Wal Mul	CLA Add – Dadda Mul	CLA Add-LUT Mul	KSA Add-Vedic Mul	Proposed Add-LUT Mul	Proposed Add-Vedic Mul
4 Tap	4 Bit	1355.01	1355.01	1253.13	1138.95	1377.41	1107.42
	8 Bit	1367.99	1364.26	1231.53	1089.32	1242.24	1219.51
	16 Bit	1254.71	1223.99	1215.07	1221.00	1231.53	1231.53
	32 Bit	1375.52	1367.99	1237.62	1196.17	1239.16	1144.16
8 Tap	4 Bit	1364.26	1254.71	1201.92	1117.32	1250.00	1119.82
	8 Bit	1254.71	1400.56	1219.51	1085.78	1216.55	1095.29
	16 Bit	1390.82	1390.82	1237.62	592.07	1228.50	1103.75
	32 Bit	1355.01	1162.79	1245.33	878.73	1218.03	1095.29
16 Tap	4 Bit	1234.57	1226.99	1236.09	1095.29	1207.73	1090.51
	8 Bit	1226.99	1228.50	1226.99	1092.90	1207.73	1092.90
	16 Bit	1225.49	1231.53	1239.16	1091.70	1225.49	1094.09
	32 Bit	1225.49	1223.99	1237.62	1106.19	1215.07	1094.09
32 Tap	4 Bit	780.64	935.45	1225.49	686.81	967.12	553.71
	8 Bit	789.27	778.82	688.71	605.33	821.69	651.04
	16 Bit	713.27	659.63	938.97	880.28	654.88	934.58
	32 Bit	761.61	573.39	819.00	485.20	756.43	667.11
64 Tap	4 Bit	630.52	660.94	584.80	584.45	527.43	513.08
	8 Bit	526.59	531.07	500.50	510.20	607.53	462.32
	16 Bit	572.41	593.12	586.85	524.65	558.35	472.14
	32 Bit	540.12	545.55	505.56	547.25	515.76	547.60

Table 8. Overall comparison of Maximum Frequency (Fmax) for 8-tap with 8-bit word length capability

Design/ Parameters	Fmax (MHz)
D. Kaplun[31]	180.00
Pavel Lyakhov[30]	285.00
Burhan Khurshid - Transposed form[4]	525.88
H.M. Kamboh [29]	535.00
Burhan Khurshid –Direct form [4]	568.04
Proposed Adder, LUT Multiplier	**1216.55**
CLA Adder, LUT Multiplier	**1219.51**
CLA Adder –Wallace Multiplier	**1254.71**
KSA Adder, Vedic Multiplier	**1085.78**
Proposed Adder, Vedic Multiplier	**1095.29**
CLA Adder – Dadda Multiplier	**1400.56**

The overall comparison of the maximum frequency produced by the proposed works when compared with the existing works is shown in Table 8. The CLA adder with the Dadda multiplier gives better results than the Fmax, which is increased by a maximum of 87.14% with the existing work [31]. Compared with the proposed works, it increases by a maximum of 13.14% for 8-tap with 8-bit input word length combination.

5. Practical Implementation of Denoising the ECG Signal Using Designed RNS FIR Filter

Various sources, including external interference from power lines or electrical equipment, poor electrode contact or skin preparation, and physiological sources, such as muscle activity or tremors, can cause high-frequency noise in ECG signals. One of the most common techniques for removing high-frequency noise from ECG signals is low-pass filtering.

A low-pass filter facilitates the passage of low-frequency signals while suppressing high-frequency ones. Choosing the cutoff frequency is crucial when configuring the low-pass filter for ECG signal processing [32]. This frequency should be set sufficiently high to preserve the integrity of the ECG signal and, at the same time, low enough to effectively eliminate high-frequency noise from the signal. The design process for a digital low-pass FIR filter using the Kaiser window method involves specifying the filter specifications, choosing the filter length and beta value, calculating the normalized cutoff frequency and ideal frequency response, computing the Kaiser window coefficients, multiplying the desired filter coefficients with the Kaiser window coefficients, normalizing the filter coefficients, and implementing the filter.

The binary sequence is derived from ECG signal data, utilizing samples from the renowned MIT-BIH Arrhythmia database (MIT-BIH-AR). This ECG database is collected from PhysioNet, and the signal is plotted using the MATLAB tool.

Figure 8 shows that most ECG signals contain noise and interference in the higher frequency range, such as power line interference (50 Hz or 60 Hz) and muscle artefacts. Applying a low-pass filter with a cutoff frequency of 50 Hz can attenuate these unwanted components, improving the overall signal quality. Random noise with a frequency of more than 50 Hz, shown in Figure 9, is generated in MATLAB and added that generated noise to the ECG Signal, as shown in Figure 10.

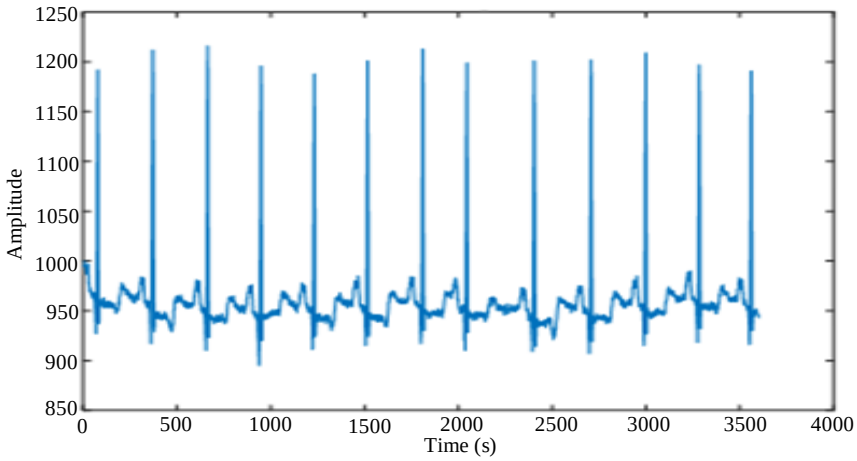

Fig. 8 The ECG signal

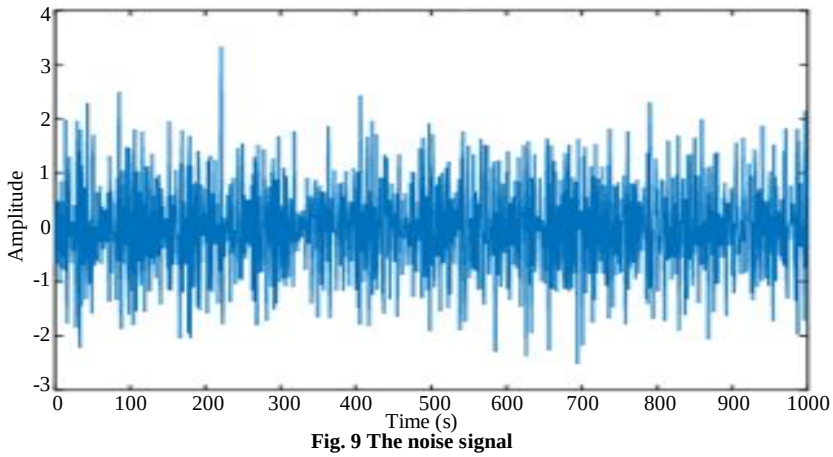

Fig. 9 The noise signal

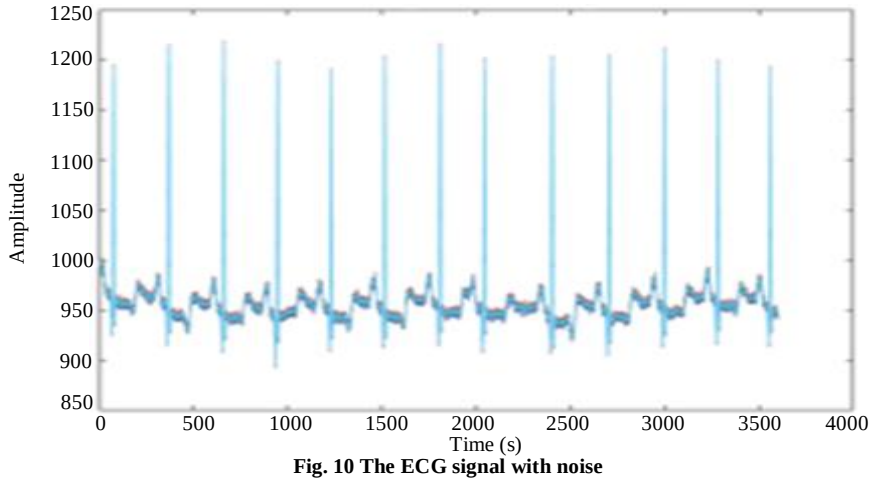

Fig. 10 The ECG signal with noise

The generated coefficients from ECG signal added with noise are given as inputs to low pass RNS digital FIR filter for 4-Tap, 8-Tap, 16-Tap, 32-Tap and 64-Tap. As the number of taps in a filter increases, it improves noise elimination by enhancing frequency selectivity, improving stop-band attenuation. The generated coefficient for ECG signal, noise signal and ECG signal added with noise is shown in Figure 11.

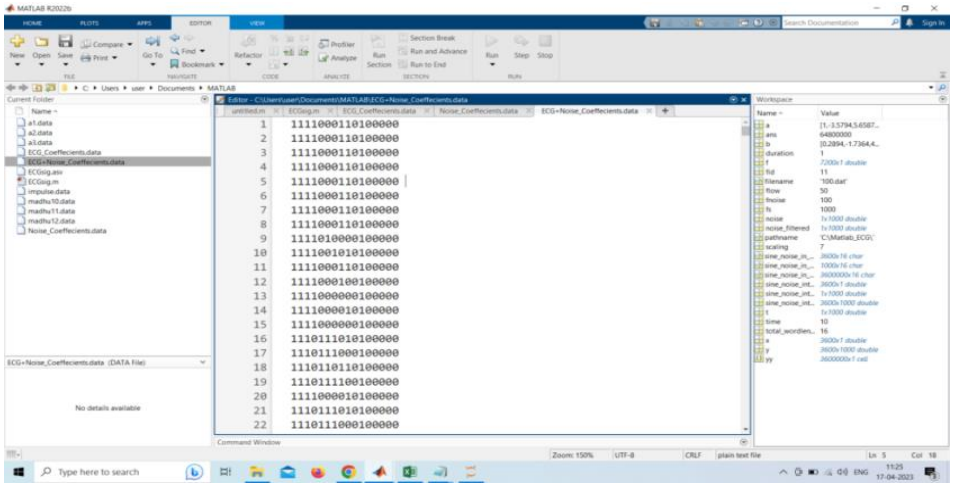

Fig. 11 The MATLAB generated coefficients of ECG signal with noise

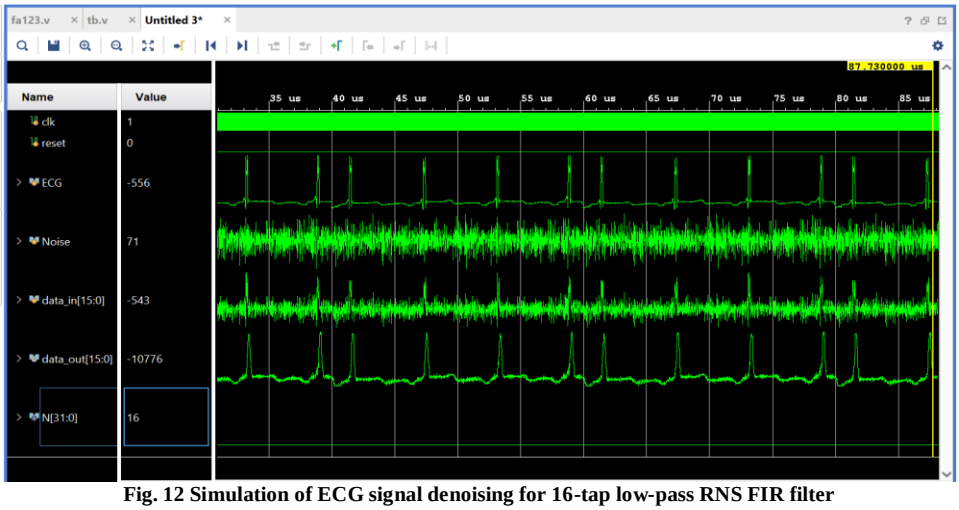

Fig. 12 Simulation of ECG signal denoising for 16-tap low-pass RNS FIR filter

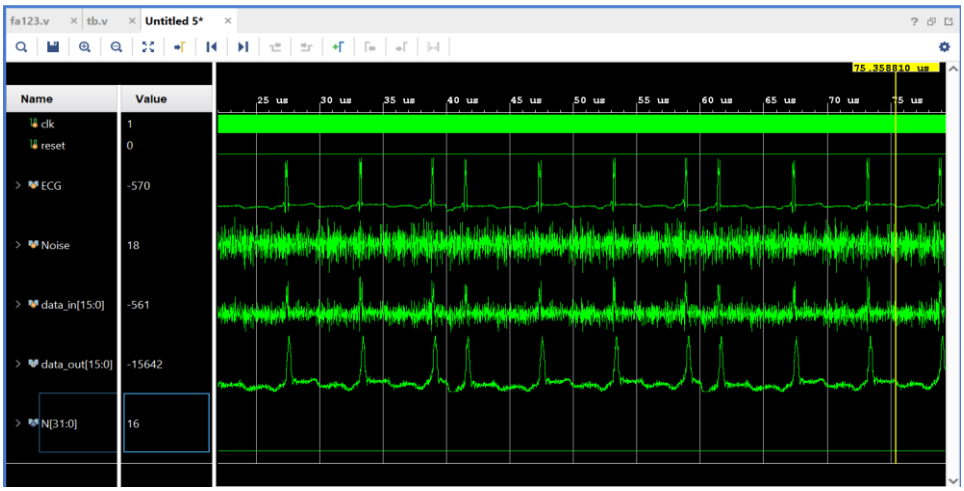

Fig. 13 Simulation of ECG signal denoising for 32-tap low-pass RNS FIR filter

Figure 12 and 13 represent the simulated waveform of a 16-tap and 32-tap low pass RNS FIR filter along with ECG signal, noise signal, and noised ECG signal. The filtered signal is obtained by processing the input through a filter with 32 coefficients-the 32-tap RNS FIR filter results from more nose elimination compared to 4-tap, 8-tap and 16-tap. With more coefficients, it can more precisely shape the frequency response, allowing for a narrower transition band and improved attenuation of unwanted high-frequency components.

6. Conclusion

The RNS system supports the rapid development of FIR filters of varying moduli sets and forward and reverse converters. Various design considerations are considered for the RNS-based FIR filter, including Logic Elements usage, delay, power consumption, maximum operating frequency, filter order, and the number of taps. The design modules were implemented using Verilog HDL, and their functionality was validated through ModelSim. The FPGA QUARTUS II 9.0 version was employed to establish the RNS design specifications and develop the FIR filter designs to meet the requirements of ALTERA CYCLONE III logic family devices in the 65nm technology. The proposed adder with LUT multiplier with 8-tap saves 6.29% of the hardware resource utilization (Logic Elements) for 4-bit RNS FIR Filter, 14.05% of Logic Elements for 8-bit, 28.88% of logic elements for 16-bit, and 22.07% of Logic Elements for 32-bit RNS FIR filter combinations.

The CLA adder with LUT multiplier with 8-tap saves 9.92% for 4-bit RNS FIR Filter, 5.25% for 8-bit, 8.99% for 16-bit, and 8.11% for 32-bit RNS FIR filter combinations in terms of critical path delay. The proposed adder with Vedic multiplier with 8-tap saves 17.91% for 4-bit RNS FIR filter, 12.70% for 8-bit, 20.64% for 16-bit, and 19.16% for 32-bit RNS FIR filter in terms of maximum frequency.

The Low-pass RNS FIR filter is designed for a cutoff frequency of 50 Hz and generated filter coefficients in MATLAB and implemented to denoise the ECG signal. Efforts were undertaken to combine the abovementioned processes to construct a high-performance RNS FIR filter design, which was subsequently applied to ECG denoising. The integration of all modules produced encouraging outcomes within the filter design process.

References

[1] Grande Naga Jyothi, Kishore Sanapala, and A. Vijayalakshmi, "ASIC Implementation of Distributed Arithmetic Based FIR Filter Using RNS for High-Speed DSP Systems," *International Journal of Speech Technology*, vol. 23, no. 2, pp. 259-264, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[2] Shaheen Khan, and Zainul Abdin Jaffery, "Modified High-Speed FIR Filter Using DA-RNS Architecture," *International Journal of Advanced Science and Technology*, vol. 29, no. 4, pp. 554-570, 2020. [Google Scholar] [Publisher Link]

[3] G. Reddy Hemantha, S. Varadarajan, and M.N. Giri Prasad, "FPGA Implementation of Speculative Prefix Accumulation-Driven RNS for High-Performance FIR Filter," *Innovations in Electronics and Communication Engineering*, pp. 365-375, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[4] Burhan Khurshid, and Roohie Naaz Mir, "An Efficient FIR Filter Structure Based on Technology-Optimized Multiply-Adder Unit Targeting LUT-Based FPGAs," *Circuits System and Signal Processing*, vol. 36, pp. 600-639, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[5] E. Chitra, T. Vigneswaran, and S. Malarvizhi, "Analysis and Implementation of High Performance Reconfigurable Finite Impulse Response Filter Using Distributed Arithmetic," *Wireless Personal Communications*, vol. 102, no. 4, pp. 3413-3425, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[6] Lavanya Maddisetti, Ranjan K. Senapati, and J.V.R. Ravindra, *Image Multiplication with a Power-Efficient Approximate Multiplier Using A 4:2 Compressor*, Advances in Image and Data Processing Using VLSI Design, Smart Vision Systems, 13th ed., IOP Publishing Ltd, pp. 13-15, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] Piotr Patronik, and Stanisław J. Piestrak, "Hardware/Software Approach to Designing Low-Power RNS-Enhanced Arithmetic Units," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 5, pp. 1031-1039, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[8] Yufeng Xu, Yi Guo, and Shinji Kimura, "Approximate Multiplier Using Reordered 4–2 Compressor with OR-Based Error Compensation," *2019 IEEE 13th International Conference on ASIC (ASICON)*, Chongqing, China, pp. 1-4, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[9] Raj Kamal et al., "Efficient VLSI Architecture for FIR Filter Using DA-RNS," *2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE)*, Hosur, India, pp. 184- 187, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[10] E. Jagadeeswara Rao, and P. Samundiswary, "Error-Efficient Approximate Multiplier Design Using Rounding Based Approach for Image Smoothing Application," *Journal of Electronic Testing*, vol. 37, pp. 623-631, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[11] Bharat Garg, and Sujit Patel, "Reconfigurable Rounding Based Approximate Multiplier for Energy Efficient Multimedia Applications," *Wireless Personal Communications*, vol. 118, pp. 919-931, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[12] Seyed Amir Hossein Ejtahed, and Somayeh Timarch, "Efficient Approximate Multiplier Based on a New 1-Gate Approximate Compressor," *Circuits Systems and Signal Processing*, vol. 41, pp. 2699-2718, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Mostafa Abbasmollaei et al., "A Power Constrained Approximate Multiplier with a High Level of Configurability," *Microprocessors and Microsystems*, vol. 90, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[14] Shaghayegh Vahdat et al., "LETAM: A Low Energy Truncation-Based Approximate Multiplier," *Computers & Electrical Engineering*, vol. 63, pp. 1-17, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[15] T.K. Shahana et al., "Performance Analysis of FIR Digital Filter Design: RNS Versus Traditional," *2007 International Symposium on Communications and Information Technologies*, Sydney, NSW, Australia, pp. 1-5, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[16] Jia Miao, and Shuguo Li, "A Novel Implementation of 4-Bit Carry Look-Ahead Adder," *2017 International Conference on Electron Devices and Solid-State Circuits (EDSSC)*, Hsinchu, Taiwan, pp. 1-2, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[17] Srinivasan Narayanamoorthy et al., "Energy-Efficient Approximate Multiplication for Digital Signal Processing and Classification Applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 6, pp. 1180-1184, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[18] Nikolay N. Kucherov et al., "A High-Speed Residue-to-Binary Converter Based on Approximate Chinese Remainder Theorem," *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Moscow and St. Petersburg, Russia, pp. 325-328, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[19] H. Toyoshima, K. Satoh, and K. Ariyama, "High-Speed Hardware Algorithms for Chinese Remainder Theorem," *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96*, Atlanta, GA, USA, vol. 2, pp. 265-268, 1996. [CrossRef] [Google Scholar] [Publisher Link]

[20] S. Chinnapparaj, and D. Somasundareswari, "Incorporation of Reduced Full Adder and Half Adder into Wallace Multiplier and Improved Carry-Save Adder for Digital FIR Filter," *Circuits and Systems*, vol. 7, no. 9, pp. 2467-2475, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[21] G. Anusha, and P. Deepa, "Design of Approximate Adders and Multipliers for Error Tolerant Image Processing," *Microprocessors and Microsystems*, vol. 72, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[22] M. Balaji, and N. Padmaja, "High-Speed DSP Pipelining and Retiming techniques for Distributed-Arithmetic RNS-based FIR Filter Design," *WSEAS Transactions on Systems and Control*, vol. 17, pp. 549-556, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[23] S. Madhavi et al., "Implementation of Programmable FIR Filter Using Dadda Multiplier and Parallel Prefix Adder," *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, India, pp. 585-589, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[24] H.R. Mahdiani et al., "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850-862, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[25] K. Vijetha, and B. Rajendra Naik, "High Performance Area Efficient DA Based FIR Filter for Concurrent Decision Feedback Equalizer," *International Journal of Speech Technology*, vol. 23, no. 2, pp. 297-303, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[26] A. Uma, P. Kalpana, and T. Naveen Kumar, "Design of DA-Based FIR Filter Architectures Using LUT Reduction Techniques," *Proceedings of the International Conference on Microelectronics, Computing & Communication Systems*, Springer, Singapore, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[27] Grande Naga Jyothi, and Sriadibhatla Sridevi, "Low Power, Low Area Adaptive Finite Impulse Response Filter Based on Memory Less Distributed Arithmetic," *Journal of Computational and Theoretical Nanoscience*, vol. 15, no. 6-7, pp. 2003-2008, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[28] N. Sathya, "An Area Efficient Denoising Architecture Using Adaptive Rank Order Filter," *International Journal of Recent Engineering Science*, vol. 1, no. 4, pp. 11-14, 2014. [Publisher Link]

[29] Hamid M. Kamboh, and Shoab A. Khan, "An Algorithmic Transformation for FPGA Implementation of High Throughput Filters," *2011 7th International Conference on Emerging Technologies*, Islamabad, Pakistan, pp. 1-6, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[30] Pavel Lyakhov et al., "High-Performance Digital Filtering on Truncated Multiply-Accumulate Units in the Residue Number System," *IEEE Access*, vol. 8, pp. 209181-209190, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[31] Dmitry Kaplun et al., "Optimization of the FIR Filter Structure in Finite Residue Field Algebra," *Electronics*, vol. 7, no. 12, pp. 1-14, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[32] Veerabomma Supraja, Pasumarthy Nageswara Rao, and Mahendra Nanjappa Giri Prasad, "Supervised Learning-Based Noise Detection to Improve the Performance of Filter-Based ECG Signal Denoising," *SSRG International Journal of Electronics and Communication Engineering*, vol. 10, no. 6, pp. 35-51, 2023. [CrossRef] [Publisher Link]

[33] Che-Wei Tung, and Shih-Hsu Huang, "A High-Performance Multiply-Accumulate Unit by Integrating Additions and Accumulations into Partial Product Reduction Process," *IEEE Access*, vol. 8, pp. 87367-87377, 2020. [CrossRef] [Google Scholar] [Publisher Link]