Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2018) 26: 2194 – 2205

doi:10.3906/elk-1707-101

A comparative analysis of 1-level multiplier-free discrete wavelet transform implementations on FPGAs

Husam ALZAQ*, Burak Berk ÜSTÜNDAĞ
Department of Computer Engineering, Faculty of Computer Engineering, İstanbul Technical University, İstanbul, Turkey

Received: 08.07.2017 • **Accepted/Published Online:** 26.06.2018 • **Final Version:** 28.09.2018

Abstract: In this article, we investigated the design and implementation aspects of multilevel discrete wavelet transform (DWT) by employing a finite impulse response filter on field programmable gate array platform. We presented two key multiplication-free architectures, namely, the distributed arithmetic algorithm (DAA) and residue number system (RNS). Our goal is to estimate the performance requirements and hardware resources for each approach, allowing for selection of the proper algorithm and implementation of multilevel DAA- and RNS-based DWT. The design has been implemented and synthesized in Xilinx Virtex 6 ML605, taking advantage of Virtex 6's embedded block RAMs. The results reveal that the DAA-based approach is appropriate for a small number of filter taps, while the RNS-based approach would be more appropriate for more than 10 filter taps, yet both DAA- and RNS-based approaches offer high signal quality with peak signal-to-noise ratio as 73.5 and 56.5 dB, respectively.

Key words: Discrete wavelet transform, distributed arithmetic algorithm, field programmable gate array, residue number system

1. Introduction

Discrete wavelet transform (DWT) [1–3], a linear signal processing technique that transforms a signal from the time domain to the wavelet domain [4], employs different algorithms for decomposing a signal into an orthonormal time series with different frequency bands. The signal analysis can be performed using either the pyramid algorithm (PA) [4] or recursive pyramid transform (RPT) [5]. The PA algorithm is based on convolutions with quadrature mirror filters, which is not feasible for practical implementation. However, RPT decomposes the signal $x[n]$ into two parts using high- and low-pass filters, which can be implemented using filter banks [6].

1-D DWT has been implemented for signal denoising, feature extraction, and pattern recognition and compression in [2, 3, 7, 8]. The conventional convolution-based DWT requires massive computations and consumes much area and power, which could be overcome by using the lifting-based scheme for the DWT that was introduced by Sweldens [9]. Although, the lifting scheme is used to compute the output of low- and high-pass filters using fewer components, it may not be well suited for our application due to the nature of the pattern-based cognitive communication system, where the low frequencies components are more important than those of the higher ones [2]. Another advantage of using convolution-based DWT over lifting approach is

*Correspondence: alzaq@itu.edu.tr

that it does not require temporary registers to store the intermediate results and with the appropriate design strategy, it could have better area and power efficiency [10, 11].

Rather than implementing finite impulse response (FIR) filter via multipliers and an adder tree, a multiplier-free architecture is suggested due to their resultant low-complexity systems and their high throughput processing capability [12–14]. There are essentially two approaches for facilitating parallel processing: the distributed arithmetic algorithm (DAA) and residue number system (RNS).

DAA efficiently performs the inner product function in a bit-serial manner via a look-up table (LUT) scheme that is followed by shift accumulation operations of the LUT output [12]. The LUT, a memory element, stores precomputed partial results [13]. Alternatively, RNS is a highly parallel nonweighted arithmetic system that is based on the residue of division operation of integers using the LUT scheme [14]. Eventually, the RNS-based results are converted back to the equivalent binary number format using a Chinese remainder theorem (CRT) [15]. The key advantage of RNS is gained by reducing an arithmetic operation to a set of concurrent but simple operations. Several applications such as digital filters benefit from the RNS design, as in [16].

1.1. Contribution of this paper

In this paper, three major 1-D DWT approaches are implemented on field programmable gate array (FPGA)-based platforms and compared in terms of performance and energy requirements. The implementations are compared for different numbers of multipliers, memory consumptions, number of taps (N), and levels (L) of the transform to show their advantages. To the best of our knowledge, no detailed comparisons of hardware implementations of the three major 1-D DWT designs exist in the literature. This comparison will give significant insight on which implementation is the most suitable for given values of relevant algorithmic parameters. Although there are many efficient designs in the literature, we did not optimize the number of memories in any approach so that we have a fair comparison.

The paper unfolds as follows. In Section 2, the theoretical backgrounds of DAA and RNS are reviewed. In Section 3, the implementation of discrete wavelet transform is described. We further provide an analytical comparison between these approaches. In Section 4, the performance results are presented. Finally, the paper is concluded in Section 5.

2. Background

2.1. Discrete wavelet transform

The wavelet decomposition mainly depends on the orthonormal filter banks. Figure 1 shows a two-channel wavelet structure for decomposition, where $x[n]$ is the input signal, $g[n]$ is the high-pass filter, $h[n]$ is the low-pass filter, and $\downarrow 2$ is the down-sampling by a factor of two. In this way, each filter creates a series of coefficients that represent and compact the original signal information.

Mathematically, a signal, $y[n]$, consists of high- and low-frequency components as shown in Eq. (1). It shows that the obtained signal can be represented by using half of the coefficients because they are decimated by Eq. 2.

$$y[n] = y_{high}[n-1] + y_{low}[n-1]. \tag{1}$$

The decimated low-pass-filtered output is recursively passed through identical filter banks to add the dimension of varying resolution at every stage. Eq. (2) mathematically expresses the filtering process of a signal through a digital low-pass filter $h[k]$. This operation corresponds to a convolution with an impulse response of k-tap

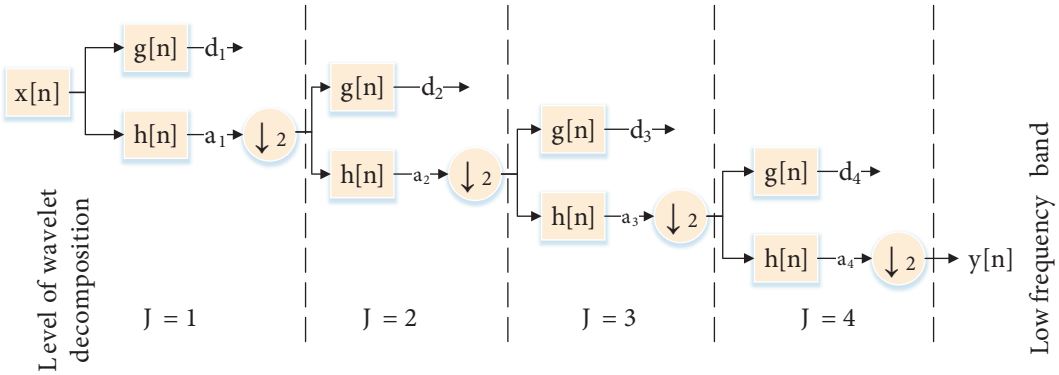

Figure 1. Multilevel wavelet decomposition.

filters.

$$y_{low}[n] = \sum_{k} h[k].x[2n-k], \tag{2}$$

where n becomes $2*n$, representing the down-sampling process. The output $y_{low}[n]$ provides an approximation signal, while $y_{high}[n]$ provides the detailed signal.

2.2. Distributed arithmetic algorithm (DAA)

Eq. (2) shows that output y is the sum of the multiplication of the filter coefficients and the input. DAA eliminates the need for hardware multipliers by performing the arithmetic operations in a bit serial computation[13]. Because the down-sampling process follows each filter (as shown in Figure 1), Eq. (2) can be rewritten without the decimation factor as:

$$y_{low}[n] = \sum_{k=0}^{N-1} x[k].h[k]. \tag{3}$$

Particularly, Eq. (3) is a computational intensive operation owing to multiplication of the real input values with the filter coefficients. Further simplification can be performed on the $x[k]$ in Eq. (3). Considering the representation of $x[k]$ as a fixed-point arithmetic with length $L, x[k]$ becomes:

$$x[k] = -x[k]_0 + \sum_{l=1}^{L-1} x[k]_l.2^{-l}, \tag{4}$$

where $x[k]_l$ is the l^{th} bit of $x[k]$ and $x[k]_0$ is the sign bit. Substituting Eq. (4) into Eq. (3), the output of the filter becomes:

$$y[n] = \left[\sum_{l=1}^{L-1} 2^{-l}.\sum_{k=0}^{N-1} h[k].x[k]_l\right] + \sum_{k=0}^{N-1} h[k](-x[k]_0). \tag{5}$$

Since $x[k]_l$ takes the value of either 0 or 1, $\sum_{k=0}^{N-1} h[k].x[k]_l$ may have only 2^N possible values. That is, rather than computing the summation at each iteration online, it is precomputed and stored in a ROM, indexed by $x[k]_l$. In other words, Eq. (5) efficiently realizes the sum of product computation by memory (LUT), adders, and shift register.

2.3. Residue number system (RNS)

The RNS [15] is a nonweighted number system that performs parallel carry-free addition and multiplication arithmetic. In digital signal processing (DSP) applications, which require intensive computations, the carry-free propagation allows for concurrent computation in each residue channel. The RNS moduli set, $P = m_1, m_2, \ldots, m_q$, consists of q channels. Each m_i represents a positive relatively prime integer, that is $\mathrm{GCD}(m_i, m_j) = 1$, for $i \neq j$.[1] Any number, $X \in \mathbb{Z}_M = 0, 1, \ldots M-1$, is uniquely represented in the RNS by its residues $|X|_{m_i}$, which is the remainder of division X by m_i and M is defined in Eq. (6):

$$M = \Pi_{i=1}^{q} m_i = m_1 * m_2 * \cdots * m_q, \tag{6}$$

where M determines the range of unsigned numbers in $[0, M-1]$ and should be greater than the largest performed results. The implementation of RNS-based DWT is obtained from Eq. (3) as follows:

$$y[n]_{m_i} = y_{m_i} = |(\sum_{k=0}^{N-1} |h[k]_{m_i}.x[n-k]_{m_i}|_{m_i})|_{m_i}, \tag{7}$$

where for each $m_i \in P$. This implies that a $q-$channel DWT is implemented by q FIR filters that work in parallel.

Designing a robust RNS-based DWT requires choosing the moduli set and hardware design of residue-to-binary conversion. Most widely studied moduli sets are given as a power of two due to the attractive arithmetic properties of these modulo sets. For example, $\{2^n - 1, 2^n, 2^{n+1} - 1\}$ [17] and $\{2^n, 2^{2n} - 1, 2^{2n} + 1\}$ [18] have been investigated. In this work, the moduli set $P_n = \{2^n - 1, 2^n, 2^{n+1} - 1\}$ is used for three reasons. First, the multiplicative adder (MA) is simple and identical for $m_1 = 2^n - 1$ and $m_3 = 2^{n+1} - 1$. Second, for small $n = 7$, the dynamic range of P_7 is large, $M = 4145280$, which could efficiently express real numbers in the range $[-2.5, 2.5]$ using 16-bit fixed-point representation, provided scaling and rounding are done properly. Third, the reverse converter unit is simple and regular [19] because it does not employ any ROM.

3. DWT implementation methodology

3.1. DWT implementation using DAA

DAA hides the explicit multiplications with a ROM look-up table. The memory stores all possible values of the inner product of a fixed w-bit with any possible combination of the DWT filter coefficients. The input data, $x[n]$, are signed fixed points of 22-bit width, with 16 binary-point bits ($Q_{5,16}$). We assumed that the memory contents have the same precision as the input, which is reasonable to give high enough accuracy for the fixed-point implementation. As a consequence, 22 ROMs, each consisting of 16 words, are required. Each ROM stores any possible combination of the four DWT filter coefficients, where the final result is a 22-bit signed fixed point ($Q_{5,16}$). In order to decrease the number of the memories, the width should be reduced, which will have impact on the output precision.

Figure 2 shows the block diagram of one-bit DAA at position l. This block contains one ROM (4×22) and one shift register. Because the word length, w, of the input x is 22 bits, the actual design contains 22 blocks. In addition, 21 adders are required to sum up the partial results.

[1] The greatest common divisor (GCD) of two nonzero integers is the largest positive integer that divides them without a remainder.

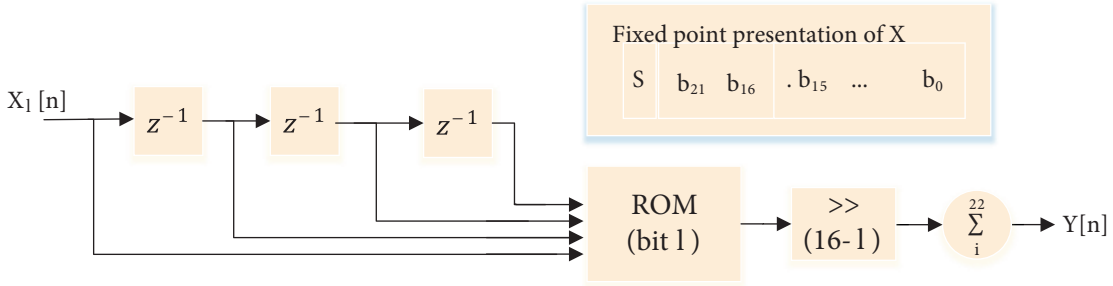

Figure 2. The block diagram of the DAA-based architecture of the DB2. For simplicity, we showed one ROM and one shift register.

3.2. DWT implementation using RNS

The DWT implementation that employs RNS has mainly three components which are the forward converter, the modulo adders, and reverse converter. The forward converter, which is also known as the binary-to-residue converter (BRC), is used to convert a binary input number to residue numbers. In contrast, the reverse converter or the residue-to-binary converter (RBC) is used to obtain the result in a binary format from the residue numbers. We will refer to the RNS system, which does not include RBC, as a forward converter and modular adders (FCMA), as illustrated in Figure 3a.

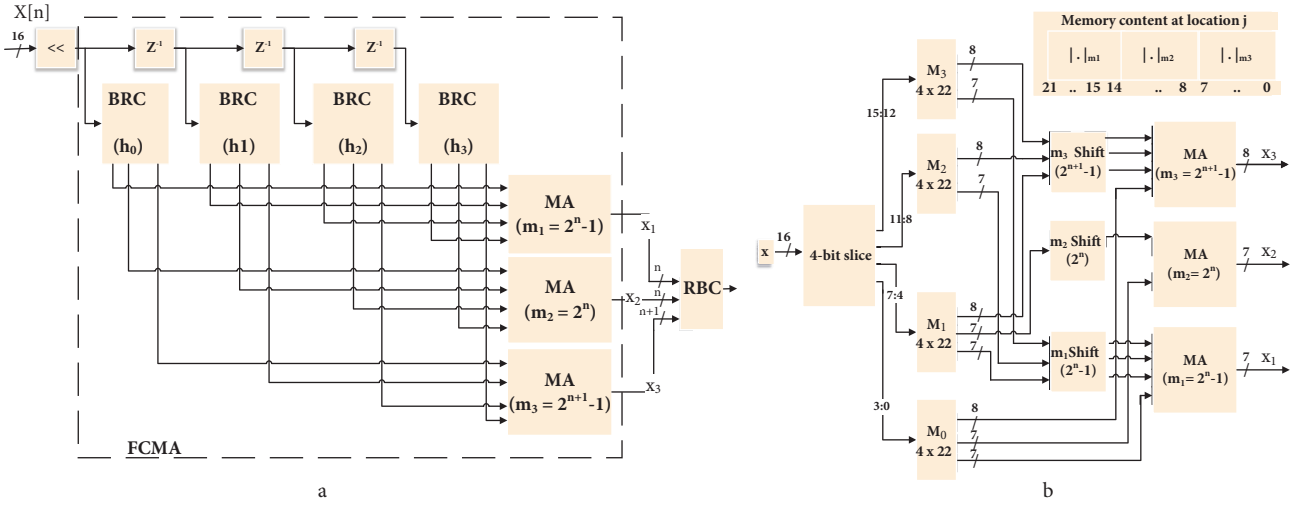

Figure 3. The block diagram of DB2 RNS-based architecture: a) The complete model, b) the block diagram of the BRC for the 3-channel RNS-based, $P_7 = \{127, 128, 255\}$.

The forward converter is used to convert the result of multiplying an input number by a wavelet coefficient to q residue numbers via LUT, shift, and modulo adders, where q is the number of channels.

In RNS system number conversion, the received samples and wavelet coefficients span the real number and might take small values. One of the main drawbacks of RNS number representation is that it only operates with positive integer numbers from $[0, M-1]$. The DWT coefficients are generally between 1 and -1. As a possible solution, we have divided the range of RNS, $[0, M-1]$, to handle those numbers.

In addition, the received sample $x[i]$ is scaled up by shifting y positions to the left (multiplying by 2^y), which ensures that $x[i]$ is a y-bit fixed-point integer. In a similar manner, the wavelet coefficients are scaled by shifting z positions to the left.

In Modulo m_i multiplier, the multiplication of the received sample, $x[i]$, by the filter coefficients, which are constants, is performed by indexing the ROM. As the word length, w, of the received sample $X[i]$ is increased, the memory size becomes 2^w. In addition, q ROMs are required to perform the modulo multiplication.

We propose a few improvements to this design. First, instead of preserving a dedicated memory for each modulo m_i, a ROM that contains all module results is used. Thus, each word at location j contains the q modules of $h_k * j * 2^{11}$. Figure 3b shows the internal BRC block design of the 3-channel moduli set $P_7 = \{127, 128, 255\}$ with its memory map at the top-right corner. It shows that, for a location j, the least significant 8-bit contains $|h_k * x|_{m_3}$, the next 7-bit contains $|h_k * x|_{m_2}$, and the most significant 7-bit contains $|h_k * x|_{m_1}$, which can be generalized as shown in Eq. (8). The advantage of this method is that no extra hardware is required to separate each module value.

$$ROM(j) = |h_k * j * 2^{11}|_{m_1} * 2^{2n+1} + |h_k * j * 2^{11}|_{m_2} * 2^{n+1} + |h_k * j * 2^{11}|_{m_3}, \quad j = [0, 2^w]. \tag{8}$$

As with the DAA-based approach, if the input word length is 16 bits, the ROM should contain 2^{16} locations. One way to reduce the size of the memory is to divide it into four ROMs of 4×22. Figure 3a shows the block diagram of the binary-to-residue converter with four ROMs; each is indexed by four bits of x. However, the output of each ROM should be combined so that the final result can be corrected. It is worth noting that this division comes with a cost in terms of adders and registers.

According to the previous improvements, the RNS-based works are as follows. The input $X_{16-bit} = (x_1, x_2, x_3, x_4)$ is divided into four segments. Each of the 4-bit segments is fed into one ROM so that three outputs corresponding to $|h_k * x_l * 2^{11}|_{m_i}$ are produced. To obtain the final multiplication's result, each m_i output should be shifted by l positions, where l is the index of the lowest input bit (4, 8, or 12). The modular multiplication and shift for $2^n - 1$ and $2^{n+1} - 1$ can be achieved by a left circular shift (left rotate) for l positions, whereas the modular multiplication and shift for 2^n can be achieved by a left shift for l positions [17]. Finally, the modulo adder adds the corresponding output.

Modulo adders (MAs) are required for adding the results from a modular multiplier as well as for a reverse converter. In this work, we have two MAs: one is based on 2^n and the other is based on $2^n - 1$. Modulo 2^n adder is just the lowest n bits of adding two integer numbers, where the carry is ignored.

Figure 4 shows the block diagram of the $2^n - 1$ modulo adder. It shows that MA is slow and less efficient than the conventional binary adder because MA needs one adder and one subtractor to perform the modulo addition. Therefore, the output of each MA will be delayed by one clock cycle compared to the conventional adder. The Chinese remainder theorem (CRT) [15] provides the theoretical basis for converting a residue

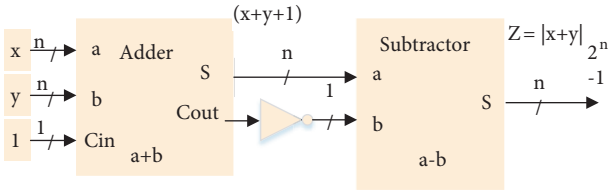

Figure 4. The block diagram of $(2^n - 1)$ modulo adder.

number into a natural integer. The moduli set $P_n = \{2^n - 1, 2^n, 2^{n+1} - 1\}$ can be efficiently implemented by four modulo adders and two multiplexers [19]. The output of the RBC is unsigned $(3 * n + 1)$-bit integer

number. The actual signed number can be found by shifting the result $y+z$ positions to the left, which is equivalent to dividing by $2^{(y+z)}$. Both y and z are the scaled values of the input and wavelet coefficients, respectively. Generally, the word length of 1-level DWT is bounded by Eq. (9) and should not exceed $(3*n-2)$ bits.

$$3*n+1 \geq y+z+3. \tag{9}$$

3.3. Hardware complexity

DAA and RNS techniques employ the memory as a key resource to avoid multiplying two input variables. In each approach, as the number of filter taps increases, both the size and number of memories change. Assuming that the length of the received word is w-bit and there are N filter taps, the size of a memory element can be considered as $a \times b$, where a and b are the word lengths in bits of the input and output, respectively. The value of a determines the size of the memory, 2^a.

The total number of memory elements that are occupied by the DAA-based filter is $w*(N \times 22)$. The output is a 16-bit fixed point and the word length is 22 bits. The number of memory elements remains constant as the filter taps increase, whereas the size of the memory exponentially increases to 2^N.

On the other hand, the total number of memory elements occupied by an RNS-based filter is $N*\lceil \log_2(w) \rceil *(4 \times 22)$. This equation shows that the number of memory elements increases linearly with the number of filter taps, while the memory size remains constant (4×22). Table 1 shows a comparison of the memory usage with $w=16$ for different DWT families.

Table 1. Occupied memories when DAA- and RNS-based approaches are used. The word lengths, w, are 22 bits and 16 bits for DAA- and RNS-based approaches, respectively.

	DB2	DB4	DB10
Number of filter taps	4	8	10
DAA memory usage	$22*(4 \times 22)$	$22*(8 \times 22)$	$22*(10 \times 22)$
RNS memory usage	$16*(4 \times 22)$	$32*(4 \times 22)$	$40*(4 \times 22)$

DAA-based implementation employs shift registers and adders to sum the result at each bit level (Figure 2). For a word length w with m magnitude bits, we need $(w-1)$ shift registers and $(w-1)$ 2-input adders (data combined by a tree adder architecture). To handle the negative numbers, the two's complement operation requires additional $(m-1)$ shift registers and $(m-1)$ adders. Thus, for l-level DAA-based implementation, a total of $l*(w-m-2)$ shift registers and 2-input adders is required.

On the other hand, for a word length w and N-tap filter, the q-channel FCMA implementation requires N BRC blocks and $(q*(N-1))$ MA blocks to compute the final result. Each BRC block has $(\lceil \log_2 w \rceil)-1$, $(\lceil \log_2 n \rceil -1)$, and $(\lceil \log_2 w \rceil)-1$ MA blocks for 2^n-1, 2^n, and $2^{n+1}-1$ modulo, respectively. The modulo 2^n requires $log_2 n$ because shifting operations is not circular where shifting n-bit numbers to the left by n positions or more is always zero. Likewise, the RBC has 4 MA blocks (for $2^{n+1}-1$), 2 multiplexers, and two subtractors. Thus, the total number of MA blocks at 1-level RNS-based is:

$$MA_t = 2N*((\lceil \log_2 w \rceil -1)_{2^n-1}) + (\lceil \log_2 n \rceil -1)_{2^n} + q\ *(N-1)+4. \tag{10}$$

For instance, 3-channel DB2 implementation requires 9 MA blocks to sum the result and P_7 RNS-based implementation has a total of 45 MA blocks when $w = 16$. Meanwhile, the number of RNS-based adders depends on the design of the MA block. For example, each MA block of $(2^n - 1)$ and $(2^{n+1} - 1)$ requires 2 adders, while each MA block of 2^n requires 1 adder. Thus, $a_t = 12N + N(\lceil \log_2 n \rceil - 1) + 5(N-1) + 10$ adders are required, which can be simplified as follows (summarized in Table 2):

$$a_t = 17N + N(\lceil \log_2 n \rceil - 1) + 10. \tag{11}$$

Table 2. Memory usage and adders for 1-level N-tap DAA- and RNS-based DWT.

	DAA-based	RNS-based
Memory usage	$w * (N \times 22)$	$N * \lceil \log_2 w \rceil * (4 \times 22)$
Num. of adders	$w - m - 2$	$17N + N(\lceil \log_2 n \rceil - 1) + 10$

4. Simulation results, performance analysis and validation

Hardware analysis was carried out by using a Xilinx System Generator for DSP, which is a high-level software tool that enables the use of MATLAB/Simulink environment to create and verify hardware designs for Xilinx FPGAs. The hardware-software cosimulation design was synthesized and implemented on ML605 Xilinx Virtex 6.

The implementation of RNS and DAA is compared with the multiplier-accumulate based DWT structure (MAC). We also consider the direct DWT implementation using an IP FIR Compiler 6.3 (FIR6.3) block, which provides a common interface to generate highly area-efficient and high-performance FIR filters. For RNS implementation, the moduli sets of $P_7 = \{127, 128, 255\}$ and $P_{10} = \{1023, 1024, 2047\}$ were used.

4.1. Resource utilization and system performance

Table 3 summarizes the resource use by RNS-based components, i.e., FCMA and reverse converter (RBC). The RBC consumes fewer resources and less power. However, the operating frequency is equal in all models and greater than that of the entire RNS-based filter.

Table 4 summarizes the resource consumption of each filter implementation. It shows that the MAC- and IP FIR-based implementations have 4 multiplier units (DSP48E1s) with maximum frequencies of 296 and 472 MHz, respectively. In contrast, the proposed approaches are more complex than MAC. However, DAA- and

Table 3. The resource use and system performance of the RNS components, i.e., FCMA and reverse converter. FCMA involves the forward converters and modulo adders.

Resources	RNS-based (P_7)		RNS-based (P_{10})	
	FCMA	RBC	FCMA	RBC
Number of slice registers	656	157	883	190
Number of slice LUTs	591	138	854	180
Number of RAMB18E1	16	0	16	0
Max. operating freq. (MHz)	291.2	311.62	283.85	298.67

Table 4. Resource use and system performance for the DWT implementation.

Resources	MAC	DAA	FIR	RNS(P_7)	RNS(P_{10})
Number of slice registers	282	661	167	767	1089
Number of slice LUTs	128	520	71	721	1055
Number of occupied slices	58	188	60	240	358
Number of RAMB18E1	0	22	0	16	16
Max. operating freq. (MHz)	296.38	229.83	472.59	258.86	261.028

RNS-based implementations have 22 and 16 memory blocks (BRAMs) used to store the precalculated wavelet coefficients. It also shows that the number of slice registers, slice LUTs, and occupied slices of P_{10} RNS-based is greater than that of the P_7 because the former has 31 output signals, while the latter has 22 output signals. As a result, the number of flip-flops is increased and the number of resources is approximately increased by one third, while the maximum frequency in both designs is greater than 235 MHz.

Table 5 shows a comparison between the DAA- and RNS-based 1-level DWT implementations when using larger filter banks, i.e., DB4 and DB5. It shows that the DAA-based implementation occupies a fixed number of RAMB18E1. Figure 5 depicts the impact of using different wavelet families on memory. The number of memory elements of the DAA-based implementation is fixed and depends on the word length (Table 2).

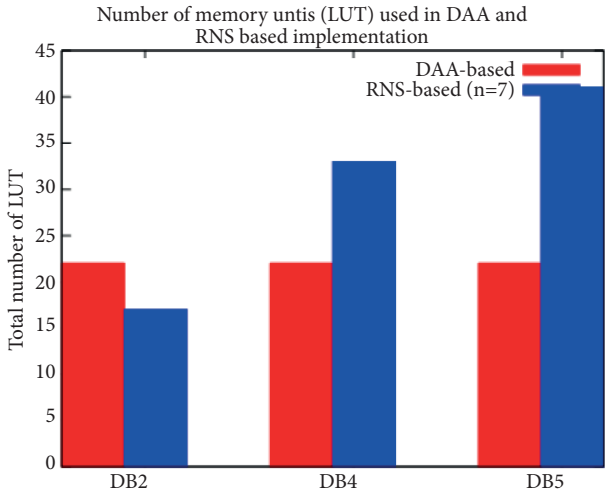

Figure 5. Occupied memory by the first level of DAA- and RNS-based DWT.

However, as the number of filter taps increases, the memory size exponentially increases to 2^N. In contrast, the number of memory elements that are used in the RNS-based implementation linearly increases as the number of filter taps increases. Similarly, the number of memories that are used at multilevel DAA-based and RNS-based implementations with the l-level would be an aggregate of levels 1 through l.

Table 6 shows the performance based on the signal-to-noise-ratio (SNR) and peak-signal-to-noise-ratio (PSNR). Both DAA- and RNS-based approaches offer high signal quality with a PSNR of 73.5 and 56.5 dB, respectively.

Table 5. Resource use for the DWT implementation of DB2, DB4, and DB5.

Resources	DAA-based			RNS-based (P_7)		
	DB2	DB4	DB5	DB2	DB4	DB5
Number of slice registers	650	737	780	767	1441	1898
Number of slice LUTs	521	539	568	721	1320	1677
Number of RAMB18E1	22	22	22	16	32	40
Max. operating frequency (MHz)	232.7	205.5	223.3	258.9	265.3	258.8

Table 6. The SNR and PSNR values of 1-Level of different DWT implementations.

				RNS-based	
	FIR	MAC	DAA	P_7	P_{10}
SNR (dB)	83.2	78.7	70.4	53.41	54.78
PSNR (dB)	86.3	81.8	73.5	56.5	57.9

4.2. Comparison

Table 7 provides a quantitative, comprehensive, and comparative study of different implementation methods of DWT, including the convolutional MAC [20], Parallel MAC loop-based filter (PMLBF) [21], and algebraic integer (AI) [22], as well as our implementation of 1-D 1-level DWT. From the data, we observed that the proposed 22-bit DAA-based architecture almost has the same complexity as the lowest-complexity implementation, as seen in [21]. However, our DAA- and RNS-based architectures consume less power. Even if the architectures in [21] and [22] have a throughput of one sample per clock cycle, the lower power required by DAA- and RNS-based implementations makes them superior in terms of throughput to power consumption.

4.3. Discussion

Hardware availability and system performance requirements are critical for selecting the appropriate architecture of the embedded platform. Overall, the number of filter taps and word length have a substantial influence.

Because the only difference between P_7 and P_{10} RNS-based implementations is the signal width, the maximum operating frequencies slightly changes. Furthermore, the 1-level DB2 filter bank was designed with maximum operating frequencies of 232 and 258 MHz for DAA- and RNS-based approaches, respectively. However, all high-frequency implementations introduce a latency of at least 10 clock cycles for 1-level DAA-based DWT.

Another critical parameter that affects the DWT performance is the filter order. The DAA-based implementation outperforms the RNS-based with at most 10 taps. When the number of taps increases, the number of memory units and binary adders within the RNS-based implementation constantly increases, while the memory size is not affected (Table 2). The memory requirement for DAA-based implementation exponentially increases as the number of filter taps increases.

Table 7. The SNR and PSNR values of 1-Level of different DWT implementations.

	Jarrah [20]	Mamatha [21]	Madishetty [22]	Avinash [23]	Current work	
Technology	Artix-7	Virtex-6	Virtex-6	Virtex-5	Virtex-6	
Wavelet type	db1(Haar)	db4	db3	db2	db2	
Architecture	MAC	MAC *	AI **	DAA	DAA	RNS
Word length	N/A	8-bit	8-bit	8-bit	22-bit	
Slice reg.	2247 (1.77%)	139 (0.046%)	2890 (0.95%)	66 (0.52%)	661 (0.22%)	1089 (0.36%)
Slice LUTs	N/A	417 (0.39%)	5470 (5.17%)	136 (1.01%)	520 (0.49%)	1055 (0.99%)
Occupied slices	N/A	N/A	N/A	61 (86%)	188 (0.5%)	358 (0.95%)
Maximum freq. (MHz)	214.6	633.4	344	301.8	229.8	261.0
Dynamic power (mW)	101	632	204	N/A	66.54	53.05
DSP48Es	N/A	16	N/A	0	0	0
Throughput	N/A	1 ip/op	1 ip/op	8 ip/op	2 ip/op	2 ip/op

*Parallel MAC loop-based filter (PMLBF); a pipelined real-time architecture.

**Algebraic integer (AI).

5. Conclusion

The performance requirements and hardware resource availability has a great impact on choosing the proper algorithm for implementing a DWT on FPGA. In this paper, we addressed the effect of increasing the number of filter taps and word length, which have a substantial influence on the overall performance of the design and resource availability. Moreover, we presented pipelined DAA- and RNS-based implementations and compared them with the pipelined MAC-based approach. DAA- and RNS-based approaches are multiplierless architectures that intensively use memory to speed up the entire processing time. The trade-off between system performances and resource consumption was also presented. Experiment results show that the DAA-based approach is appropriate for a small number of filter taps, while the RNS-based approach would be more appropriate for a number of filter taps that are greater than 10, yet both DAA- and RNS-based approaches offer high signal quality with PSNR as 73.5 and 56.5 dB, respectively.

References

[1] Martina M, Masera G, Roch MR, Piccinini G. Result-biased distributed-arithmetic-based filter architectures for approximately computing the DWT. IEEE Trans Circuits Syst I 2015; 62: 2103-2113.

[2] Alzaq H, Ustundag BB. Very-low-SNR cognitive receiver based on wavelet preprocessed signal patterns and neural network. EURASIP J Wirel Comm 2017; 2017: 120.

[3] Carta N, Pani D, Raffo L. Impact of threshold computation methods in hardware wavelet denoising implementations for neural signal processing. In: Plantier G, Schultz T, Fred A, Gamboa H, editors. Biomedical Engineering Systems and Technologies. Cham, Switzerland: Springer International Publishing, 2015. pp. 66-81.

[4] Mallat S. A Wavelet Tour of Signal Processing. 3rd ed. Philadelphia, PA, USA: The Sparse Way Academic Press, 2008.

[5] Vishwanath M. The recursive pyramid algorithm for the discrete wavelet transform. IEEE Trans Signal Proces 1994; 42: 673–676.

[6] Vetterli M, Herley C. Wavelets and filter banks: theory and design. IEEE Trans Signal Proces 1992; 40: 2207-2232.

[7] Chen S, Chen Y. Hardware design and implementation of a wavelet de-noising procedure for medical signal preprocessing. Sensors 2015; 15: 26396-26414.

[8] Duan F, Dai L, Chang W, Chen Z, Zhu C, Li W. sEMG-based identification of hand motion commands using wavelet neural network combined with discrete wavelet transform. IEEE Trans Ind Electron 2016; 63: 1923-1934.

[9] Daubechies I, Sweldens W. Factoring wavelet transforms into lifting steps. J Fourier Anal and Appl 1998; 4: 247-269.

[10] Meher PK, Mohanty BK, Swamy MMS. Low-area and low-power reconfigurable architecture for convolution-based 1-D DWT using 9/7 and 5/3 filters. In: 28th International Conference on VLSI Design; 3-7 Jan. 2015; Bangalore, India. New York, NY, USA: IEEE. pp. 327-332.

[11] Madanayake A, Cintra RJ, Dimitrov V, Bayer F, Wahid KA, Kulasekera S, Edirisuriya A, Potluri U, Madishetty S, Rajapaksha N. Low-power VLSI architectures for DCT/DWT: precision vs approximation for HD video, biomedical, and smart antenna applications. IEEE Circuits Syst Mag 2015; 15: 25-47.

[12] Taylor F. Residue arithmetic: a tutorial with examples. Computer 1984; 17: 50-62.

[13] White S. Applications of distributed arithmetic to digital signal processing: a tutorial review. ASSP Mag, IEEE 1989; 6: 4-19.

[14] Pontarelli S, Cardarilli G, Re M, Salsano A. Optimized implementation of RNS FIR filters based on FPGAs. J Signal Process Sys 2012; 67: 201-212.

[15] Rosen KH. Elementary Number Theory and its Applications, 5th ed. New York, NY, USA: Addison-Wesley, 2004.

[16] Cardarilli GC, Nannarelli A, Petricca M, Re M. Characterization of RNS multiply-add units for power efficient DSP. In: IEEE International Midwest Symposium on Circuits and Systems; 2-5 Aug. 2015; Fort Collins, FO, USA. New York, NY, USA: IEEE. pp. 1-4.

[17] Reddy K, Bajaj S, Kumar S. Shift add approach based implementation of RNS-FIR filter using modified product encoder. In: TENCON IEEE Region Conference; 22-25 Oct. 2014; Bangkok, Thailand. New York, NY, USA: IEEE. pp. 1-6.

[18] Hariri A, Navi K, Rastegar R. A new high dynamic range moduli set with efficient reverse converter. Comput Math Appl 2008; 55: 660-668.

[19] Lin S, Sheu M, Wang C, Kuo Y. Area-time-power efficient VLSI design for residue-to-binary converter based on moduli set (2n, 2(n + 1)-1, 2n + 1). In: IEEE Asia Pacific Conference on Circuits and Systems; 30-31 Nov. 2008; Macao, China. New York, NY, USA: IEEE. pp. 168-171.

[20] Jarrah A, Jamali MM. Optimized FPGA based implementation of discrete wavelet transform. In: Asilomar Conference on Signals, Systems and Computers; 2-5 Nov. 2014; Pacific Grove, CA, USA. New York, NY, USA: IEEE. pp. 1839-1842.

[21] Mamatha I, Tripathi S, Sudarshan TSB. Pipelined architecture for filter bank based 1-D DWT. In: International Conference on Signal Processing and Integrated Networks; 11-12 Feb. 2016; Noida, India. New York, NY, USA: IEEE. pp. 47-52.

[22] Madishetty SK, Madanayake A, Cintra RJ, Dimitrov VS. Precise VLSI architecture for AI based 1-D/ 2-D Daub-6 wavelet filter banks with low adder-count. IEEE Trans Circuits Syst I 2014; 61: 1984-1993.

[23] Avinash CS, Alex JSR. FPGA implementation of discrete wavelet transform using distributed arithmetic architecture. In: International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials; 6-8 May 2015; Chennai, India. New York, NY, USA: IEEE. pp. 326-330.