We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

154
Countries delivered to

Our authors are among the

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us? Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter 6

A Comparative Performance of Discrete Wavelet Transform Implementations Using Multiplierless

Husam Alzaq and Burak Berk Üstündağ

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.76522

Abstract

Using discrete wavelet transform (DWT) in high-speed signal-processing applications imposes a high degree of care to hardware resource availability, latency, and power consumption. In this chapter, the design aspects and performance of multiplierless DWT is analyzed. We presented the two key multiplierless approaches, namely the distributed arithmetic algorithm (DAA) and the residue number system (RNS). We aim to estimate the performance requirements and hardware resources for each approach, allowing for the selection of proper algorithm and implementation of multi-level 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 (BRAMs).

Keywords: discrete wavelet transform (DWT), distributed arithmetic algorithm (DAA), field programmable gate array (FPGA), residue number system (RNS), multiplierless implementation

1. Introduction

The architecture of the embedded platform plays a significant role in ensuring that real-time systems meet the performance requirements. Moreover, software development suffers from increased implementation complexity and a lack of standard methodology for partitioning the implementation of signal-processing functionalities to heterogeneous hardware platforms. For instance, digital signal processor (DSP) is cheaper, consumes less power, and is easy to develop software applications, but it has a considerable latency and less throughput compared with field programmable gate arrays (FPGAs) [1]. For high-speed signal-processing (HSP) communication systems, such as cognitive radio (CR) [2, 3] and software-defined radio (SDR) [4], DSP may fail to capture and process the received data due to data loss. In addition, implementing

applications such as finite impulse response (FIR) filtering, discrete wavelet transform (DWT), or fast Fourier transform (FFT) by software application limits the throughput, which is not sufficient to meet the requirements of high-bandwidth and high-performance applications. As a result, HSP systems are enhanced by off-loading complex signal-processing operations to hardware platforms.

Although FPGAs exhibit an increased development time and design complexity, they are preferred to meet high-performance requirements for two reasons. First, they efficiently address signal-processing tasks that can be pipelined. Second, they have the capacity to develop a programmable circuit architecture with the flexibility of computational, memory, speed, and power requirements. However, FPGA has its own resources such as memory, configurable logic blocks (CLBs), and multipliers that influence on the performance and selected algorithm. As a consequence, the choice of algorithm is determined by the hardware resource availability and performance requirements. These factors have an impact on each other and create many challenges that need to be optimized.

As an example, the discrete wavelet transform (DWT) [5–9], a linear signal-processing technique that transforms a signal from the time domain to the wavelet domain [10], employs various techniques for signal decomposing into an orthonormal time series with different frequency bands. The signal decomposition is performed using a pyramid algorithm (PA) [10, 11] or a recursive pyramid transform (RPT) [12]. While the PA algorithm is based on convolutions with quadrature mirror filters, which is infeasible for HW implementation, RPT decomposes the signal x[n] into two parts using high- and low-pass filters, which can be implemented using FIR filter [13]. **Figure 1** shows a four-tap FIR filter with four multipliers, named as multiplier accumulator (MAC). By using the MAC structure, multipliers are involved in multiplying an input with filter coefficients, $\boldsymbol{b_i}$. It is clear that the direct implementation of the N-tap filter requires N multipliers.

This work focuses exclusively on implementing a one-level multiplierless DWT for a pattern-based cognitive communication system receiver (PBCCS) [8] by means of FPGA. The DWT is required to extract the received signal's features. Then, the extracted features are fed into a multilayer perceptron (MLP) neural network (NN) to identify the received symbol. The most challenging part is that the NN could consume most of the available multipliers inside the FPGA. As an example, Ntoune et al. [14] have implemented a real-valued time-delay neural network (RVTDNN) and real-valued recurrent neural network (RVRNN) architecture with 600

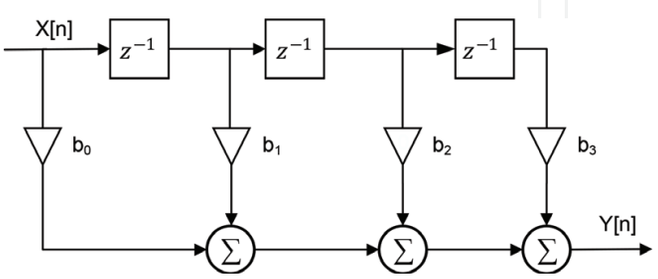

Figure 1. Four-tap finite impulse response filter.

and 720 multipliers, respectively, while ML605 [15], ZC706 [16], and VC707 [17] have 768, 900, and 2800 multipliers (DSP48Es), respectively.

Although the modern FPGAs come with a reasonable number of multipliers, designers prefer to implement multiplier-free DWT architecture for many reasons. First, a partial number of multipliers can be preserved for tasks, such as pulse shape filter, digital-up and digital-down converter that are used at SDR front-ends. Second, in contrast to DWT, the MLP weights depend on the learning step. Third, MLP weights could be frequently changed at runtime in an adaptive manner, whereas the DWT coefficients are fixed and known. Therefore, the multiplier-free DWT architecture could simplify the design process and allow the designers to focus on the MLP design.

In this work, we present the 1-D DWT implementation on FPGA by means of memory-based approaches. The aim is to compare different implementations in terms of system performance and resource consumption. We demonstrate the implementation of Daubechies wavelets (DB2, DB4, and DB5) using DAA and RNS approaches. These approaches do not employ explicit multipliers in the design. Because the main focus of this work is on extracting the key features of a signal via DWT, the inverse DWT (IDWT) and high-pass filter coefficients are not considered in this work.

1.1. Related work

Implementations of 1-D DWT for signal de-noising, feature extraction, and pattern recognition and compression can be found in [8, 9, 18, 19]. 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, which is introduced by Sweldens [20]. Although, the lifting scheme is used to compute the output of low- and high-pass using fewer components, it may not be well suited to our application, owing to the PBCCS's nature, where the low-frequency components are much important than the higher ones. Therefore, in this study, 1-D DWT decomposition, which is implemented by means of filter banks, is considered. Another advantage of using convolution-based DWT over lifting approach is that they do not require temporary registers to store the intermediate results, and with an appropriate design strategy, they could have better area and power efficiency [21].

Rather than the simplest implementation of FIR filter via multipliers and an adder tree, a multiplier-free architecture is used because they result in low-complexity systems and for their high-throughput-processing capability [22]. Fundamentally, there are two techniques for facilitating parallel processing. They are the distributed arithmetic algorithm (DAA) and the residue number system (RNS). DAA is an algorithm that performs the inner product in a bit serial with the assist of a lookup table (LUT) scheme followed by shift accumulation operations [23, 24]. Several techniques have been proposed to improve the design, such as the partial sum technique [25], a multiple memory bank technique [26, 27], and an LUT-less adder-based [28]. The DAA approach has been adapted in many applications, such as least mean square (LMS) adaptive filter [29] and square-root-raised cosine filter [30].

On the other hand, RNS is an integer number system, in which the operations are performed based on the residue of division operation [31–33]. Eventually, the RNS-based results are converted back to the equivalent binary number format using a Chinese reminder theorem (CRT) [34]. 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 implementation, for example, [35–37]. In addition, RNS was combined with DAA in one architecture, called RNS-DA [38, 39], which benefits from the advantages of both approaches.

In this chapter, three major 1-D DWT approaches are implemented on FPGA-based platforms and compared in terms of performance and energy requirements. The implementations are compared for different number 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 study. 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 study, we did not optimize the number of memories in any approach, so that we have a fair comparison.

The remainder of this chapter is organized as follows. Section 2 presents the preliminaries information to understand DWT. It also reviews the theoretical background of DAA and RNS. Section 3 describes the implementation of discrete wavelet transform. We further show an analytical comparison between these approaches. Section 4 presents the performance results. Finally, this chapter concludes in Section 5.

2. Fundamentals and basic concepts

2.1. Discrete wavelet transform

The wavelet decomposition mainly depends on the orthonormal filter banks. **Figure 2** shows a two-channel wavelet structure for decomposition, where $x[n]$ is the input signal, $g[n]$ is the

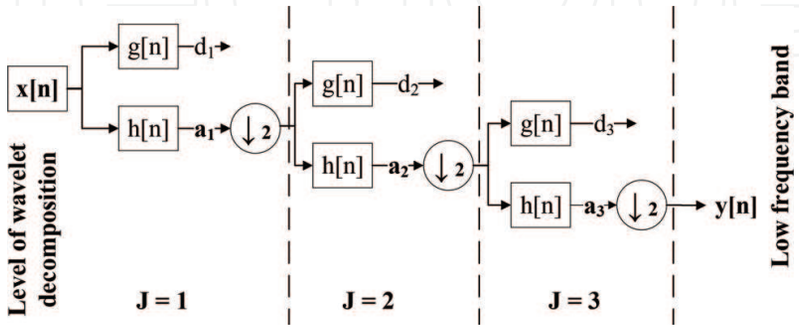

Figure 2. Multi-resolution wavelet decomposition. The block diagram of the two-channel four-level discrete wavelet transform decomposition ($J = 3$) that decomposes a discrete signal into two parts. Note that ↓**2** is maintaining one sample out of two, $\mathbf{a_i}$ and $\mathbf{d_i}$ are the approximation and details at level i, respectively.

high-pass filter, $\boldsymbol{h[n]}$ is the low-pass filter, and ↓2 is the down-sampling by a factor of two. The output of each low-pass filter is fed to the next level, so that each filter creates a series of coefficients (a_i and d_i), which represent and compact the original signal information.

Mathematically, a signal $\boldsymbol{y[n]}$ 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 **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. Eqs. (2) and (3) represent the filtering process through a digital low-pass filter $\boldsymbol{h[k]}$ and high-pass filter $\boldsymbol{g[k]}$, corresponding to a convolution with an impulse response of $\boldsymbol{k}$-tap filters

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

$$y_{low}[n] = \sum_{k} g\ [k].x\ [2n-k] \tag{3}$$

where $\boldsymbol{2n}$ is the down-sampling process. The outputs $\boldsymbol{y_{low}\ [n]}$ and $\boldsymbol{y_{high}\ [n]}$ provide an approximation signal and of the detailed signal, respectively [40].

2.2. Distributed arithmetic algorithm

The distributed arithmetic algorithm (DAA) gets rid of multipliers by performing the arithmetic operations in a bit-serial computation [13]. Because the down-sampling process follows each filter (as shown in **Figure 2**), Eq. (2) can be rewritten without the decimation factor as

$$y_{low}\ [n] = \sum_{k=0}^{N-1} x\ [k]\ \ .\ \ h\ [k] \tag{4}$$

Obviously, Eq. (4) requires an intensive operation due to multiplication of the real input values with the filter coefficients. Eq. (3) can be simplified by representing $\boldsymbol{x[k]}$ as a fixed point arithmetic of length $\boldsymbol{L}$:

$$x[k] = -\ \ x[k]_0 + \sum_{l=1}^{L-1} x\ [k]_l.2^{-l} \tag{5}$$

where $\boldsymbol{x[k]_l}$ is the lth bit of $\boldsymbol{x[k]}$ and $\boldsymbol{x[k]_0}$ is the sign bit. Substituting Eq. (5) into Eq. (4), the output of the filter becomes

$$y\ [n] = \left[\sum_{l=1}^{L-1} 2^{-l}.\sum_{k=0}^{N-\ 1} h[k].x[k]_l\right] \quad + \quad \sum_{k=0}^{N-\ 1} h[k]\ \left(-x[k]_0\right) \tag{6}$$

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 possible to pre-compute and store these values in a ROM, indexed by $x[k]_l$. In other words, Eq. (6) simply realizes the sum of product computation by memory (LUT), adders, and shift register.

2.3. Residue number system

The RNS is a non-weighted number system that performs parallel carry-free addition and multiplication arithmetic. In DSP applications, which require intensive computations, the carry-free propagation allows for a 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; the greatest common divisor (GCD) $(m_i, m_j) = 1$ for $i \neq j$.

Any number, $X \in Z_M = 0, 1, \ldots, M - 1$, is uniquely represented in RNS by its residues $|X|_{m_i}$, which is the remainder of division X by m_i and M is defined in Eq. (7) as

$$M = \Pi_{i=1}^{q} m_i = m_1{}^* \ m_2{}^* \ \ldots \ {}^* \ m_q \tag{7}$$

where M determines the range of unsigned numbers in $[0, M - 1]$, and should be greater than the largest performed results. In addition, M uniquely represents any signed numbers. The implementation of RNS-based DWT obtained from Eq. (4) is given by Eq. (8) as follows:

$$y[n]_{m_i} = y_{m_i} = \left| \left(\sum_{k=0}^{N-1} |h\ [k]_{m_i}.x\ [n-k]_{m_i} \Big|_{m_i} \right) \right|_{m_i}. \tag{8}$$

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

Mapping from the RNS system to integers is performed by the Chinese reminder theorem (CRT) [34, 41, 42]. The CRT states that binary/decimal representation of a number can be obtained from its RNS if all elements of the moduli set are pairwise relatively prime.

Designing a robust RNS-based DWT requires selecting a moduli set and implementing the 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, $\{2^n - 1\ ,\ 2^n,\ 2^{n+1} - \ 1\}$ [43], $\{2^n - 1\ ,\ 2^n,\ 2^n + \ 1\}$ [39], and $\{2^n,\ 2^{2n} - \ 1,\ \ 2^{2n} + \ 1\}$ [44] have been investigated.

For the purpose of illustrating, 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 = 4,145,280**, which could efficiently express real numbers in the range $[-2.5, 2.5]$ using a **16**-bit fixed-point representation, provided scaling and rounding are done properly. We assume that this interval is sufficient to map the input values, which does not exceed ± 2. Third, the reverse converter unit is simple and regular [42] due to using simple circuits design.

3. DWT implementation methodology

3.1. DWT implementation using DA

DAA hides the explicit multiplications with a ROM lookup 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-point of a **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 memory, the width should be reduced, which will have an impact on the output precision.

Figure 3 shows the block diagram of 1-bit DAA at position l. This block contains one ROM ($\mathbf{4} \times \mathbf{22}$) and one shift register. Because the word's length w of the input x is **22** bits, the actual design contains **22** memory blocks and **21** adders for summing up the partial results.

3.2. DWT implementation using RNS

The RNS-based DWT implementation has mainly three components. They are the forward converter, the modulo adders (MAs), and the 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. By 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 refer to the RNS system, which does not include RBC, as a forward converter and modular-adders (FCMA), as illustrated in **Figure 4**.

3.2.1. The forward converter

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.

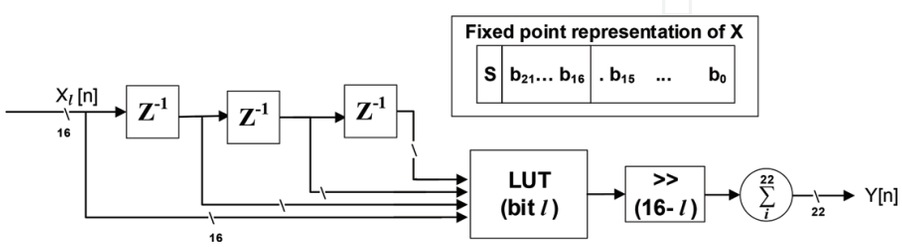

Figure 3. The block diagram of DAA-based architecture of the DB2. For simplicity, we showed one ROM and one shift register. In the actual design, there are **22** ROMs and shift registers. >> is a $\mathbf{16} - l$ shift operation, where **16** is the number of the binary point.

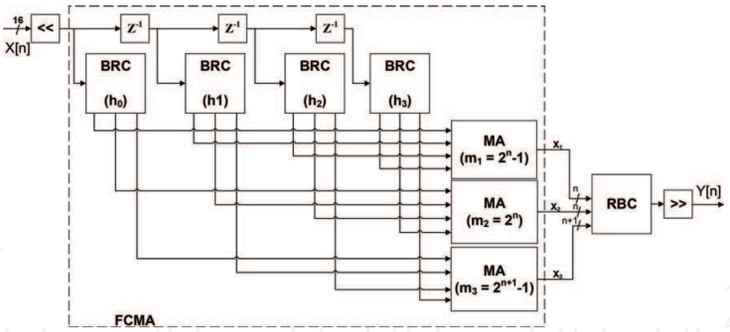

Figure 4. The block diagram of DB2 RNS-based architecture. BRC is an abbreviation for binary-to-residue converter, RBC for residue-to-binary converter, and MA for modulo adder.

Coefficient(h_k)	Real value	RNS-system value
h_0	−0.129409522550921	−266
h_1	0.224143868041857	459
h_2	0.836516303737469	1713
h_3	0.482962913144690	989

Table 1. The DB2 low-pass real and RNS-system number equivalent, multiplied by 2^{11}.

3.2.2. 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, $\boldsymbol{M}$ – 1]. The DWT coefficients are generally between **1** and − **1**. As a possible solution, we have divided the range of RNS, [0, $\boldsymbol{M}$ – 1], to handle those numbers.

In addition, the received sample $\boldsymbol{X[i]}$ is scaled up by shifting $\boldsymbol{y}$ positions to the left (multiplying by $\mathbf{2}^{\boldsymbol{y}}$), which ensures that $\boldsymbol{X[i]}$ is a $\boldsymbol{y}$-bit fixed point integer. In a similar manner, the wavelet coefficients are scaled by shifting its z positions to the left. In our design, we set the filter scaling factor z to **11**. **Table 1** presents the low-pass filter of DB2 before and after scaling.

3.2.3. Modulo $\mathrm{m_i}$ *multiplier*

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

We propose few improvements to this design. First, instead of preserving a dedicated memory for each modulo $\boldsymbol{m_i}$, 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 5** shows the internal BRC block design of the three-channel moduli set $\boldsymbol{P_7}$ **= {127, 128, 255}** with its memory map at the right top corner. It shows that, for a location j, the least significant **8** bit contains $|h_k{}^* x|_{m_3}$, the next **7** bit

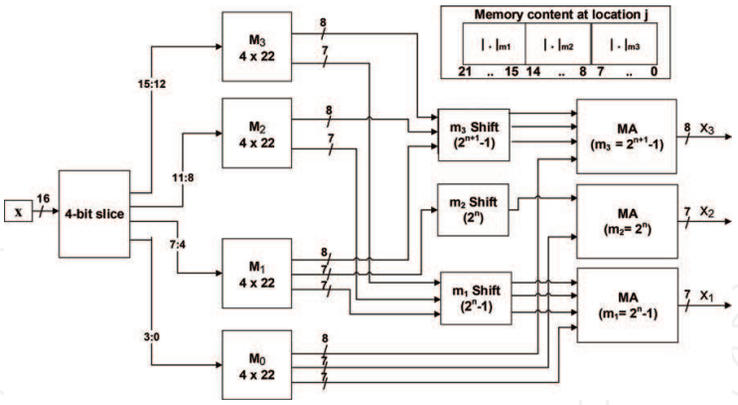

Figure 5. The block diagram of the binary-to-residue converter for the three-channel RNS-based DWT, $\mathbf{P_7 = \{127, 128, 255\}}$. Four identical memories are used for each tap. The upper corner shows the memory content at location $\mathbf{j \in [0, 15]}$.

contains $|h_k{}^* x|_{m_2}$ and the most significant **7** bit contains $|h_k{}^* x|_{m_1}$, which is generalized by Eq. (9). The advantage of this method is that no extra hardware is required to separate each module value.

$$ROM(j) = |h_k{}^*\, j{*}2^{11}|_{m_1}{}^*\, 2^{2n+1} + |h_k{}^* j{*}2^{11}|_{m_2}{}^*\, 2^{n+1} + \left|h_k{}^* j^*\, 2^{11}\right|_{m_3}, \quad j = [0, 2^w] \tag{9}$$

As with DAA-based approach, if the input word length is **16** bits, the ROM should contain $\mathbf{2^{16}}$ locations. One way to reduce the size of the memory is to divide it into four ROMs of $\mathbf{4 \times 22}$. **Figure 4** shows the block diagram of the binary-to-residue converter with four ROMs; each is indexed by four bits of $\boldsymbol{x}$. 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 segment is fed into one ROM, so that three outputs, corresponding to $\left|h_k{}^* x_l{*}2^{11}\right|_{m_i}$, are produced.

To obtain the final multiplications' result, each $\boldsymbol{m_i}$ output should be shifted by $\boldsymbol{l}$ positions, where $\boldsymbol{l}$ is the index of the lowest input bit (**4, 8,** or **12**). The modular multiplication and shift for $(\mathbf{2^n - 1})$ and $(\mathbf{2^{n+1} - 1})$ can be achieved by a left circular shift (left rotate) for $\boldsymbol{l}$ positions, whereas the modular multiplication and shift for $\mathbf{2^n}$ can be achieved by a left shift for $\boldsymbol{l}$ positions [17]. Finally, the modulo adder adds the corresponding output (**Figure 6**).

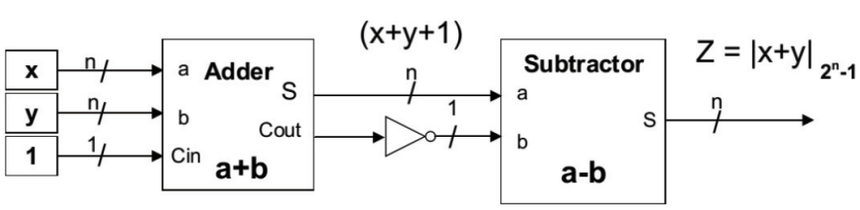

Figure 6. The block diagram of $(\mathbf{2^n - 1})$ modulo adder.

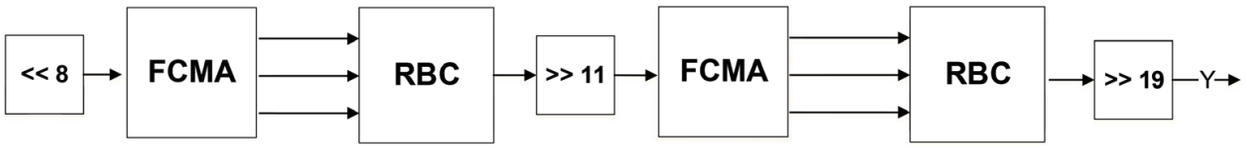

Figure 7. The block diagram of two-level RNS-based DWT design, and FCMA represents FIR-filtering process in RNS.

3.2.4. Modulo adder (MA)

The modulo adders are required for adding the results from a modular multiplier as well as for a reverse converter. In this work, we have two MAs—that is, one is based on $\mathbf{2^n}$ and the other is based on $\mathbf{2^n - 1}$. Modulo $\mathbf{2^n}$ adder is just the lowest $\boldsymbol{n}$ bits of adding two integer numbers, where the carry is ignored. **Figure 7** shows the block diagram of the $2^n - 1$ modulo adder.

3.2.5. The reverse converter

The Chinese remainder theorem (CRT) [34] provides the theoretical basis for converting a residue 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 [42]. The output of the RBC is unsigned ($\mathbf{3 * n + 1}$)-bits integer number. The actual signed number can be found by shifting the result $\boldsymbol{y + z}$ positions to the left, which is equivalent to dividing by $\mathbf{2^{(y+z)}}$. $\boldsymbol{y}$ and $\boldsymbol{z}$ are the scaled values of the input and wavelet coefficients, respectively. Generally, the word length of one-level DWT is bounded by Eq. (10) and should not exceed ($\mathbf{3 * n - 2}$) bits

$$3^* n + 1 \geq y + z + 3 \tag{10}$$

As a consequence, the range of the moduli set should be greater than the maximum output, th_o, which can be computed as follows:

$$th_o = \left(\sum_k h_k\right)^2 * \max(x[n]) * (2^z)^2 * 2^y \leq M - 1 \tag{11}$$

where $\boldsymbol{h_k}$ is the $\boldsymbol{k^{th}}$ filter coefficient, $\boldsymbol{x[n]}$ is the input, y and z are the input and filter scaling factors, respectively, and $\boldsymbol{M}$ is the maximum range.

3.3. Two-level DWT implementation

The two-level discrete wavelet transform compromises two one-level DWTs, where the output of the first level is fed into the second level (as shown in **Figure 7**). The one-level DWT at each level is identical, but the output of each level is halved. For example, if a signal of 1800 samples is applied to the input, then 900 and 450 samples are produced by the first and second levels, respectively.

Figure 7 shows the design of two-level RNS-based DWT, which involves two FCMA blocks and two RBC blocks. Each FCMA requires converting the result of the first stage to binary, shifting the number by 11 and converting it to residue number again.

3.4. Hardware complexity

3.4.1. Memory usage

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 the 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 length 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 fixed **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 2** shows a comparison of the memory usage with $w = 16$ for different DWT families.

3.4.2. Shift register and adder counts

DAA-based implementation employs shift registers and adders to sum the result at each bit level (**Figure 3**). 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 DA-based implementation, a total of $l * (w - m - 2)$ shift registers and two-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 and shifting **n**-bit numbers to the left by n positions or more is always zero. Likewise, the RBC has four MA blocks (for $2^{n+1} - 1$), two multiplexers, and two subtractors. Thus, the total number of MA blocks at one-level RNS-based is

	DB2	DB4	DB5
Number of filter taps	4	8	10
DA memory usage	22*(4 × 22)	22*(8 × 22)	22*(10 × 22)
RNS memory usage	16*(4 × 22)	32*(4 × 22)	40*(4 × 22)

Table 2. Occupied memories when DA- and RNS-based approaches are used. The word length, w, is 22 and 16 bits for DA- and RNS-based, respectively.

$$MA_t = 2N * \left(\left(\lceil \log_2 w \rceil - 1\right)_{2^n-1}\right) + \left(\lceil \log_2 n \rceil - 1\right)_{2^n} + \ q * (N-1) \ + \ 4 \tag{12}$$

For instance, three-channel DB2 implementation requires nine 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 two adders, while each MA block of 2^n requires one 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 3**):

$$a_t = 17N + N\left(\lceil \log_2 n \rceil - 1\right) + 10 \tag{13}$$

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

Table 3. Memory usage and adders for **1-L N**-tap DA and RNS-based approaches DWT.

4. Performance analysis and validation

Hardware analysis was carried out by using a Xilinx System Generator for DSP (SysGen) [45], which is a high-level software tool that enables the use of MATLAB/Simulink environment to create and verify hardware designs for Xilinx FPGAs. It enables the use of the MathWorks model-based Simulink design environment for FPGA design. Furthermore, the hardware-software co-simulation design was synthesized and implemented on ML605 Xilinx Vertex 6 [15].

The implementation of RNS and DA is compared with the multiplier-accumulate-based DWT structure (MAC), as shown in **Figure 8**. We also consider the direct DWT implementation using an IP FIR Compiler 6.3 (FIR6.3) block [46], which provides a common interface to generate highly area-efficient and high-performance FIR filters (**Figure 9**).

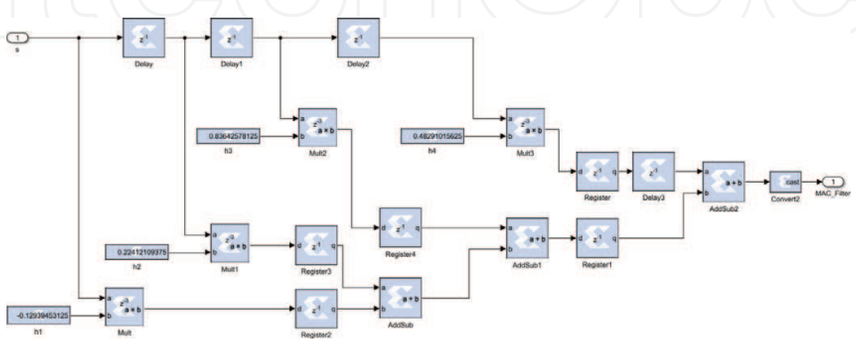

Figure 8. The Simulink model of MAC-based one-level DB2 discrete wavelet transform. Filter coefficients are stored as constants.

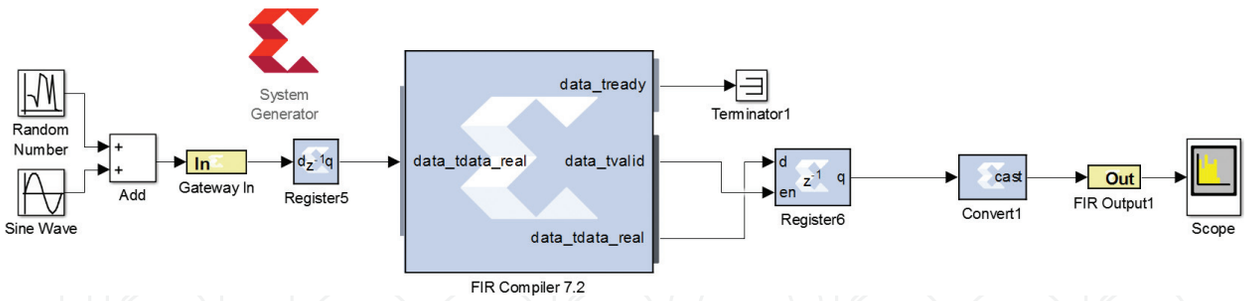

Figure 9. The Simulink model of FIR-based one-level DB2 discrete wavelet transform. The IP FIR compiler 6.3 of the system generator is used.

For RNS implementation, the moduli sets of $\boldsymbol{P_7}$ **= {127, 128, 255}** and $\boldsymbol{P_{10}}$ = **{1023, 1024, 2047}** were used. The dynamic ranges of these sets are M = 4,161,536 and 2,144,338,944, respectively. The moduli set of $\boldsymbol{P_{10}}$ is selected because its dynamic range is greater than th_o (Eq. (11) with $y = 6$, $z = 11$ and $\sum(hi) = 1.5436$). In all RNS-based implementations, the word length was set to 16 bits.

4.1. Resource utilization and system performance

Table 4 summarizes the resource use by RNS-based components—that is, 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 the entire RNS-based filter.

Table 5 summarizes the resource consumption of each filter implementation. It shows that the MAC and IP FIR-based implementations have four multiplier units (DSP48E1s) with maximum frequencies of **296** and **472** MHz, respectively. By contrast, the proposed approaches are more complex than MAC. However; DAA- and RNS-based implementations has **22** and **16** memory blocks (BRAMs) used to store the pre-calculated wavelet coefficients. It also shows that the number of slice registers, slice LUTs, and occupied slices of $\boldsymbol{P_{10}}$ RNS-based is greater than one of $\boldsymbol{P_7}$ 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

Resources	**RNS-based (n = 7)**		**RNS-based (n = 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
Min. period (ns)	3.434	3.21	3.523	3.348
Estimated total power (mW)	40.5	6.59	43.08	7.33
Latency (clock cycle (CC))	6	6	6	6

Table 4. The resource use and system performance of the RNS components—that is, FCMA.

Resources	MAC	DA	FIR	RNS (n = 7)	RNS (n = 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 DSP48E1s	4	0	4	0	0
Number of RAMB18E1	0	22	0	16	16
Max. operating freq. (MHz)	296.38	229.83	472.59	258.86	261.028
Min. period (ns)	3.374	4.351	2.030	3.863	3.831
Estimated total power (mW)	8.44	66.54	9.05	56.22	53.05

Table 5. The resource use and system performance of the DWT implementation for one-level DB2 implementation.

Resources	DA-based			RNS (n = 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	132	1677
Number of RAMB18E1	22	22	22	16	32	40
Max. operating freq. (MHz)	232.72	205.55	223.31	258.87	265.32	258.80

Table 6. Resource use for the DWT implementation of DB2, DB4, and DB5.

increased by one-third, while the maximum frequency in both designs is greater than **235** MHz.

Table 6 shows a comparison between the DA- and RNS-based one-level DWT implementations when using larger filter banks—that is, DB4 and DB5. It shows that DAA-based implementation occupies a fixed number of RAMB18E1. The number of memory elements of DAA-based implementation is fixed and depends on the word length (**Table 2**).

However, as the number of filter taps increases, the memory size is exponentially increased to **2N**. By contrast, the number of memory elements that are used in RNS-based implementation is linearly increased as the number of filter taps is increased. 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.

4.2. Functionality verification

The discrete wavelet transform was simulated by means of ModelSim simulator. **Figure 10** shows that the MAC and DAA have lower latency than other approaches. It depicts that the FIR- and RNS-based of P_7 and P_{10} implementations lag behind MAC and DAA by four clock cycles.

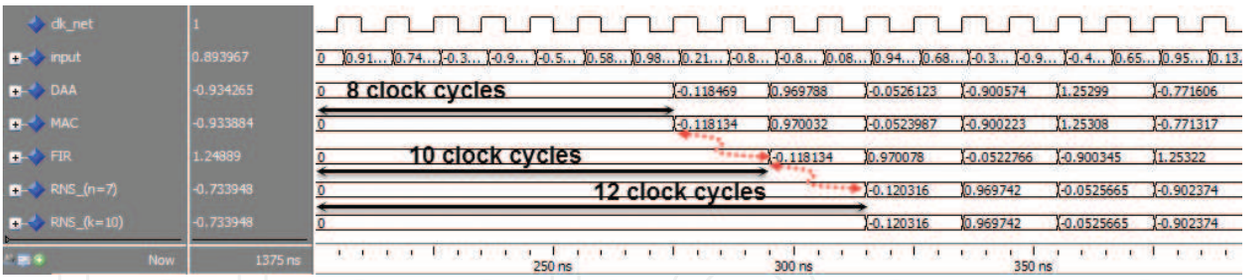

Figure 10. The output and latency of one-level DWT using a ModelSim simulator when a sin wave is applied. Each clock cycle is **10** ns.

4.3. Precision analysis

We carried out the precision analysis for the first and DWT levels, and the result is presented in **Table 7**. The output bit precision is set to $Q_{5,16}$ for all implementations. **Table 7** shows the maximum performance based on the signal-to-noise-ratio (SNR) and peak-signal-to-noise-ratio (PSNR). For P_7, we could not achieve a better accuracy with the specified scaling factors because $y + z = 19 < (3 * 7) + 1 = 22$. However, both DAA- and RNS-based approaches offer high-signal quality with a peak signal-to-noise ratio (PSNR) of **73.5** and **56.5** dB, respectively. **Figure 11** shows the effect of changing the scaling factors of P_{10} for DB2 RNS-based approach. The input scaling factor is increased from 8 to 13 bit and the filter scaling factor is increased from 11 to 18. As expected, lower scaler factors produce PSNR equal to 56 dB, whereas the maximum PSNR equal to 84 is obtained when $y = 12$ and $z = 16$.

4.4. Discussion

Hardware availability and system performance requirements are critical for selecting the appropriate architecture of the embedded platform. The number of DWT levels, filter taps, and word length has a substantial influence on the performance of the design and complexity.

Increasing the number of DWT levels has roughly the same effect on the operating frequency. Because the only change between the RNS-based with $\boldsymbol{P_7}$ or $\boldsymbol{P_{10}}$ implementations is the output signal width, and the maximum operating frequencies slightly change. Furthermore, the one-level DB2 filter bank was designed with maximum operating frequencies of **232** and **258** MHz for

Resources	FIR	MAC	DAA-based	RNS-based	
				P_7	P_7
Input precision	$Q_{5,16}$	$Q_{5,16}$	$Q_{5,16}$	$y = 8$	$y = 8$
Coefficients precision	$Q_{0,12}$	$Q_{1,15}$	$Q_{0,15}$	$z = 11$	$z = 11$
Internal word length	22 bit	22 bit	NA	22 bit	31 bit
SNR (dB)	83.2	78.7	70.4	53.41	54.78
PSNR (dB)	86.3	81.8	73.5	56.5	57.9

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

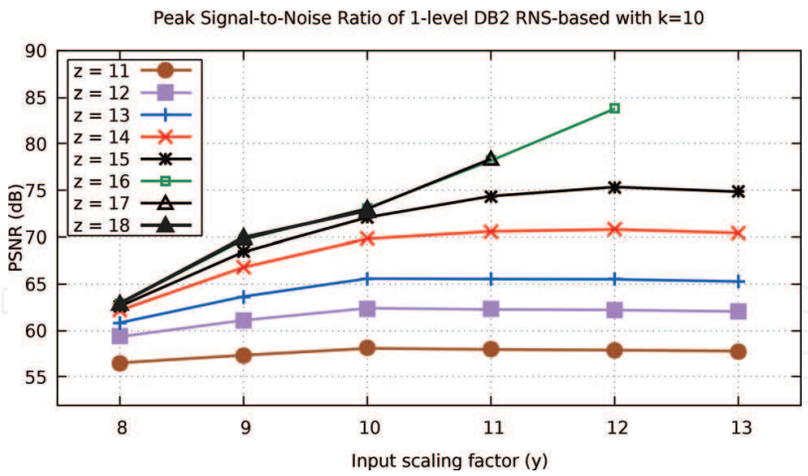

Figure 11. The impact of input and wavelet filter scaling factors of one-level RNS-based implementation with respect to P10 and P13 moduli sets on PSNR.

DAA- and RNS-based approaches, respectively. However, all high-frequency implementations introduce a latency of at least **10** clock cycles for one-level DA-based DWT.

Another critical parameter that affects the DWT performance is the filter order. 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, and the size is not affected as shown in **Table 2**. The memory requirement for DAA-based implementation is exponentially increased as the number of filter taps increases.

In addition, the two approaches have different memory content. Whereas the memory content of DAA-based implementation is consistent and identical, the memory content of RNS-based varies from tap to tap. This is obvious because each memory 590 stores the multiplication values of each filter coefficient by the moduli set.

The word length determines the number of occupied memory in both implementations. As the word length increases, the number of memory within the DAA- and RNS-based approaches increases linearly by w and $w*\mathbf{log2}(w)$, respectively. Furthermore, we could not neglect the effect of output word length on the accuracy and the internal structure. The DAA-based approach requires large memories to have high precision. By contrast, RNS-based approach could achieve high precision with adopting the scaling factors, which do not require any change to the design, except updating memory contents.

5. Conclusion

In this chapter, we addressed the effect of multiplierless DWT implementations, which have a substantial impact on the overall performance of the design and resource availability. We presented DAA- and RNS-based implementations of DWT and compared them with the

MAC-based approach. The former approaches are multiplierless architectures that intensively use memory to speed up the entire processing time.

Given implementation examples for experimental verifications and analysis, the approaches were simulated using Simulink and validated on a Xilinx Virtex 6 FPGA platform. The co-simulation results have also been verified and compared with the simulation environment. The complexity and optimization of multi-level DWT with respect to hardware structure provides a foundation for employing an appropriate algorithm for high-performance applications, such as in cognitive communication when combining the DWT analysis with machine-learning algorithms.

Author details

Husam Alzaq* and Burak Berk Üstündağ

*Address all correspondence to: alzaq@itu.edu.tr

Faculty of Computer Engineering, Department of Computer Engineering, Istanbul Technical University, Istanbul, Turkey

References

[1] Sklivanitis G, Gannon A, Batalama SN, Pados DA. Addressing next-generation wireless hallenges with commercial software-DefinedRadio platforms. IEEE Communications Magazine. 2016;**54**(1):59-67. DOI: 10.1109/MCOM.2016.7378427

[2] Mitola J, Maguire JGQ. Cognitive radio: Making software radios more personal. IEEE Personal Communications. 1999;**6**(4):13-18. DOI: 10.1109/98.788210

[3] Akyildiz IF, Lee WY, Vuran MC, Mohanty S. Next generation/dynamic Spectrum access/cognitive radio wireless networks: A survey. Computer Networks. 2006;**50**(13):2127-2159. DOI: 10.1016/j.comnet.2006.05.001

[4] Mitola J. The software radio architecture. IEEE Communications Magazine. 1995;**33**(5):26-38. DOI: 10.1109/35.393001

[5] Yang P, Li Q. Wavelet transform-based feature extraction for ultrasonic flaw signal classification. Neural Computing and Applications. 2014;**24**(3–4):817-826. DOI: 10.1007/s00521-012-1305-7

[6] 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 Transactions on Circuits and Systems I: Regular Papers. 2014;**61**(7):1984-1993. DOI: 10.1109/TCSI.20142298283

[7] Martina M, Masera G, Roch MR, Piccinini G. Result-biased distributed-arithmetic-based filter architectures for approximately computing the DWT. IEEE Transactions on Circuits and Systems I: Regular Papers. 2015;**62**(8):2103-2113. DOI: 10.1109/TCSI.2015.2437513

[8] Alzaq H, Ustundag BB. Wavelet preprocessed neural network based receiver for low SNR communication system. In: European Wireless 2015; Proceedings of 21th European Wireless Conference; 2015. pp. 1-6

[9] Carta N, Pani D, Raffo L. Biomedical Engineering Systems and Technologies: 7th International Joint Conference, BIOSTEC 2014; 3–6 March 2014; Angers; Revised Selected Papers. Cham: Springer International Publishing; 2015. pp. 66-81. DOI: 10.1007/978-3-319-26129-4_5

[10] Mallat S. A Wavelet Tour of Signal Processing. The Sparse Way. 3rd ed. Philadelphia: Academic Press; 2008

[11] Mallat SG. A theory for multiresolution signal decomposition: The wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1989;**11**(7):674-693. DOI: 10.1109/34.192463

[12] Vishwanath M. The recursive pyramid algorithm for the discrete wavelet transform. IEEE Transactions on Signal Processing. 1994;**42**(3):673-676. DOI: 10.1109/78.277863

[13] Vetterli M, Herley C. Wavelets and filter banks: Theory and design. IEEE Transactions on Signal Processing. 1992;**40**(9):2207-2232. DOI: 10.1109/78.157221

[14] Ntoune RSN, Bahoura M, Park CW. Power amplifier behavioral modeling by neural networks and their implementation on FPGA. In: 2012 IEEE Vehicular Technology Conference (VTC Fall); 2012. pp. 1-5

[15] Xilinx Inc. Virtex-6 FPGA ML605 Evaluation Kit. Available from: http://www.xilinx.com/products/boards-and-kits/ek-v6-ml605-g.html

[16] Xilinx Inc. Zynq-7000 All Programmable SoC ZC706 Evaluation Kit. Available from: https://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html

[17] Xilinx Inc. Virtex-7 FPGA VC707 Evaluation Kit. Available from: https://www.xilinx.com/products/boards-and-kits/ek-v7-vc707-g.html

[18] Chen SW, Chen YH. Hardware design and implementation of a wavelet De-noising procedure for medical signal preprocessing. Sensors. 2015;**15**(10):26396-26414. Available from: http://www.mdpi.com/1424-8220/15/10/26396

[19] 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 Transactions on Industrial Electronics. 2016;**63**(3):1923-1934

[20] Daubechies I, Sweldens W. Factoring wavelet transforms into lifting steps. Journal of Fourier Analysis and Applications. 1998;**4**(3):247-269. Available from: http://dx.doi.org/10.1007/BF02476026

[21] 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: 2015 28th International Conference on VLSI Design; 2015. pp. 327-332

[22] Madanayake A, Cintra RJ, Dimitrov V, Bayer F, Wahid KA, Kulasekera S, et al. Low-power VLSI architectures for DCT/DWT: Precision vs approximation for HD video, biomedical, and smart antenna applications. IEEE Circuits and Systems Magazine. 2015 Firstquarter; **15**(1):25-47

[23] Peled A, Liu B. A new hardware realization of digital filters. IEEE Transactions on Acoustics, Speech and Signal Processing. 1974;**22**(6):456-462

[24] Taylor FJ. Residue arithmetic a tutorial with examples. Computer. 1984;**17**(5):50-62

[25] White SA. Applications of distributed arithmetic to digital signal processing: A tutorial review. IEEE ASSP Magazine. 1989;**6**(3):4-19

[26] Jeng SS, Lin HC, Chang SM. FPGA implementation of fir filter using M-bit parallel distributed arithmetic. In: 2006 IEEE International Symposium on Circuits and Systems; 2006. pp. 4-878

[27] Meher PK, Chandrasekaran S, Amira A. FPGA realization of FIR filters by efficient and flexible Systolization using distributed arithmetic. IEEE Transactions on Signal Processing. 2008;**56**(7):3009-3017

[28] Yoo H, Anderson DV. Hardware-efficient Distributed Arithmetic Architecture for High-Order Digital Filters. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'05). Vol. 5; 2005. pp. v/125-v/128

[29] Allred DJ, Huang W, Krishnan V, Yoo H, Anderson DV. An FPGA Implementation for a High Throughput Adaptive Filter using Distributed Arithmetic. In: 12th Annual IEEE Symposium on Field-programmable custom computing machines. FCCM 2004; 2004. pp. 324-325

[30] Srividya P, Nataraj KR, Rekha KR. FPGA implementation of multiplier less matched filters to transmit video signals over satellites. In: 2014 International Conference on Communications and Signal Processing (ICCSP); 2014. pp. 602-606

[31] Pontarelli S, Cardarilli G, Re M, Salsano A. Optimized implementation of RNS FIR filters based on FPGAs. Journal of Signal Processing Systems. 2012;**67**(3):201-212. Available from: http://dx.doi.org/10.1007/s11265-010-0537-y

[32] Jenkins W, Leon B. The use of residue number systems in the design of finite impulse response digital filters. IEEE Transactions on Circuits and Systems. 1977;**24**(4):191-201

[33] Chang CH, Molahosseini AS, Zarandi AAE, Tay TF. Residue number systems: A new paradigm to Datapath optimization for low-power and high-performance digital signal processing applications. IEEE Circuits and Systems Magazine. 2015 Fourthquarter;**15**(4): 26-44

[34] Rosen KH. Elementary Number Theory and Its Applications. 5th ed. Reading: Addison-Wesley; 2004

[35] Ramírez J, Meyer-Base U, Taylor F, García A, Lloris A. Design and Implementation of High-Performance RNS Wavelet Processors Using Custom IC Technologies. Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology. 2003;**34**(3):227-237. Available from: http://dx.doi.org/10.1023/A:1023296218588

[36] Cardarilli GC, Nannarelli A, Petricca M, Re M. Characterization of RNS multiply-add units for power efficient DSP. In: 2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS); 2015. pp. 1-4

[37] Conway R, Nelson J. Improved RNS FIR filter architectures. IEEE Transactions on Circuits and Systems II: Express Briefs. 2004;**51**(1):26-28

[38] Ramírez J, García A, Meyer-Base U, Taylor F, Lloris A. Implementation of rns-based distributed arithmetic discrete wavelet transform architectures using field-programmable logic. Journal of VLSI Signal Processing Systems. 2002;**33**(1):171-120. Available from: http://dx.doi.org/10.1023/A:1021158221825

[39] Vun CH, Premkumar AB, Zhang W. A new RNS based DA approach for inner product computation. IEEE Transactions on Circuits and Systems I: Regular Papers. 2013;**60**(8): 2139-2152

[40] Daubechies I. Ten Lectures on Wavelets. Philadelphia: Society for Industrial and Applied Mathematics; 1992

[41] Mohan PVA. RNS-to-binary converter for a new three-moduli set $2n+1-1$, $2n$, $2n-1$. IEEE Transactions on Circuits and Systems II: Express Briefs. 2007;**54**(9):775-779

[42] Lin SH, hwa Sheu M, Wang CH, Kuo YC. Area-time-power efficient VLSI design for residue-to-binary converter based on moduli set ($2n$, $2n+1-1$, $2n+1$). In: IEEE Asia Pacific Conference on Circuits and systems. APCCAS 2008; 2008. pp. 168-171

[43] Reddy KS, Bajaj S, Kumar SS. Shift add approach based implementation of RNS-FIR filter using modified product encoder. In: TENCON 2014–2014 IEEE Region 10 Conference; 2014. pp. 1-6

[44] Hariri A, Navi K, Rastegar R. A New High Dynamic Range Moduli Set with Efficient Reverse Converter. Computers & Mathematics with Applications. 2008;**55**(4):660-668. Available from: http://www.sciencedirect.com/science/article/pii/S0898122107004993

[45] Xilinx Inc. System Generator for DSP. Available from: http://www.xilinx.com/products/design-tools/vivado/integration/sysgen.html

[46] Xilinx. LogiCORE IP FIR Compiler v6.3; 2011. DS795. Available from: http://www.xilinx.com/support/documentation/ip_documentation/fir_compiler/v6_3/ds795_fir_compiler.pdf