Multiplier-less 1-Level Discrete Wavelet Transform Implementations on ZC706 development kit

Husam ALZAQ, Burak Berk Üstündağ

Department of Computer Engineering, Faculty of Computer Engineering,
Istanbul Technical University, Istanbul, Turkey.
alzaq@itu.edu.tr, ustundag@itu.edu.tr

Abstract

In this paper, we investigate the design and implementation aspects of 1-level DWT by employing a finite impulse response (FIR) filter on FPGA platform. We estimate the performance requirements and hardware resources for two key multiplication-free architectures, namely, distributed arithmetic algorithm (DAA) and residue number system (RNS), allowing for selection of the proper algorithm and implementation of DAA and RNS-based DWT. The design has been implemented and synthesized in Xilinx ZYNQ-7000 FPGA, taking advantage of embedded block RAMs (BRAMs). The results show that the DAA-based approach is appropriate and feasible for a small number of filter taps, while the RNS-based approach would be more appropriate for more than 10 **filter taps.**

1. Introduction

Discrete wavelet implementation (DWT) is implemented by finite impulse response (FIR) filter. The minor problem of multiplier-accumulator (MAC) implementation is the computationally intensive multiplication. These multipliers determine the hardware complexity of the design and area usage. Moreover, if the number of multiplication units is insufficient to perform the task, the applied algorithm must be replaced with a new one that should meet the same requirements. In this case, the multiplier units can be replaced by new algorithm that internally performs the multiplication by using different units such as memory and adders.

This work presents 1-D DWT implementations on Xilinx ZYNQ ZC706 by means of memory-based approaches. We compare various implementations in terms of system performance and resource consumption. Because the fundamental niche 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.

The remainder of this paper is organized as follows. In section 2, the related works are briefly reviewed. In Section 3, the theoretical background on DAA and RNS is given. Section 4 illustrates the implementation of discrete wavelet transform. Section 5 presents the performance results. Finally, conclusions are drawn in Section 6.

2. Related work

Implementations of 1-D DWT for signal de-noising, feature extraction and pattern recognition can be found in [1, 2]. 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 DWT, introduced by Sweldens [3]. The key advantage of using convolution-based DWT over lifting approach is that they do not require temporary registers to store the intermediate results and with appropriate design strategy they could have better area and power efficiency [4].

Rather than the simplest implementation of FIR filter via multipliers and an adder tree, a multiplier-free architecture is suggested because they result in low complexity systems [4]. 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 [5, 6]. The LUT, a memory element, stores precomputed partial results [7]. Several techniques have been proposed to improve the design, such as the partial sum technique [7]. Alternatively, RNS is a highly parallel non-weighted arithmetic system that is based on the residue of division operation of integers using the LUT scheme [4, 8]. Eventually, the RNS-based results are converted back to the equivalent binary number format using a Chinese reminder theorem (CRT) [9]. The key advantage of RNS is gained by reducing an arithmetic operation to a set of concurrent, but simple, operations.

In this paper, a 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 and memory consumptions. To the best of our knowledge, no detailed comparisons of hardware implementations of the three major 1-D DWT design exist in the literature. This comparison will give a significant insight on which implementation is the most suitable for given values of the relevant algorithmic parameters.

3. Background

This section provides background information and preliminaries on the discrete wavelet transform, distributed arithmetic algorithm and the residue number system that are directly relevant and useful for designing and implementing the proposed architecture. A brief description of these methods is as follows.

3.1. Discrete Wavelet Transform

The wavelet decomposition mainly depends on the orthonormal filter banks. Fig. 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, DWT filter creates a series of coefficients that represents and compacts the

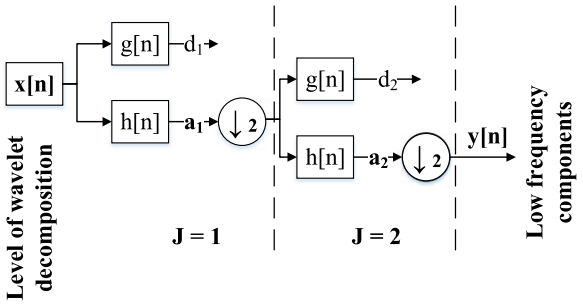

Figure 1. Multi-resolution wavelet decomposition. The block diagram of two-channel two-level DWT decomposition ($J = 2$) that decomposes a discrete signal into two parts. Note that $\downarrow 2$ is keeping one sample out of two, a_i and d_i are the approximation and details at level i, respectively.

original signal information.

Mathematically, a signal $y[n]$ consists of high and low frequency components, as shown in equation (1).

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

The decimated low-pass filtered output is recursively passed through identical filter banks to add the dimension of varying resolution at every stage. Equations (2) and (3) mathematically express the filtering process of a signal through a digital high-pass filter $g[k]$, and low-pass filter $h[k]$. This operation corresponds to a convolution with an impulse response of $k-$tap filters.

$$y_{high}[n] = \sum_k g[k].x[2n-k] \tag{2}$$

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

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

3.2. Distributed Arithmetic Algorithm

Equation (3) shows that output y is the sum of multiplication of the filter coefficients and the input. The distributed arithmetic algorithm (DAA) eliminates the need for hardware multipliers by performing the arithmetic operations in a bit serial computation [5, 7]. Because the down sampling process follows each filter (as shown in Fig. 1), equation (3) can be rewritten without the decimation factor as:

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

where N is the filter tap. For the sake of simplicity of representing equation (4), $x[n-k]$ is replaced by $x[k]$. Hence, equation (4) can be expressed as:

$$y_{low}[n] = \sum_{k=0}^{N-1} x[k].h[k] \tag{5}$$

For DB2 wavelet, where the number of tap is 4, equation (5) can be expanded as

$$y_{low}[n] = h[0].x[n]+h[1].x[n-1]+h[2].x[n-2]+h[3].x[n-3] \tag{6}$$

Further simplification can be performed on the $x[k]$, equation (4). Considering the representation of $x[k]$ as a fixed point arithmetic with length L, $x[k]$ becomes

$$x[k] = -x[k]_0 + \sum_{l=1}^{L-1} x[k]_l.2^{-l} \tag{7}$$

where $x[k]_l$ is the l^{th} bit of $x[k]$ and $x[k]_0$ is the sign bit. Substituting equation (7) into (4), the output of the filter becomes

$$y[n] = \sum_{k=0}^{N-1} h[k].(-x[k]_0 + \sum_{l=1}^{L-1} x[k]_l.2^{-l}) \tag{8}$$

By interchanging the order of summations, equation (8) can be written as

$$y[n] = \Big[\sum_{l=1}^{L-1} 2^{-l}.\sum_{k=0}^{N-1} h[k].x[k]_l\Big] + \sum_{k=0}^{N-1} h[k](-x[k]_0) \tag{9}$$

Considering $x[k]_l$ which takes a 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 can be pre-computed and stored in a ROM, indexed by $x[k]_l$.

3.3. Residue Number System

The RNS [10, 11] 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 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 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 RNS by its residues $|X|_{m_i}$, which is the remainder of division X by m_i and M is defined in equation (10),

$$M = \Pi_{i=1}^{q} m_i = m_1 * m_2 * \cdots * m_q \tag{10}$$

M determines the range of unsigned numbers in $[0, M-1]$, and should be greater than the largest performed results.

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. We assume that this interval is sufficient to map the input values, which is not exceeded ± 2. Third, the reverse converter unit is simple and regular [12] because it does not employ any ROM.

4. DWT Implementation Methodology

4.1. DWT implementation using DAA

DAA hides the explicit multiplications with an ROM lookup table, which 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 $22-$bit width, with 16 binary-point bits ($Q_{5,16}$).

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

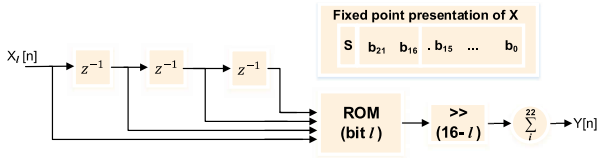

Figure 2. The block diagram of DAA-based architecture of the DB2. For simplicity, we showed one ROM and one shift register. In the actual design, there are 22 ROMs and shift registers. $>>$ is a $16-l$ shift operation, where 16 is the number of the binary point.

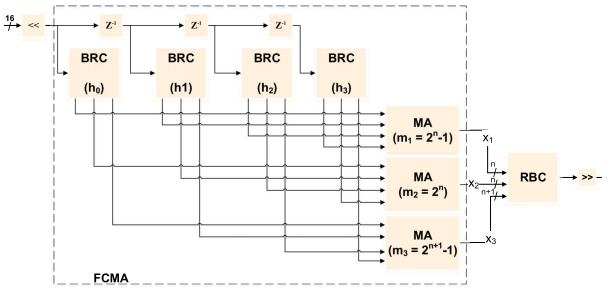

Figure 3. The block diagram of DB2 RNS-based architecture. BRC is an abbreviation for binary-to-residue converter, RBC for residue-to-binary converter and MA for modulo adder.

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

4.2. DWT implementation using RNS

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

4.2.1. The forward converter

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

4.2.2. RNS-system number conversion

The received sample, $X[i]$, is scaled up by shifting y positions to the left (multiplying by 2^y). This ensures that $X[i]$ is a $y-$bit fixed point integer. In a similar manner, the wavelet coefficients are scaled by shifting it z positions to the left. In our design, we set the filter scaling factor z to 11. As a result, the low-pass filter of DB2 is multiplied by 2^{11} and rewritten as,

$$y_{low}[n] = -266x[n]+459x[n-1]+1713x[n-2]+989x[n-3] \quad (11)$$

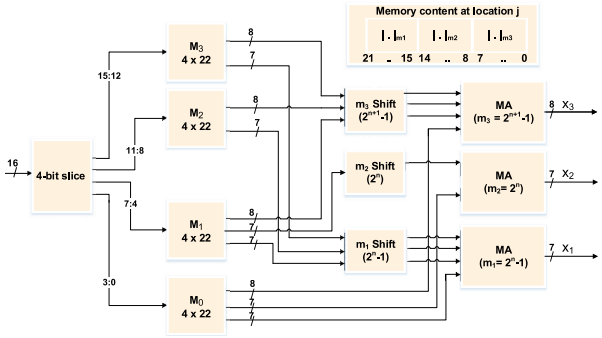

Figure 4. The block diagram of the binary-to-residue converter for the 3-channel RNS-based DWT, $P_7 = \{127, 128, 255\}$. Four identical memories are used at each tap. The upper corner shows the memory content at location $j \in [0, 15]$.

4.2.3. Modulo m_i multiplier

The multiplication of the received sample, $X[i]$, by the filter coefficients, is performed via indexing the ROM. As the word-length, w of the received sample $X[i]$ is increased, the memory size becomes 2^w. To improve the design, we suggest to preserve one ROM that contains all module results. In this way, each word at location j contains the q modules of $h_k * j * 2^{11}$. Fig. 4 shows the internal BRC block design of the 3-channel moduli set $P_7 = \{127, 128, 255\}$ with its memory-map at 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 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 equation (12). The advantage of this method is that no extra hardware is required to separate each module value.

$$\begin{aligned} ROM(j) = &|h_k * j * 2^{11}|_{m_1} * 2^{2n+1} + \\ &|h_k * j * 2^{11}|_{m_2} * 2^{n+1} + \\ &|h_k * j * 2^{11}|_{m_3}, \quad j = [0, 2^w] \end{aligned} \quad (12)$$

As 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, each consisting of $4 \times 22-$bits. Fig. 4 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 is correct. According to the previous improvements, the RNS-based works 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 $|h_k * x_l * 2^{11}|_{m_i}$, 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 [8].

4.2.4. 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 types of MA — i.e., 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. Modulo $2^n - 1$ adder differs from modulo 2^n adder in that the carry should be considered to limit the result to not greater than $2^n - 1$, as in equation (13).

$$|x+y|_{2^n-1} = \begin{cases} x+y & \text{if } x+y \leqslant 2^n-1, \\ x+y+1 & \text{otherwise} \end{cases} \quad (13)$$

4.2.5. The reverse converter

Mapping from the RNS system to integers, $\mathbb{Z}$, is performed by the Chinese reminder theorem (CRT) [9, 12]. The direct implementation of CRT is inefficient because it requires a divider unit and several multipliers to determine the final output. However, 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 [12].

5. Performance Analysis and Validation

In this section, we show the performance of the two approaches, in addition to the synthesis results. Hardware analysis was performed 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 co-simulation design has been synthesized and implemented on Xilinx Zynq-7000 All Programmable SoC ZC706 Evaluation Kit [13].

The implementation of 1-level RNS and DAA is compared with the multiplier-accumulate based DWT structure (MAC) and by the one that use an IP FIR Compiler 6.3 (FIR6.3) block, which provides a common interface to generate highly parameterizable, area-efficient, high-performance FIR filters [14].

For RNS implementation, the moduli sets of $P_7 = \{127, 128, 255\}$ and $P_{10} = \{1023, 1024, 2047\}$ were used. In all implementations, the word-length was set to 16 bits.

5.1. Resource utilization and system performance

Table 1 summarizes the resource consumption of each filter implementation. It shows that the MAC and IP FIR-based implementations have 4 multiplier units (DSP48E1s) and maximum frequencies of 296 and 472 MHz, respectively. In contrast, DAA- and RNS-based implementations occupy 22 and 16 memory blocks (BRAMs), where BRAM is an optimized lookup tables (LUTs) used to store the pre-calculated wavelet coefficients.

Table 1 also shows that the number of slice registers, slice LUTs and occupied slices of P_{10} RNS-based is greater than one of P_7 because the former is 31 bits (bus width), while the later is 22 bits. As a result, the number of flip-flops is increased and the number of resources is approximately increased by one third. However, the maximum frequency in both designs is greater than 200 MHz.

5.2. Functionality Verification

The 1-level DB2 discrete wavelet transform was simulated by a ModelSim simulator. Fig. 5 shows that the MAC and DAA have lower latency than other approaches.

Eventually, we have verified the simulated result on ZC706 kit. The simulation and hardware co-simulation results of the 1-level DB2 implementations are highly correlated, as shown in Fig. 6. It also shows that there is a stroke at the beginning of each RNS-based DWT implementations considering the initialization of the internal registers is zero, and this value is propagated to influence the first two values.

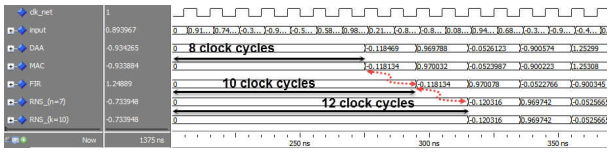

Figure 5. The output and latency of 1-level DWT using a ModelSim simulator when a sin wave is applied. Each clock cycle is 10 ns.

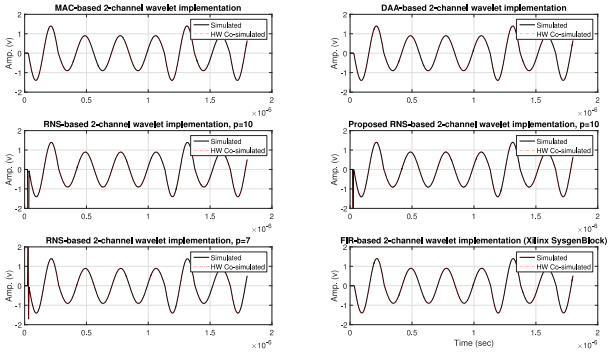

Figure 6. The output of 1-level DWT when a pattern-based signal [1] is applied. It is used to extract the main features of the received signal.

5.3. Discussion

Previous results reveal that the DWT performance is effected by several factors, including number of filter taps and word-width. Generally, DAA-based implementation outperforms the RNS-based in terms of number of memory for small number of filter taps. As the number of taps increases, the number of memory keeps constant and its size is proportional to number of taps (2^N). On the other hand, RNS-based approach is appropriate for large number of filter taps because the size does not change as the number of taps increases.

Moreover, the two approaches differ in their memory content. While 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 stores the multiplication values of each filter coefficient by the moduli set, as shown in equation (12).

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 $w * \lceil \log_2(w) \rceil$, respectively. In addition, we could not omit the impact of output word-length on the accuracy and the internal structure of DAA-based approach.

6. Conclusion

In this work, we have studied the effect of multiplier-less architectures DWT, which has a substantial influence on the overall performance of the design and resource availability. Moreover, we presented pipelining DAA- and RNS-based implemen-

Table 1. The resource use and system performance of the implementation of DWT with Xilinx ZYNQ ZC706 kit for 1-level DB2 implementation.

	FIR	DAA	MAC	RNS($n = 7$)	RNS($n = 10$)
Slice LUTs (218600)	50	492	72	905	1335
Slice Registers (437200)	176	623	250	589	795
Slice (54650)	41	159	52	319	419
LUT Flip Flop Pairs (218600)	126	580	194	1037	1452
Block RAM Tile (1090)	0	22	0	16	16
DSPs DSP48E1 (900)	4	0	4	0	0
Max. Operating Freq. (MHz)	680,76	375,94	290,61	216,54	206,78
Min. Period (ns)	8,531	6,559	7,340	5,382	5,164
Data Path Delay (ns)	1,208	3,402	2,632	4,296	4,573
Total On-Chip Power (W)	0,245	0,288	0.246	0,302	0,315
On-Chip Components (mW)	7	51	9	64	76
Block RAM (mW)	0	36	0	32	32

tations of discrete wavelet transform and compared them with the pipelining MAC-based approach. These approaches intensively employs memory (LUT) to speed up the entire processing time. The trade-off between system performances and resource consumption was also addressed. 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 is greater than 10. Future work will focus on optimizing the FPGA resource utilization for RNS-based approach.

7. References

[1] H. Alzaq and B. Ustundag, "Wavelet Preprocessed Neural Network Based Receiver for Low SNR Communication System," in *European Wireless 2015; 21th European Wireless Conference; Proceedings of*, May 2015, pp. 1–6.

[2] F. Duan, L. Dai, W. Chang, Z. Chen, C. Zhu, and W. Li, "sEMG-Based Identification of Hand Motion Commands Using Wavelet Neural Network Combined With Discrete Wavelet Transform," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 3, pp. 1923–1934, March 2016.

[3] I. Daubechies and W. Sweldens, "Factoring Wavelet Transforms into Lifting Steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, pp. 247–269, 1998. [Online]. Available: http://dx.doi.org/10.1007/BF02476026

[4] P. K. Meher, B. K. Mohanty, and M. M. S. Swamy, "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*, Jan 2015, pp. 327–332.

[5] A. Peled and B. Liu, "A New Hardware Realization of Digital Filters," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 22, no. 6, pp. 456–462, Dec 1974.

[6] F. Taylor, "Residue Arithmetic A Tutorial with Examples," *Computer*, vol. 17, no. 5, pp. 50–62, May 1984.

[7] S. White, "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review," *ASSP Magazine, IEEE*, vol. 6, no. 3, pp. 4–19, July 1989.

[8] K. Reddy, S. Bajaj, and S. Kumar, "Shift Add Approach Based Implementation of RNS-FIR Filter Using Modified Product Encoder," in *TENCON 2014 - 2014 IEEE Region 10 Conference*, Oct 2014, pp. 1–6.

[9] K. H. Rosen, *Elementary Number Theory and its Applications*, 5th ed. Reading, MA: Addison-Wesley, 2004.

[10] W. Jenkins and B. Leon, "The use of Residue Number Systems in the Design of Finite Impulse Response Digital Filters," *Circuits and Systems, IEEE Transactions on*, vol. 24, no. 4, pp. 191–201, Apr 1977.

[11] C. H. Chang, A. S. Molahosseini, A. A. E. Zarandi, and T. F. Tay, "Residue Number Systems: A New Paradigm to Datapath Optimization for Low-Power and High-Performance Digital Signal Processing Applications," *IEEE Circuits and Systems Magazine*, vol. 15, no. 4, pp. 26–44, Fourthquarter 2015.

[12] S.-H. Lin, M. hwa Sheu, C.-H. Wang, and Y.-C. Kuo, "Area-Time-Power Efficient VLSI Design for Residue-to-binary Converter Based on Moduli Set $(2^n, 2^{n+1}-1, 2^n+1)$," in *Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on*, Nov 2008, pp. 168–171.

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

[14] Xilinx, "LogiCORE IP FIR Compiler v6.3," Product Specification DS795, Oct 2011.