Performance Analysis of FIR Digital Filter Design: RNS Versus Traditional

Shahana T. K. [*], Rekha K. James[*], Babita R. Jose[*], K. Poulose Jacob[*] and Sreela Sasi[†]
[*] Cochin University of Science and Technology
Kochi, Kerala, India
Tel: +91-484-2576368, E-mail: {shahanatk, rekhajames, babitajose, kpj}@cusat.ac.in
[†] Gannon University, Erie, Pennsylvania, USA
Tel: +(814) 871-5635, E-mail: sasi001@gannon.edu

***Abstract*— This paper presents performance analysis of Residue Number System (RNS) based Finite Impulse Response (FIR) digital filters and traditional FIR filters. This research is motivated by the importance of an efficient filter implementation for digital signal processing. The comparison is done in terms of speed and area requirement for various filter specifications. RNS based FIR filters operate more than three times faster and consumes only about 60% of the area than traditional filter when number of filter taps is more than 32. The area for RNS filter is increasing at a lesser rate than that for traditional resulting in low-power consumption. RNS is a nonweighted number system without carry propagation between different residue digits. This enables simultaneous parallel processing on all the digits resulting in high speed addition and multiplication in RNS domain. Such compact and high speed real-time digital filters find applications in radar, communications and image processing systems.**

I. INTRODUCTION

Finite Impulse Response (FIR) digital filters have attracted a great deal of interest because they are inherently stable structures that are much less sensitive to quantization errors than filters of the recursive type. The major disadvantage of an FIR design is that usually a large number of nonzero terms are required in the impulse response in order to adequately control the frequency response of the filter. This results in large number of multiplications and additions that must be executed during a short sample interval which in turn will seriously limit the speed of the filter. An FIR filter is described by equation (1), where X(n) is the input to the filter, H(k) represents the filter coefficients, N is the order of the filter and Y(n) is the output from the filter.

$$Y(n) = \sum_{k=0}^{N} H(k)X(n-k) \qquad (1)$$

For very large N, filters implemented in the traditional binary weighted number system suffer from the disadvantages of the carry propagation delay in binary adders and multipliers. In RNS a large integer is broken into smaller residues which are independent of each other, and each digit is processed in parallel channels without any carry propagation from one to another [1]. This leads to significant speed up of multiply and accumulate (MAC) operations which in turn results in high data rate for RNS based FIR filters.

Several researchers have proposed various applications of RNS for Digital Signal Processing. RNS based digital signal processors are reported in [2] and [3]. A number of FIR filters based on residue arithmetic are available in the literature. The implementation of high speed recursive digital filters using residue arithmetic properties and high speed multiplication by a fraction using a table look up is given in [4]. Another technique is presented in [5] for implementing a FIR digital filter in residue number system with a modified hardware implementation of the Chinese Remainder Theorem for translation of residue coded outputs into natural numbers. The feasibility of combining multiple valued logic (MVL) with RNS arithmetic is explored in [6], and developed a very detailed block diagram representation of an MVL-RNS digital filter independent of the choice of levels in MVL or moduli in RNS. Structurally passive digital filters are realized in [7] using only RNS based rotators and delays. Low power realization of RNS based FIR filter is proposed in [8] with coefficient ordering and coefficient encoding techniques. A polyphase filter bank in the Quadratic Residue Number System (QRNS) has been implemented in [9] and compared in terms of area and power dissipation to the implementation of a polyphase filter bank in the traditional two's complement system. Implementation of RNS FIR filters with low static and dynamic power dissipation using standard cell libraries with dual threshold transistors is presented in [10]. Most of the work in the current literature addresses only the implementation in RNS domain and none makes a fair comparison of RNS based FIR filter with a traditional implementation of FIR filter.

This paper presents hardware implementations and graphical analysis of speed and area for FIR digital filters implemented in RNS versus traditional binary number system. The organization of the paper is as follows. First, an overview of RNS basics is provided in Section II. In Section III an illustration of traditional and RNS based filter architecture is given. The simulation results for the hardware implementation of both types of filters along with a

comparison of performance are presented in Section IV. The hardware implementation using LEONARDO SPECTRUM logic synthesis tool for a particular RNS FIR filter is also presented in this section. Conclusion is given in Section V.

II. RNS BASICS

RNS is defined by a set of 'r' relatively prime integers (m_1, m_2,…, m_r) which are called the moduli. Any integer 'X' in the interval of [0, M) can be represented as a set of 'r' residues (x_1, x_2,…, x_r), where $x_i = X \bmod m_i$ and $M = \prod_{i=1}^{r} m_i$. 'M' is defined as the range of the number system. Negative numbers can be represented by partitioning the dynamic range into two sets. A common assignment is that all numbers X, $0 \le X \le (M/2)-1$ are considered positive and the rest are considered as negative. In RNS, arithmetic operations are computed by the formula: $(x_1, x_2, ..., x_r) \otimes (y_1, y_2, ..., y_r) = (z_1, z_2, ..., z_r)$ where $z_i = |x_i \otimes y_i|_{m_i}$ and $\otimes$ denotes one of the operations of addition, subtraction or multiplication. Thus arithmetic operations on residues can be performed in parallel without any carry propagation among the residue digits. However, before performing any operations on residues the number has to be converted from binary to residue by performing modulo operations with respect to each modulus in the moduli set. The process of translating a binary integer X, in the range 0 to M-1 (both inclusive), to the residue representation (x_1, x_2,…, x_r) with respect to a relatively prime moduli set (m_1, m_2,…, mr) is called forward conversion. Getting back to the weighted representation of 'X' from a given residue representation is referred to as reverse conversion.

The reverse conversion can be done using Mixed Radix Conversion (MRC) and Chinese Remainder Theorem (CRT). MRC is usually used for sign determination, magnitude comparison and overflow detection while CRT is more adapted for generation of binary number directly from its residue. CRT is based on the general formula:

$$X = \left| \sum_{i=1}^{r} a_i x_i \hat{M}_i \right|_M, \text{ where } \hat{M}_i = \frac{M}{m_i} \text{ and } a_i = \left| \frac{1}{\hat{M}_i} \right|_{m_i}.$$

The CRT implementation presented in [11] uses 'r' ROMs to store the precomputed values $\left| a_i x_i \hat{M}_i \right|_M$ each indexed by x_i. This reduces CRT implementation to a summation of 'r' values, followed by modulo correction with 'M' using carry save adder stages and a final carry propagate adder. The overall hardware including the size of ROM stage used in the design is further reduced in [12] by selecting one of the moduli of the form 2^n, so that the least significant 'n' bits of the binary number are directly available.

The set of moduli chosen for RNS affects both the representational efficiency and the complexity of arithmetic algorithms. Since the magnitude of the largest modulus decides the speed of arithmetic operations, all the remaining moduli can be chosen so that they are comparable with the largest one. The optimal choice is dependent on both the application and the target implementation technology.

III. FIR FILTER ARCHITECTURE

The architecture of a FIR filter based on traditional binary number system is shown in Fig. 1. As the input samples are multiplied by the filter coefficient, the advantage of multiplication by a constant is considered while designing the multipliers. In the proposed work Carry Save Adder (CSA) tree based multiplier is used for fast multiplications and Carry Propagate Adder (CPA) is used for addition.

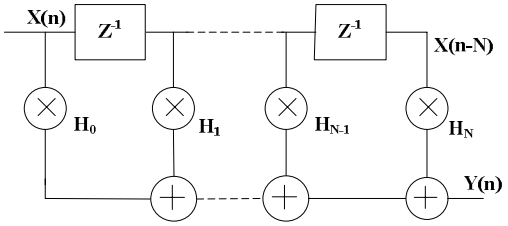

Fig. 1. Traditional FIR filter architecture

For RNS based FIR filter, the first step is to choose appropriate moduli set that provides sufficient dynamic range for the filter. Let the moduli set be (m_1, m_2,…, mr). Then there will be 'r' parallel filter channels as in Fig. 2, which process the signals from binary to RNS converter. Finally, the RNS to binary converter combines the signals from all the channels and puts the output signal back in binary form. A single channel corresponding to modulus 'm_i' of such a filter is shown in Fig. 3, where $\otimes$ and $\oplus$ represent modulo multiplication and modulo addition respectively. The filter coefficients are directly represented in residue form. Modulo multiplication is implemented with look up table (LUT) for faster multiplication, and the size of the LUT is small as the operation is in the residue domain with one of the operands constant. The modulo adder receives the residue digits and performs usual binary addition, followed by modulo correction if a carry out is produced.

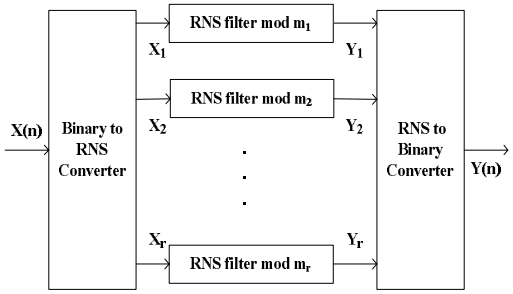

Fig. 2. RNS implementation of FIR filter

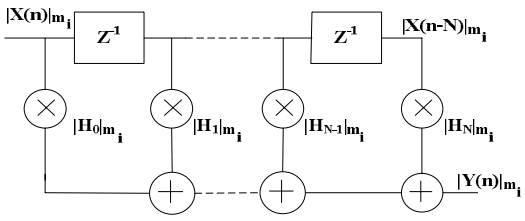

Fig. 3. i^{th} modulo filter channel

IV. HARDWARE IMPLEMENTATION

To compare the filtering operation of traditional and RNS based filter, both are simulated with 64 taps and a passband frequency of 500 Hz and a stopband frequency of 1400 Hz respectively. The frequency response of the filters is shown in Fig. 4. The original signal consists of 500 Hz message signal along with 3000 Hz noise. The filtered outputs with the two types of filters are shown in Fig. 5 and 6. The noise attenuation is exactly the same in both cases.

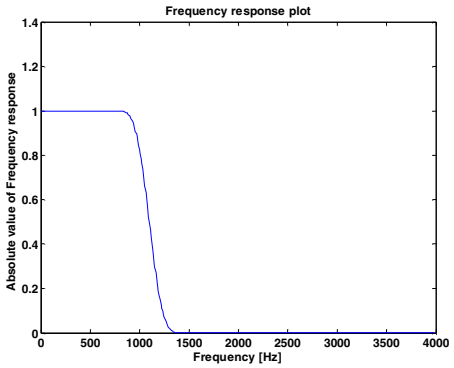

Fig. 4. Frequency response of FIR filter

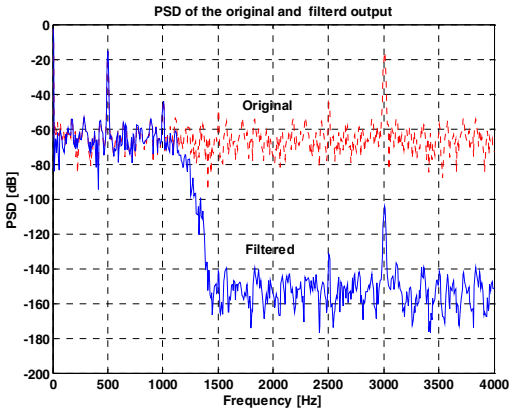

Fig. 5. Power spectral density of original and filtered output with Traditional filter

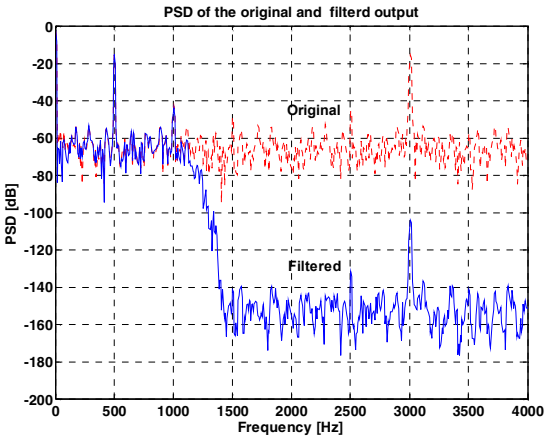

Fig. 6. Power spectral density of original and filtered output with RNS filter

Hardware analysis is done with the logic synthesis tool *Leonardo Spectrum* from Mentor Graphics Corporation, using ASIC library. For traditional filter carry propagate adders are used whose size increases progressively with each additional filter tap. So, the total number of full adders used for implementing the filter is in the form of an arithmetic progression. As the multiplier is multiplying the input sample by the filter coefficient, one of the operands of the multiplier is constant for a particular filter structure. Accordingly a reduced hardware is designed using CSA tree for faster multiplication.

The RNS filter architecture includes a forward converter, modulo adders and modulo multipliers for each parallel channel, and a reverse converter. In the forward converter binary number is split up into various fields, and the corresponding residue values are selected using vector multiplexers. Then these are combined with modulo adder trees to get the final residue for each modulus. As the operands are smaller size residues modulo multiplier based on LUT is used, and as one operand is a constant the size of the LUT is small. Modulo addition and multiplication produces fixed width output so that for each additional filter tap the rate at which the hardware increases is less compared to the traditional filter.

The graphs shown in Fig. 7 and Fig. 8 give a comparison of critical path delay and area respectively for a traditional and RNS filter compared to that of a full adder. Fig. 9 and Fig. 10 demonstrate the speed up factor and area requirement for RNS filter compared to traditional filter as the number of filter taps increases, assuming 6-bit binary input. The graphs indicate that RNS filter becomes more than three times faster, and requires only 60% or less area than the corresponding traditional filter implementation as the number of filter taps increases above 32.

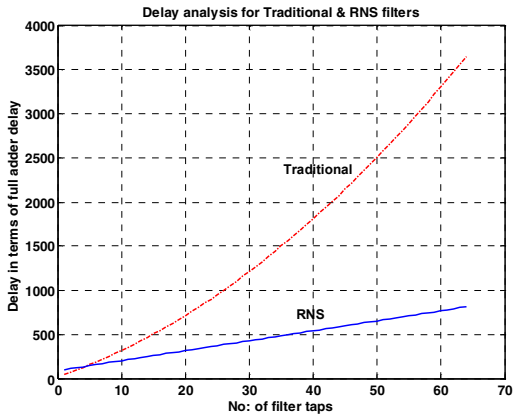

Fig. 7. Critical Path delay: Traditional vs. RNS implementation

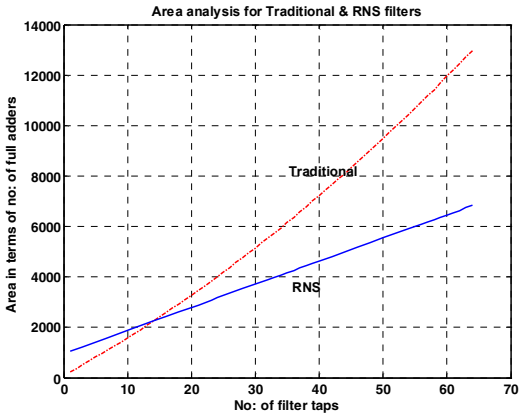

Fig. 8. Area: Traditional vs. RNS implementation

Fig. 9. Speed up factor for RNS filter vs. traditional filter

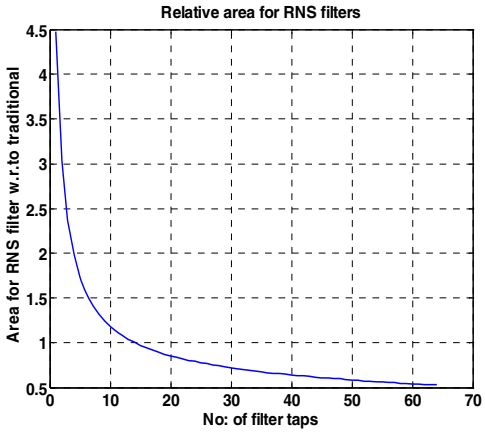

Fig. 10. Percentage area improvement of RNS filter vs. traditional

The synthesized results obtained using Leonardo Spectrum logic synthesis tool for a 64 taps RNS FIR filter with an input word length of 6-bits, and passband and stopband frequencies of 800Hz and 1400Hz, is shown in Table 1. The critical path delay and area for each block of the filter is normalized with respect to a full adder critical path delay of 1.98 ns and area of 38μm^2. The moduli set are selected as (23, 25, 27, 29, 31, 32) which provide 29 bit dynamic range with 446623200 unique integer representations. So, the range of negative and positive numbers represented by the selected RNS is from -223311600 to +223311599. The filter has six parallel channels processing 5-bit residues corresponding to each modulus. The overall critical path delay for the filter is 813.28 full adder delays, and the area requirement is 6825.81 times the full adder area. The percentage of area required by each block of the filter is also provided in Table I.

TABLE I. CRITICAL PATH DELAY AND AREA FOR 64 TAPS RNS FIR FILTER

Name of the block	Critical path delay (Normalized w.r.to FA)	Area (Normalized w.r.to FA)	Percentage area of each block (%)
Forward converter	13.28	603.42	8.84
LUTs for modulo multipliers	1.25	717.6	10.51
Modulo adders	724.48	5160.96	75.6
Reverse converter	74.27	343.83	5.04
Total	813.28	6825.81	100

The forward converter consists of vector multiplexers to select the residues for each field of the binary number which are then combined using modulo adder tree structure. As the largest value in the selected moduli set is 32, the least significant 5 bits of the binary number directly gives the residue value for the modulus 32. To produce the remaining residue digits for the other 5 channels the area consumed is 603.42 times the full adder area. A reverse converter

proposed in [12] is implemented with 6 ROMs and three levels CSA tree. A 5-bit binary adder and a LUT are used for converting the most significant field from CSA tree. The output is combined with the remaining output field of the CSA tree using another CSA and CPAs for modulo correction. Finally, a vector multiplexer is used to select the required 24-bit output from the CPAs. This is concatenated with the 5-bits of modulo 32 residue to form the final 29-bit binary number. Then the number is converted to a sign magnitude representation. The synthesized results using Leonardo Spectrum logic synthesis tool for delay and area required in each block of this reverse converter is shown in Table II.

TABLE II. CRITICAL PATH DELAY AND AREA FOR REVERSE CONVERTER

Name of the block	Critical path delay (Normalized w.r.to FA)	Area (Normalized w.r.to FA)	Percentage area of each block (%)
ROMs (6 Nos)	6.6	63.16	18.37
CSA tree	3	99	28.79
5 bit CPA and LUT	6.75	10.94	3.18
CSA	1	24	6.98
CSA and two CPAs	26	74	21.52
Vector multiplexer	0.69	15.73	4.58
Sign Magnitude converter	30.23	57	16.58
Total	74.27	343.83	100

The critical path delay and area required for a traditional filter having the same specifications using 17-bits wide filter coefficients and an input word length of 6-bits are shown in Table III. As the filter coefficients and input samples are signed numbers, signed addition and multiplication are considered. Traditional filter implementation has a critical path delay of 3650.72 times the full adder delay and area of 12983.6 times full adder area. This implies the RNS filter operates 4 times faster and requires only half the area than the corresponding traditional implementation.

TABLE III. CRITICAL PATH DELAY AND AREA FOR 64 TAPS TRADITIONAL FILTER

Name of the block	Critical path delay (Normalized w.r.to FA)	Area (Normalized w.r.to FA)	Percentage area of each block (%)
CSA tree based multipliers	20	6110	47.06
Ripple carry adders (signed addition)	3630.72	6873.6	52.94
Total	3650.72	12983.6	100

V. CONCLUSION

The hardware implementation and performance comparison of FIR digital filters implemented in RNS versus traditional binary number system are presented. The RNS coding technique is attractive for FIR filters as it requires only multiplication and addition which are very fast operations in residue domain. While designing RNS filter care must be taken to choose the moduli set depending on the filter length and input word length so as to provide sufficient dynamic range avoiding overflow. The area overhead due to forward and reverse conversion seems to be compensated after a particular filter length as the rate of increase of area for RNS filter, for each additional filter tap, is less than that for traditional filter. The speed and delay comparison graphs shows that RNS filter is more than 3 times faster and requires only less than 60 % of area than that for the corresponding traditional filter, when the filter length is increased above 32 taps for an input word length of 6-bits.

REFERENCES

[1] Soderstrand M.A., Jenkins W.K., Jullien G.A., and Taylor F.J., *Residue number system arithmetic: modern applications in digital signal processing*, (IEEE Press, New York, 1986).

[2] Di Claudio E., Piazza F., and Orlandi G., "Fast combinatorial RNS processors for DSP applications", *IEEE Trans. Comput.*, 1995, Vol. 44, pp. 624–633.

[3] Ramirez, J., Garcia A., Lopez Buedo S., and Lloris A., "RNS-enabled digital signal processor design", *Electron. Lett*. 2002, Vol.38, pp. 266–268.

[4] Michael A. Soderstrand, "A high-speed low-cost recursive digital filter using residue number arithmetic", Proceedings of the *IEEE Electron. Lett*., 1977, pp. 1065-67.

[5] W. K. Jenkins and B. J. Leon, "The use of residue number systems in the design of finite impulse response digital filters", *IEEE Trans. on Circuits and Systems*, Vol. 24, No. 4, April 1977, pp. 191-201.

[6] M. A. Soderstrand and R. A. Escott, "VLSI implementation in multiple-valued logic of an FIR digital filter using residue number system arithmetic", *IEEE Trans. on Circuits and Sys*., Vol. 33, No. 1, January 1986, pp. 5-25.

[7] G.C. Cardarilli, R. Lojacono, G. Martinelli and M. Salerno, "Structurally passive digital filters in residue number system", *IEEE Trans. Circuits and Systems*, Vol. 35, No. 2, February 1988, pp.149-158.

[8] M. N. Mahesh, M. Mehendale, "Low power realization of residue number system based FIR filters", *Proceedings of the 13th International Conference on VLSI Design*, 2000.

[9] G. C. Cardarilli, A. Del Re, A. Nannarelli, and M. Re, "Low-power implementation of polyphase filters in quadratic residue number system", *Proceedings of the 2004 IEEE International Symposium on Circuits and Systems*, Vol. 2, 23-26 May 2004, pp. II - 725-8 .

[10] G.C. Cardarilli, A. Del Re, A. Nannarelli and M. Re, "Low power and low leakage implementation of RNS FIR filters", *Proceedings of 39th Asilomar Conference on Signals, Systems, and Computers*, October 2005, pp. 1620-1624.

[11] B. Parhami and C.Y. Huang, "Optimal look up schemes for VLSI implementation of input/output conversions and other residue number operations," in VLSI Signal Processing VII, J. Rabaey, P. M. Chau and Eldon, eds., IEEE Press, New York, 1994.

[12] D. Radhakrishnan, T. Srikanthan and J. Mathew, "Using the 2^n property to implement an efficient general purpose residue-to-binary converter", *Proceedings SCS '99*, Iasi, Romania, July 1999, pp. 183-186.