Efficient Modulo Adder Architectures

Mohammed Mosin A. Junjwadkar and Virapax M. Chougala

Abstract--- In this manuscript we introduce parallel-prefix architecture for the design ofmodulo2^n+1 adder. The proposed architecture is built around a sparse carry computation unit that computes only some of the carries of the modulo 2^n+1addition.This sparse approach is enabled by the introduction of the inverted circular idem potency property of the parallel-prefix carry operator and its regularity and area efficiency are further enhanced by the introduction of a new prefix operator. The resulting diminished-1 adders can be implemented in smaller area and consume less power compared to all earlier proposals, while maintaining a high operation speed.

Index Terms--- Computer Arithmetic, Modulo2^n+1 Adders, Residue Number System, Diminished-1 Representation, Parallel-prefix Carry Computation

I. INTRODUCTION

Modulo arithmetic has been used in digital computing systems for many years. In particular,modulo2n+1 arithmetic appears to play an important role in a variety of applications. Arithmetic modulo2n+1 is commonly met in their sidue number system (RNS) which is an arithmetic system well-suited to applications in which the operations are limited to addition, subtraction and multiplication. The RNS has been used for the design of digital signal processors FIR filters and communication components.

Modulo 2^n+1has found applicability in a variety of fields ranging from pseudorandom number generation and cryptography [1], [2], [3], up to convolution computations without round-off errors [4], [5], [6]. Also, modulo 2^n+1operators are commonly included in residue number system (RNS) applications [7], [8], [9]. The RNS is an arithmetic system which decomposes a number into parts (residues) and performs arithmetic operations in parallel for each residue without the need of carry propagation among them, leading to significant speedup over the corresponding binary operations. RNS is well suited to applications that are rich of addition/subtraction and multiplication operations and has been adopted in the design of digital signal processors [7], [10], [11], FIR filters [12], [13], [14] and communication components [15], [16], [17], offering in several cases apart from enhanced operation speed, low-power characteristics [18].

The complexity of a modulo 2^n+1arithmetic unit is determined by the representation chosen for the input operands. Three representations have been considered; namely, the normal weighted one, the diminished-1 [19] and the signed-LSB representations [20]. We only consider the first two representations in the following, since the adoption of the signed-LSB representation does not lead to more efficient circuits in delay or area terms. In every case, when performing arithmetic operations modulo $2^n+1$the input operands and the results are limited between 0 and 2^n.

In the normal weighted representation, each operand requires n+1 bits for its representation but only utilizes 2^n+1 representations out of the 2^n+1 that these can provide. A denser encoding of the input operands and simplified arithmetic operations modulo 2^n+1 are offered by the diminished-1 representation. In the diminished-1 representation, A is represented as $a_z A^*$, where a_z is a single bit, often called the zero indication bit, and A^*, is an n-bit vector, often called the number part. If $A > 0$, then $a_z= 0$ and $A^* = A -1$, whereas for $A = 0$; $a_z= 1$, and $A^* = 0$. For example, the diminished-1 representation of A =9 modulo 17 is 01000_2.

Mohammed Mosin A. Junjwadkar, Industrial Electronics, ECE Department, KLSVDRIT, Haliyal, India. E-mail: 2vd12lie12@gmail.com
Virapax M. Chougala, Assistant Professor, ECE Department, KLSVDRIT, Haliyal, India.

II. PRELIMINARIES

Suppose that A = A_{n-1} ,A_{n-2} , ,…..A_0 and B= B_{n-1} ,B_{n-2} , ,…..B_0 represent the two numbers to be added and S= S_{n-1} ,S_{n-2} , ,…..S_0 denotes their sum. An adder can be considered as a three-stage circuit. The preprocessing stage computes the carry-generate bits Gi, the carry-propagate bits Pi, and the half-sum bits Hi, for every i, $0 \le i \le n-1$, according to

$$G_i = A_i \cdot B_i \qquad P_i = A_i + B_i \qquad H_i = A_i \oplus B_i,$$

Where ·, +and ⊖ denote logical AND, OR, and exclusive-OR, respectively. The second stage of the adder, hereafter called the carry computation unit, computes the carry signals Ci, for $0 \le I \le n-1$ using the carry generate and carry propagate bits Gi and Pi. The third stage computes the sum bits according to

$$S_i = H_i \oplus C_{i-1}.$$

Carry computation is transformed into a parallel prefix problem using the O operator, which associates pairs of generate and propagate signals and was defined in [35] as

$$(G, P) \circ (G', P') = (G + P \cdot G', P \cdot P').$$

In a series of associations of consecutive generate/propagate pairs (G, P), the notation (Gk:j, Pk:j), with k > j, is used to denote the group generate/propagate term produced out of bits k, k-1, . . . , j, that is,

$$(G_{k:j}, P_{k:j}) = (G_k, P_k) \circ (G_{k-1}, P_{k-1}) \circ \cdots \circ (G_j, P_j).$$

Since every carry Ci = Gi:0, a number of algorithms have been introduced for computing all the carries using only Ooperators. Fig. 1 presents the most well-known approaches for the design of an 8-bit adder, while Fig. 2 depicts the logic-level implementation of the basic cells used throughout the paper.

For large word lengths, the design of sparse parallel prefix adders is preferred, since the wiring and area of the design are significantly reduced without sacrificing delay. The design of sparse adders relies on the use of a sparse parallel-prefix carry computation unit and carry-select (CS) blocks. Only the carries at the boundaries of the carry-select blocks are computed, saving considerable amount of area in the carry-computation unit [39]. A 32-bit adder with 4-bit sparseness is shown in Fig. 3a. The carry select block computes two sets of sum bits corresponding to the two possible values of the incoming carry. When the actual carry is computed, it selects the correct sum without any delay overhead. A possible logic-level implementation of a 4-bit carry-select block is shown in Fig. 3b.

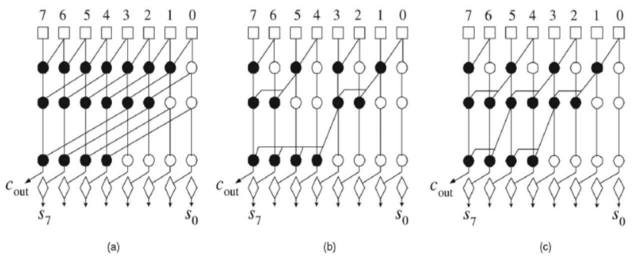


Fig. 1: Examples of 8-Bit Parallel-Prefix Structures for Integer Adders. (a) Kogge-Stone [36], (b) Ladner-Fischer [37], and (c) one Representative of the Knowles [38] Family of Adders

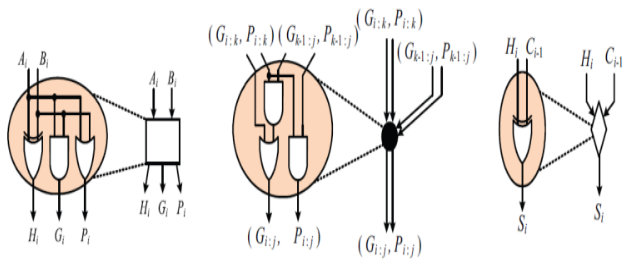


Fig. 2: The Logic-Level Implementation of the Basic Cells Used in Parallel-Prefix Adders

III. MODULO 2N+1 ADDERS

Diminished-1 modulo 2^n+1 addition is more complex since special care is required when at least one of the input operands is zero (100 . . . 0). The sum of a diminished-1 modulo adder is derived according to the following cases:

- When none of the input operands is zero (a_z $b_z \neq 0$) their number parts A^* and B^* are added modulo2^n+1. This operation as discussed in the following, can be handled by an IEAC adder.
- When one of the two inputs is zero the result is equal to the nonzero operand.
- When both operands are zero, the result is zero.

In any case that the result is equal to zero (cases 1 or 3), the zero-indication bit of the sum needs to be set and the number part of the sum should be equal to the all-zero vector. According to the above, a true modulo addition in a diminished-1 adder is needed only in case 1, while in the other cases the sum is known in advance.

When none of the input operands is zero, $a_z b_z \neq 1$, the number part of the diminished-1 sum is derived by the number parts A^* and $B^*$of the input operands as follows:

$$\begin{aligned} S^+ &= (A^\star + B^\star) \bmod (2^n + 1) \\ &= \begin{cases} (A^\star + B^\star + 1) \bmod 2^n, & A^\star + B^\star < 2^n, \\ (A^\star + B^\star) \bmod 2^n, & A^\star + B^\star \geq 2^n. \end{cases} \end{aligned} \tag{3}$$

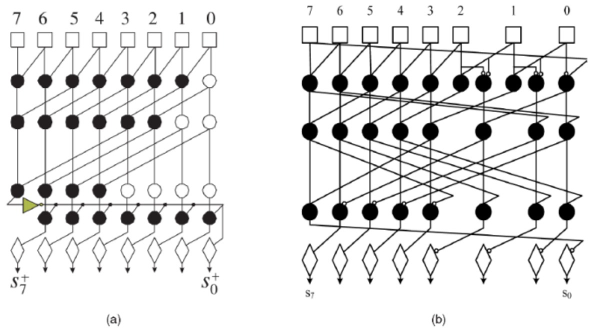


Fig. 3: Parallel Prefix Modulo 2^8+1adders: (a) Using an Additional Carry-Increment Stage and (b) Recirculating the End Around Carry within the Existing log2 n Prefix Levels

Equation (3) reveals that an IEAC adder can be used for providing the number part in this case. Fig. 3a [22], [23] presents the implementation of an IEAC adder by the addition of a carry increment stage to an integer parallel prefix adder.

IV. New Sparse Modulo 2n+1 Adders

Sparse refer to the design of the adder where a carry select block will be used to obtain the output sum. The possible outputs of sum will be calculated and the output carry will select which output sum will be selected. The design is based at parallel-prefix adders. In the sparse carry computation unit for sparse modulo 2n +1 diminished adders some prefix operators are doubled up, since 2 carry computations need to be performed in parallel; one on normal propagate and generate signals, while the other on their complements. The problem gets worse when the input operands' width is not a power of two. So there is still a lot of space for improvement. This problem is removed by introducing a new prefix operator and an even simpler carry computation unit. While calculating the carry for the spares adder we found that several operators double up since the operators is calculated in parallel. For larger adders, significantly more operators need to be doubled up, leading to increased area and wiring. To overcome this problem, we need a prefix operator that can associate the operation. We introduce a new operator, hereafter called gray operator.

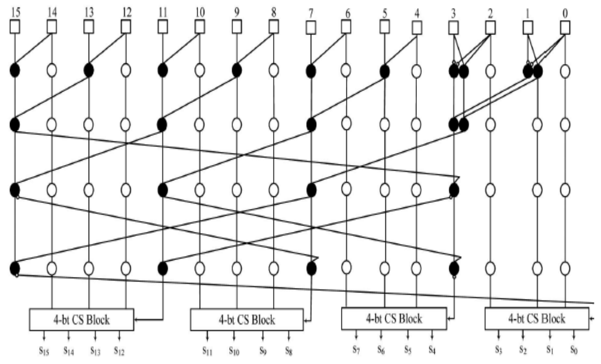


Fig. 4: Modulo 2^{16}+1 Diminished Adder Using a Sparse Carry Computation Unit

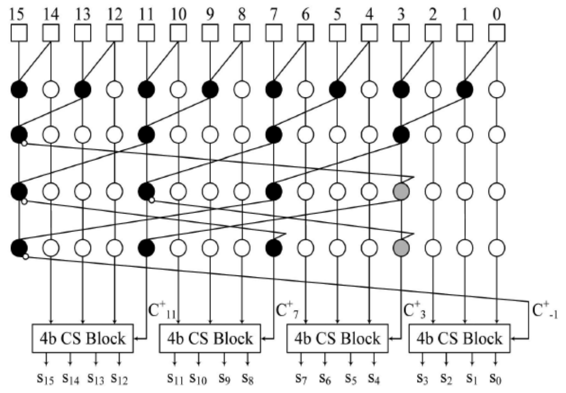


Fig. 5: Sparse-4 Modulo2^{16}+1 Diminished-1 Adder

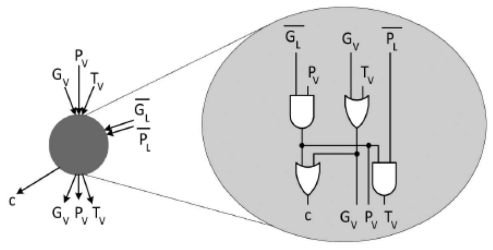


Fig. 6: Gray Prefix Operator; Notation and Implementation

Diminished-1 modulo 2n+1 addition is not used for addition of zero operands. When the input operand is zero, it should be handled separately. Zero treatment leads to slow and area-consuming implementations.

Architecture of Zero handling unexceptional modulo adders will be designed and the performance evaluation by means of area, delay, power dissipation will be compared with existing modulo adder. The problem of designing an Extra hardware for handling Zero addition problem is recovered in our proposal by implementing a MUX based selection line is used to decide whether to perform addition in case of input zero.

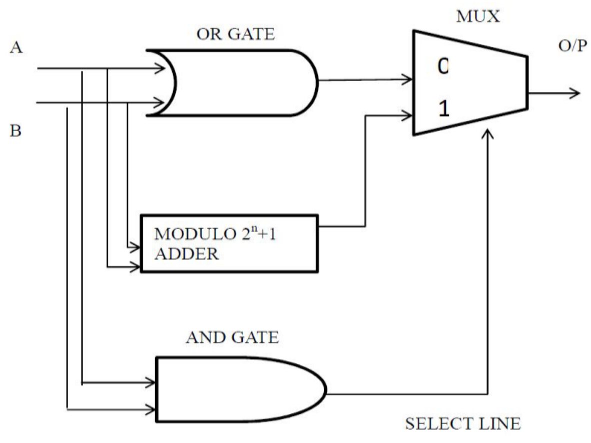


Fig. 7: Adder with Zero Handling

Table 1: Operation of Adder in Fig7

A	B	OR o/p	AND o/p	Select	Output
0	0	0	0	0	0
0	B	B	0	0	B
A	0	A	0	0	A
A	B	-	AB	1	A+B mod

V. APPLICATION (FIR FILTER)

A Filter is frequency selective network, which is used to modify an input signal in order to facilitate further processing. Basically there are two types of filters-analog and digital. Digital Filters are widely used in different areas, because Digital filters have the potential to attain much better signal to noise ratio than analog filters. The digital filter performs noiseless mathematical operations at each intermediate step in the transform and their precise reproducibility allows design engineers to achieve performance levels that are difficult to obtain with analog filters Digital filters operate on numbers opposite to analog filters, which operates on voltages. The basic operation of digital filter is to take a sequence of input numbers and compute a different sequence of output numbers. There exists a range of different digital filters. FIR and IIR filters are the two common filter forms. A drawback of IIR filters is that the closed-form IIR designs are preliminary limited to low pass, band pass, and high pass filters, etc. secondly FIR filters can have precise linear phase. Also, in the case of FIR filters, closed-form design equations do not exist and the design problem for FIR filters is much more under control than the IIR design problem. A FIR filter is a filter structure that can be used to implement almost any sort of frequency response digitally. It is usually implemented by using a series of delays, multipliers, and adders to create the filter's output. The architecture of FIR filter is shown in Fig 8.

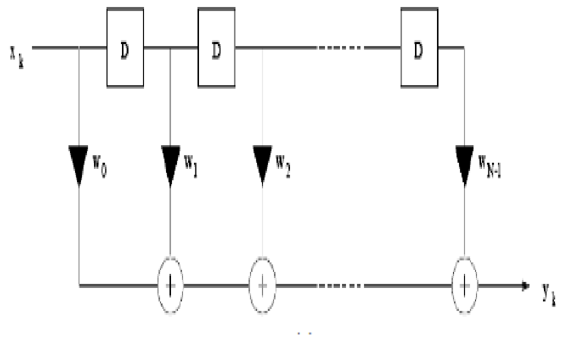
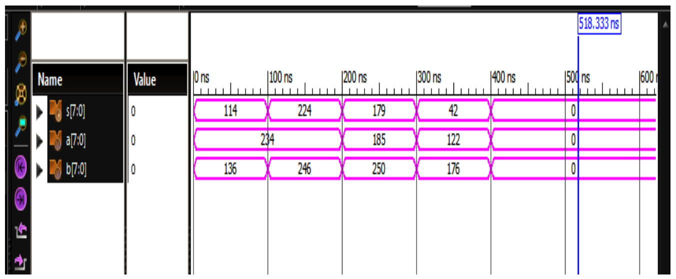


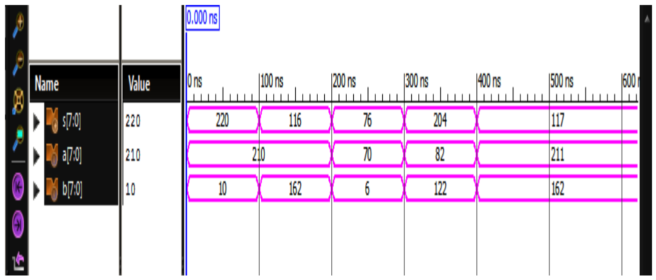
Fig. 8: FIR Filter Architecture Canonical Form

$$H(Z)=w_0+w_1z^{-1}+w_2z^{-2}+\text{-------}+w_{M-1}Z^{L-1}$$

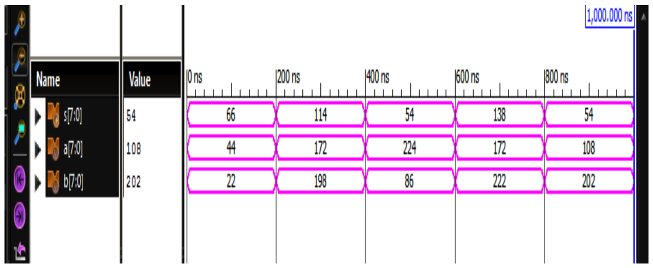
VI. SIMULATION RESULTS



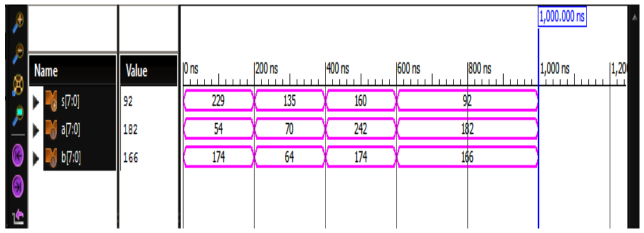
Simulation Result of Kogge stone adder in fig 1(a)



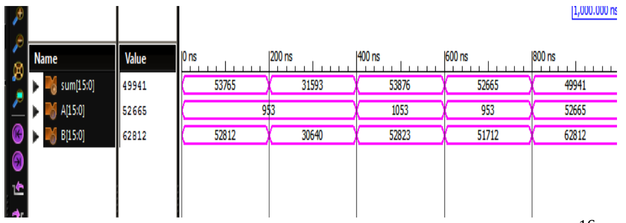
Simulation Result of Lander fishner adder in fig 1(b)



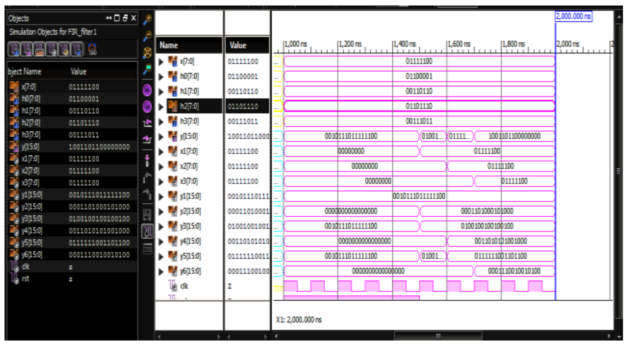
Simulation Result of Knowles adder in fig 1(c)



Simulation Result of Parallel prefix modulo 2^8+1 adders using an additional carry-increment stage and recirculating the end around carry within the Existing log2 n prefix levels in fig 3.



Simulation Result of Sparse-4 modulo $2^{16}+1$ diminished-1 adder in fig 5.



Simulation Result of FIR Filter in fig 8.

VII. PERFORMANCE COMPARISON

The proposed multiplier is synthesized in Xilinx 13.1 ISE simulator to find optimized delay and area of different modulo adder architectures .The modulo adders are compared that can be seen below

Table 2: Comparison of 8 bit Adders

	Kogge stone(8bit) fig 1(a)	Lander Fisher(8bit) fig 1(b)	Knowles (8bit) fig 1(c)	Parallel Prefix(8bit) fig 3(a)	Parallel Prefix(8bit) fig 3(b)
Delay	9.959ns	11.003ns	9.082ns	14.506ns	9.09ns
Slices	11	8	10	19	17
4 i/p LUT's	20	14	19	34	31
IO's	24	24	24	24	24
Bonded IOB's	24	24	24	24	24
Logic Levels	7	8	6	11	6

Table 3: Comparison of 16 bit Adders

	$2^{16}+1$ Diminished 16 bit	$2^{16}+1$ Sparse 4 16 bit
Delay	13.303ns	7.577ns
Slices	30	8
4 i/p LUT's	50	16
IO's	48	48
Bonded IOB's	48	48
Logic levels	11	19

Table 4: Comparison of Filter Output

	Using216+1 Diminished16 bit	Using216+1 Sparse 4 16 bit
Delay	24.962ns	17.778ns
Slices	40	24
4 i/p LUT's	70	46
IO's	56	56
Bonded IOB's	40	40
Logic levels	11	14

VIII. CONCLUSION

Efficient modulo 2n+1adders are appreciated in a variety of computer applications including all RNS implementations.

In this paper, a contribution is offered to the modulo 2n+1addition problem. A novel architecture has been proposed that uses a sparse totally regular parallel-prefix carry computation unit. This architecture was derived by proving the inverted circular idem potency property of the parallel-prefix carry operator in modulo 2n+1addition and by introducing a new prefix operator that eliminates the need for a double computation tree in the earlier fastest proposals. The experimental results indicate that the proposed architecture heavily outperforms the earlier solutions in implementation area and power consumption, while offering a high execution rate. The drawback for the modulo adder is that it is not used for adding zero operands. The adder is proposed that will eliminate this drawback, and zero operands can be added effectively and fast. An FIR filter is implemented using the proposed adders.

REFERENCES

[1] X. Lai and J.L. Massey, "A Proposal for a New Block Encryption Standard," EUROCRYPT, D.W. Davies, ed., vol. 547, pp. 389-404,Springer, 1991.

[2] R. Zimmermann et al., "A 177 Mb/s VLSI Implementation of the International Data Encryption Algorithm," IEEE J. Solid-State Circuits, vol. 29, no. 3, pp. 303-307, Mar. 1994.

[3] H. Nozaki et al., "Implementation of RSA Algorithm Based on RNS Montgomery Multiplication," Proc. Third Int'l Workshop Cryptographic Hardware and Embedded Systems, pp. 364-376, 2001.

[4] Y. Morikawa, H. Hamada, and K. Nagayasu, "Hardware Realisation of High Speed Butterfly for the Maximal Length Fermat Number Transform," Trans. IECE, vol. J66-D, no. 1, pp. 81-88, 1983.

[5] M. Benaissa, S.S. Dlay, and A.G.J. Holt, "CMOS VLSI Design of a High-Speed Fermat Number Transform Based Convolver/Correlator Using Three-Input Adders," Proc. IEE, vol. 138, no. 2, pp. 182-190, Apr. 1991.

[6] V.K. Zadiraka and E.A. Melekhina, "Computer Implementation of Efficient Discrete-Convolution Algorithms," Cybernetics and Systems Analysis, vol. 30, no. 1, pp. 106-114, Jan. 1994.

[7] M.A. Soderstrand et al., Residue Number System Arithmetic: Modern Applications in Digital Signal Processing. IEEE Press, 1986.

[8] P.V.A. Mohan, Residue Number Systems: Algorithms and Architectures. Springer-Verlag, 2002.

[9] A. Omondi and B. Premkumar, Residue Number Systems: Theory and Implementations. Imperial College Press, 2007.

[10] J. Ramirez et al., "RNS-Enabled Digital Signal Processor Design,"Electronics Letters, vol. 38, no. 6, pp. 266-268, Mar. 2002.

[11] J. Ramirez et al., "Design and Implementation of High Performance RNS Wavelet Proccessors Using Custom IC Technologies," J. VLSI Signal Processing Systems, vol. 34, no. 3,pp. 227-237, July 2003.

[12] J. Ramirez et al., "High Performance, Reduced Complexity Programmable RNS-FPL Merged FIR Filters," Electronics Letters,vol. 38, no. 4, pp. 199-200, Feb. 2002.

[13] G.C. Cardarilli, A. Nannarelli, and M. Re, "Reducing Power Dissipation in FIR Filters Using the Residue Number System," Proc. 43rd IEEE Midwest Symp. Circuits and Systems, pp. 320-323, Aug. 2000.

[14] Y. Liu and E.M.-K. Lai, "Moduli Set Selection and Cost Estimation for RNS-Based FIR Filter and Filter Bank Design," Design Automation for Embedded Systems, vol. 9, no. 2, pp. 123-139, June 2004.

[15] U. Meyer-Base, A. Garcia, and F. Taylor, "Implementation of a Communications Channelizer Using FPGAs and RNS Arithmetic," J. VLSI Signal Processing Systems, vol. 28, nos. 1/2, pp. 115-128, May/June 2001.

[16] J. Ramirez et al., "Fast RNS FPL-Based Communications Receiver Design and Implementation," Proc. 12th Int'l Conf. Field Programmable Logic, pp. 472-481, 2002.

[17] M. Panella and G. Martinelli, "An RNS Architecture for Quasi- Chaotic Oscillators," J. VLSI Signal Processing Systems, vol. 33, no. 1, pp. 199-220, Jan./Feb. 2003.

[18] R. Chokshi, K.S. Berezowski, A. Shrivastava, and S.J. Piestrak,"Exploiting Residue Number System for Power-Efficient Digital Signal Processing in Embedded processors," Proc. Int'l Conf. Compilers, Architecture, and Synthesis for Embedded Systems (CASES'09), pp. 19-28, 2009.

[19] L.M. Leibowitz, "A Simplified Binary Arithmetic for the Fermat Number Transform," IEEE Trans. Acoustics, Speech and Signal Processing, vol. ASSP-24, no. 5, pp. 356-359, Oct. 1976.

[20] G. Jaberipur and B. Parhami, "Unified Approach to the Design of Modulo-(2n -1) Adders Based on Signed-LSB Representation of Residues," Proc. 19th IEEE Symp. Computer Arithmetic, pp. 57-64,2009.

[21] R. Zimmermann, "Binary Adder Architectures for Cell-Based VLSI and Their Synthesis," PhD dissertation, Swiss Fed. Inst. Of Technology, 1997.

[22] R. Zimmerman, "Efficient VLSI Implementation of Modulo 2^n+1Addition and Multiplication," Proc. 14th IEEE Symp. Computer Arithmetic, pp. 158-167, Apr. 1999.

[23] H.T. Vergos, C. Efstathiou, and D. Nikolos, “Diminished-One Modulo 2^n+1Adder Design,” IEEE Trans. Computers, vol. 51, no. 12, pp. 1389-1399, Dec. 2002.

[24] C. Efstathiou, H.T. Vergos, and D. Nikolos, “Modulo 2n -1 Adder Design Using Select Prefix Blocks,” IEEE Trans. Computers, vol. 52, no. 11, pp. 1399-1406, Nov. 2003.

[25] H.T. Vergos and C. Efstathiou, “Efficient Modulo 2^n+1Adder Architectures,” Integration, the VLSI J., vol. 42, no. 2, pp. 149-157,Feb. 2009.

[26] G. Dimitrakopoulos and D. Nikolos, “High-Speed Parallel-Prefix VLSI Ling Adders,” IEEE Trans. Computers, vol. 54, no. 2, pp. 225-231, Feb. 2005.

[27] Rakhi Thakur and Kavita Khare “High Speed FPGA Implementation of FIR Filter for DSP Applications”,International Journal of Modeling and Optimization, Vol. 3, No. 1, February 2013