IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 6, June 2016

On Modulo $2^n + 1$ Adder Design

Krithi B Shetty[1], Ashwitha S[1] , Athmika D Shetty[1], Anusha A[1], Ashutha K[2], Kiran Kumar V G[3]

Students, ECE Dept, Sahyadri College of Engineering & Management, Mangalore, India [1]

Assistant Professor, ECE Dept, Sahyadri College of Engineering & Management, Mangalore, India [2]

Professor, ECE Dept, Sahyadri College of Engineering & Management, Mangalore, India [3]

Abstract: The paper discusses the application of VLSI technology to implement the functions of multi operand floating point addition in parallel using verilog, targeting it to a Xilinx FPGA. The multi operand floating point adders perform two additions in a single unit to achieve better performance and accuracy. To improve the performance and accuracy, several optimization techniques are applied. These are a new exponent compare and significant alignment, dual-reduction, early normalization, three input leading zero anticipation and compound addition and rounding. The traditional fused floating point three term adder takes twice the area, power consumption and delay. In order to reduce the overhead the multi-operand floating point adder has been proposed. Which perform rounding only once, which improve the accuracy.

Keywords: Xilinx FPGA, adders, zero anticipation, multi-operand floating point adder.

I. INTRODUCTION

Arithmetic modulo2^n+1 has found applicability in a variety of fields ranging from pseudorandom number generation and cryptography up to convolution computations without round-off errors. Also, modulo 2^n+1operators are commonly included in residue number system (RNS) applications. 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 FIR filters and communication components offering in several cases apart from enhanced operation speed, low-power characteristics [1].

A denser encoding of the input operands and simplified arithmetic operations modulo 2^n+1 are offered by thediminished-1 representation. In the diminished-1 representation, A is represented asaz A*, where az is a single bit, oftencalled the zero indication bit, and A* is an n-bit vector, oftencalled the number part. IfA> 0, then az= 0 and A* =A -1,whereas for A=0; az=1, and A*=0. For example, thediminished-1 representation of A=5 modulo 17 is 001002.Considering that the most common operations required in modulo2^n+1arithmetic are negation, multiplication bya power of two and addition , the adoption of the diminished-1 representation, allows to limit these operationsto n bits. Specifically, negation is performed by complementing every bit of A*, if az= 0 and inhibitingany change when az=1 [2]. Multiplication by 2i is performed by an i-bit left rotation of the bits of A*, in which the reentering bits are complemented, if az =0 and inhibiting anychange when az=1 [3].

Finally, the addition of azA* with bzB*,boils down to an n-bit modular addition of A* with B* withsome minor modifications.

Modulo arithmetic appears to play an important role in many applications. The Residue Number System (RNS) [Sonderstrand et al. 1986, Bayoumi et al [4]. 1987, Elleithy and Bayoumi 1992, Koren 1993] is a first application field. A set of moduli, suppose, that are pair-wise relative prime is used to define a RNS. Any integer X,with $0 \le X < M$, where $M = m_1 \times m_2 \times ... \times m_L$ has a unique representation in the RNS, given by the L - tuple of residues $X = (x_1, x_2, ..., x_L)$, where $x_i = X \bmod m_i$,if $X \ge 0$ and $x_i = (M - |X|) \bmod m_i$,if $X < 0$. A RNS operation, suppose*, is defined as $(z_1, z_2, ..., z_L) = (x_1, x_2, ..., x_L) * (y_1, y_2, ..., y_L)$, where $z_i = (x_i * y_i) \bmod m_i$ [5].

II. LITERATURE SURVEY

"Modulo (2n+1) arithmetic logic",Agrawal D. P. and Rao T.R.N.et.al in this paper a novel format for representing module (2^n+1) number is shown to be helpful in achieveing modular addition and complementation logic for fast addition on using carry look ahead and modular complementation is done.Drawbacks of this method include the use of customised parallel prefix adder that cannot be replaced by alternative n bitadders.Thus exploration of design spacewith regard to area/speed/power tradeoffs must be performed a new for each design [6].

"Binary adder architectures for cell-based VLSI and their synthesis", Zimmermann R..et.al In this paper if we ignore the represention of o and concentrate only on the part, then efficient bits wide address can be constructed at the output of adder a row of multiplexers is added in order to select correct output.Demerits of this paper are delay of the multiplexers makes the delay of modulo channel greater than channels of the form and the required implementation

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 6, June 2016

area is also increased substantially when compared against the other channels [7].

"Fast and flexible architectures for RNS arithmetic decoding".Elleithy K. M., and BayoumiM. A., et.al In this paper the decoder is flexible since the decoded data can be selected to either unsigned magnitude or 2's complement binary number.Two different architectures are analyzed; the first one is based on using carry saved adders,while the other is based on utilizing module adders.Demerits are the CRT provides a direct, fast and simple conversion formula,the lack of large and fast modulo M adder has held back this approach [8].

A. Problem Definition

Suppose that a number X , with $0 \le X \le 2^n$, is represented by $n+1$ bits, as $x_z X^* = x_z x^*_{n-1} x^*_{n-2} \ldots x^*_1 x^*_0$, where x_z is the zero indication bit and X^* is the diminished-one representation of X, that is :

$$x_z = \begin{cases} 0, \text{if } X \neq 0 \\ 1, \text{if } X = 0 \end{cases} \quad \text{and} \quad X^* = \begin{cases} X\text{-}1, \text{if } X \neq 0 \\ 0, \quad \text{if } X = 0 \end{cases}$$

Obviously, $X = X^* + \overline{x_z}$ (the $\overline{\ }$ notation is used for logical negation). A similarrepresentation was adopted in [Agrawal and Rao 1978]. If we decide to ignore the representation of 0 and concentrate only on the X^* part, then efficient n-bits wide adders can be constructed as proposed in [Zimmermann 1997, 1999, Efstathiou et al. 2001, Vergos et al. 2001, 2002]. However, the need to treat 0 distinctly will result in a circuit as the one shown in fig 1 [9].

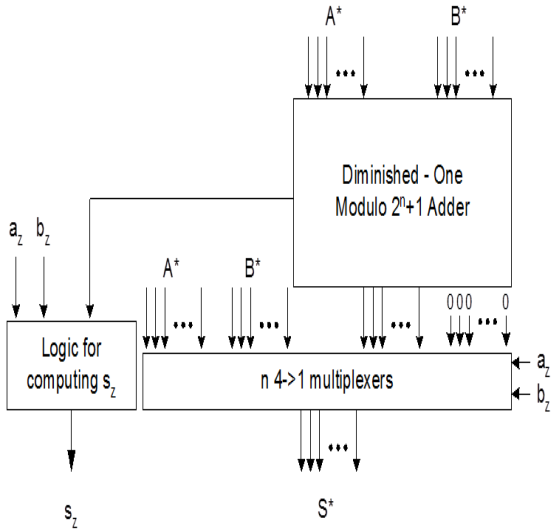

Fig. 1.The architecture of an adder that treats 0 as a special case.

At the output of the adder a row of multiplexers is added in order to select the correct output. When none of the operands is zero (both and are 0) the output of the adder is propagated. When one of the two operands is zero, the other operand is allowed to propagate. When both operands are 0, the all 0s word is propagated. Logic also needs to be added for computing the zero indication bit s_z , of the result.Obviously, the adders that follow the architecture of fig 1 are not the best choice in RNS applications, since the delay of the multiplexers makes the delay of the modulo 2^n+1 channel greater than channels of the form 2^n and 2^n-1 . The required implementation area is also increased substantially when compared against the other channels.

III.PROPOSED MODULO 2N+1 ADDERS

In this section we propose CLA and parallel-prefix adders with embedded treatment of zero operands. We first analyse the logic of s_z and S^* , for the adopted number representation.operands.Relation $|A+B|_{2^n+1}=0$, with $0 \le A, B \le 2^n$, implies that $A=B=0$ or that $A, B \neq 0$ and $A+B=2^n+1$, or , or equivalently that $A=B=0$ or $A, B \neq 0$ and $A^*+B^*=2^n-1$. Therefore, the zero indication bit, s_z , of the result should be one, when either both operands are 0, or when both operands are non-zero but their diminished-one parts are complementary. These two cases are expressed by the following relation for s_z :

$$s_z = (a_z \cdot b_z) \vee (\overline{a_z} \cdot \overline{b_z} \cdot P_{n-1}) = (a_z \cdot b_z) \vee (\overline{(a_z \vee b_z)} \cdot P_{n-1})$$

where $P_{n-1} = p_{n-1} \cdot p_{n-2} \cdot \ldots \cdot p_0$ and $p_i = a^*_i \oplus b^*_i$, with $i = 0, 1, \ldots, n-1$ and the notations $\cdot, \vee, \oplus$ are used for the logical AND, OR and exclusive-OR operations respectively. The resulting implementation is presented in fig 2. Note that the part of fig 1. which produces P_{n-1} is not required in the case of diminished-one adders whose carry propagate signals are defined as the exclusive-OR of corresponding operands' bits. Such adders are well known as exclusive-OR adders. On the other hand, the logic producing P_{n-1} is required in inclusive-OR adders, that is, in adders whose carry propagate signals are defined as the inclusive-OR of corresponding operands' bits [10].

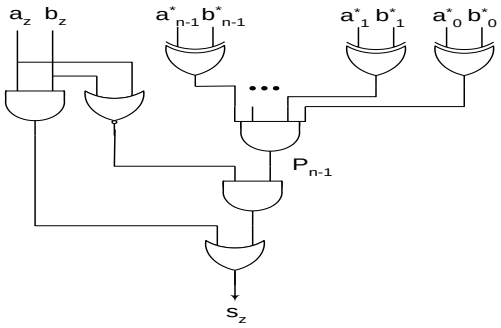

Fig. 2. The implementation of s_z.

For the S^* part of the result we can observe that :If A^* and B^* are both non zero, then according to

IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 6, June 2016

[Zimmermann 1997, 1999] :

$$S^* \bmod (2^n+1) = \begin{cases} (A^* + B^*) \bmod 2^n, & \text{if } A^* + B^* \geq 2^n \\ A^* + B^* + 1 & \text{otherwise} \end{cases}$$

or equivalently, in this case can be computed by adding the complement of the carry output ($\overline{c_{out}}$) of the modulo 2^n addition of A^* and B^* back to the sum. Since in this case $a_z = b_z = 0$, we can instead of $\overline{c_{out}}$, add $\overline{(a_z \vee b_z \vee c_{out})}$.If one or both operands are zero, $\overline{(a_z \vee b_z \vee c_{out})} = 0$ and S^* is equal to the modulo 2^n sum of A^* and B^* . That is, in this case S^* can also be computed by a modulo 2^n adder with a carry input of $\overline{(a_z \vee b_z \vee c_{out})}$.

A. CLA Adders

All One straightforward way for avoiding oscillations, is to compute two possible sums in two distinct modulo 2^n adders and then use function F for choosing between the two sums. However, since this arrangement requires two sets of n-bit adders and n multiplexers, a better solution is to design dedicated modular CLA architectures. This can be achieved by engaging the equation of into the CLA unit and simplifying the resulting equations [Agrawal and Rao 1978, Efstathiou et al. 1994, Vergos et al. 2002]. Let g_k and p_k denote the carry generate and propagate terms respectively; that is $g_k = a_k \cdot b_k$, and $p_k = a_k \oplus b_k$. The carries , with $-1 \leq k \leq n-1$ (c_{-1} is the input carry, while c_{n-1} the output carry) in an n-bit CLA adder are computed by unfolding the recursive equation $c_k = g_k \vee p_k \cdot c_{k-1}$and in parallel implementing the resulting equations. For the inverted carry at each bit position c_k , we have :

$$\overline{c_k} = \overline{(g_k \vee p_k \cdot c_{k-1})} = \overline{g_k} \cdot (\overline{p_k} \vee \overline{c_{k-1}}) = \overline{g_k} \cdot \overline{p_k} \vee \overline{g_k} \cdot \overline{c_{k-1}} = \overline{t_k} \vee \overline{g_k} \cdot \overline{c_{k-1}}$$

where $t_k = a_k \vee b_k$ .The sum bits are given by $0 \leq k \leq n-1$

B. Parallel-Prefix Adders With Carry Increment Stage

Suppose that$A = A_{n-1}A_{n-2} \ldots\ldots A_0$

And $B = B_{n-1}B_{n-2} \ldots\ldots B_0$represent the two numbers to be added and $S = S_{n-1}S_{n-2} \ldots\ldots S_0$ 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≤i≤n-1,according to$G_i = A_iB_i, P_i = A_i + B_i, H_i = A_i \oplus B_i$; where . ,+ ,⊕,denote logical AND, OR, and exclusive- OR, respectively. The second stage of the adder, hereaftercalled the carry computation unit, computes the carry signals Ci, for 0≤i≤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 * operator, which associates pairs of generate and propagate signals and was defined as $(G, P) o (G', P') = (G + P.G', P.P')$ In a series of associations of consecutive generate/propagatepairs (G,P) the notation $(G_{k:j}, P_{k:j})$ with k > j, is usedto denote the group generate/propagate term produced outof bits k; k – 1, . . . , j, that is,

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

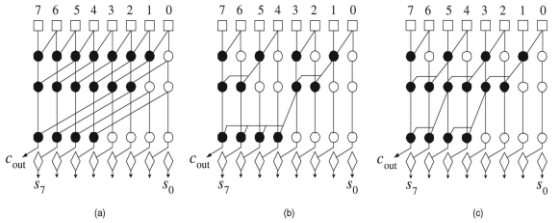

Fig. 3 Examples of 8-bit parallel-prefix structures for integer adders. (a) Kogge-Stone, (b) Ladner-Fischer, and (c) one representative of the Knowles family of adders.

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. A 32-bit adder with 4-bit sparseness is shown in Fig 3The 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 3

To solve the problem of oscillations in parallel-prefix adder architectures, a first solution is to use prefix architectures with fast carry processing as proposed in [Abraham and Gajski 1980] and then utilize the theory developed in [Zimmermann 1997, 1999], for re-entering the expression as the carry input. The resulting architecture is outlined in fig 3, presents the gate level implementations of the operators used in fig 3.

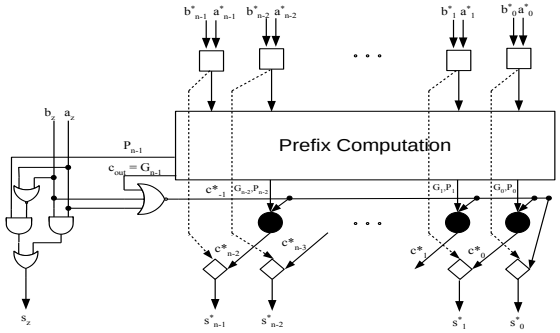

Fig. 4 The architecture of a parallel-prefix adder with a carry increment stage.

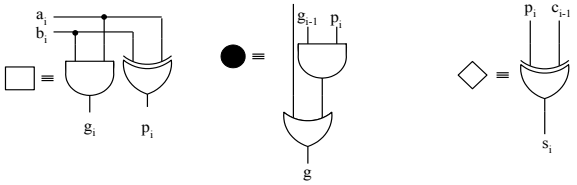

Fig. 5.Gate level implementations of the operators used in fig 3.

DOI 10.17148/IJIREEICE.2016.4641

IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 6, June 2016

C. Totally Parallel-Prefix Adders

Figs Instead of having a dedicated single stage for the re-entering carry, in [Kalamboukas et al. 2000, Vergos et al. 2001, 2002] it has been proposed to perform carry recirculation at each existing prefix level. In this way the need for an extra carry increment stage is cancelled and dedicated totally parallel-prefix adder architectures result with one less prefix level.

In the case that the re-entering carry is given by the expression $\overline{(a_z \vee b_z \vee c_{out})}$, allowing carry recirculation at each existing prefix level, we can get that the carries c_i^* of the modulo 2^n+1 addition are equal to G_i^* , where G_i^* is computed by the prefix equations :

$$(G_i^*, P_i^*) = \begin{cases} \overline{(G_{n-1}, P_{n-1})}, & \text{if } i = -1 \\ (G_i, P_i) \circ \overline{(G_{n-1,i+1}, P_{n-1,i+1})} & \text{if } 0 \le i \le n-2. \end{cases}$$

where :

$\overline{(G,P)}$ is defined to be equal to $(\overline{G}, P)$

$G_{a,b}$ and $P_{a,b}$, $a > b$, are respectively the group $a, a-1, \ldots, b$ generate and propagate signals for the group , computed by $(G_{a,b}, P_{a,b}) = (g_a, p_a) \circ (g_{a-1}, p_{a-1}) \circ \ldots \circ (g_b, p_b)$. Obviously, $(G_{a,0}, P_{a,0}) = (G_a, P_a)$, and $g_{n-1} = (a_{n-1} \cdot b_{n-1}) \vee a_z \vee b_z$ legible.

IV. MODULO $2^n \pm 1$ ADDITION BASICS

A. Modulo $2^n - 1$ adders

The computation of modulo2^n-1addition is, in fact, aconditional operation defined as

$$(A+B) \bmod (2^n-1) = \begin{cases} (A+B), & A+B < 2^n, \\ (A+B+1) \bmod 2^n & A+B \ge 2^n \end{cases}$$

A modulo2^n-1adder can be implemented using an integer adder that increments also its sum when the carry output is one, that is, when A+B≥2^n. This is also equivalent tofeeding the carry-input of the adder with the carry-output of the first addition. The conditional increment can be implemented by an additional carry increment stage as shown in Fig 4.a. In this case, one extra level of , cells driven by the carry output of the adder, is required. Depending on the implementation of the modulo 2^n-1 adder, for bitwise-complementary inputs, i.e., when A+B=2^n-1, the adder may produce an all 1s output vector, in place of the expected result which is equal to zero. In most applications, this is acceptable as a second representation for zero.

V. KOGGE STONE ADDER

The Kogge-Stone adder is a parallel prefix form of carry look-ahead adder. It generates the carry signals in O($\log_2$N) time, and is widely considered as the fastest adder design possible. It is the most common architecture for high-performance adders in industry. TheKogge-Stone adder concept was first developed by Peter M. Kogge and Harold S. Stone.

In Kogge-stone adder, carries are generated fast by computing them in parallel at the cost of increased area. Tree structures of carry propagate and generate signals in 8-bit Kogge Stone Adder (KSA) is shown in Fig 5 Carry generation network is the most important block in tree adders, and it consists of three components such as Black cell, Grey cell and Buffer. Black cells are used in the computation of both generate and propagate signals. Grey cells are used in the computation of generate signals which are needed in the computationof sum in the post-processing stage. Buffersare used to balance the loading effect.

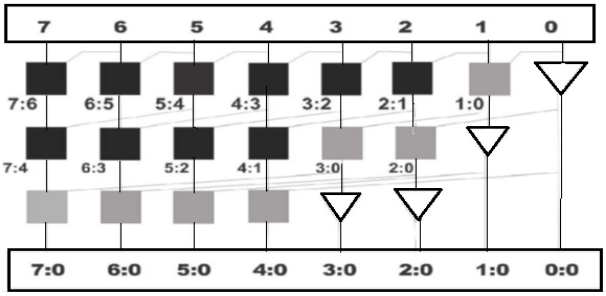

Fig. 6 bit kogge stone adder (PG)network

A.Working Of Kogge-Stone Adder

Kogge-Stone Adder has three processingstages for calculating the sum bits, they are:

1. Pre-processing stage
2. Carry generation (PG) network
3. Post-processing stage

The above steps involved in the operationofKogge-Stone adder.

VI. ADVANTAGES

1.The reduction of maximum height of Partial Product array,which will simplify the partial product reduction tree,in terms of delay and regularity of layouts,this is of special interest of short bit.

2.Width multipliers for high performance where short but high speed bit

3.Width multiplication are common operations.

VII. EXPERIMENTAL RESULT

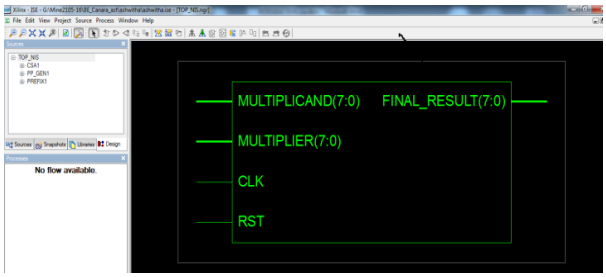

Fig. 7 RTL schematic of top module

Two input values are passed partial products are calculated using fast adder output of that will be given to booth encoder where the values will be encoded and slected values using booth selector where pre fixing and post fixing will be done as the result will be signed we use

IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 6, June 2016

Radix and booth.After all steps we get the modulo addition as the output as shown in simulation.

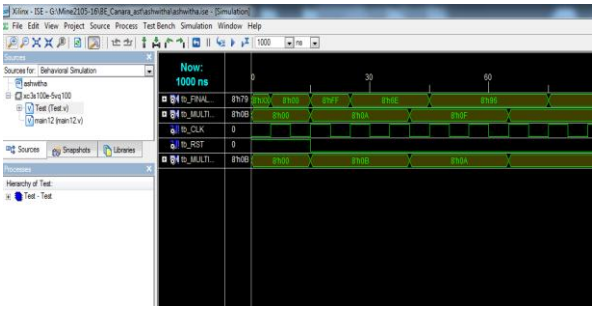

Fig. 8 Simulation output waveform

VIII. CONCLUSION

Several architectures had been recently proposed for diminished-one modulo 2^n+1 addition. None of these has dealt with the problem of handling zero operands. All of them propose to treat zero operands separately. Unfortunately, such a treatment leads to slow and area consuming implementations. In this paper, by utilizing a number representation similar to that of [Agrawal and Rao 1978], we proposed CLA and parallel-prefix adders able to also handle zero operands. Our CLA adders perform addition in a single cycle and were derived by engaging the equation of the re-entering carry into the carry computation equations. The herein proposed parallel-prefix adders with a carry increment stage offer the same logical depth and hence speed as well as the same area complexity as the diminished-one adders with a carry increment stage proposed in [Zimmermann 1997, 1999], with the extra advantage of handling zero operands. The proposed totally parallel- prefix adders perform carry re-circulation at each prefix level and therefore do not need a separate carry increment stage; their number of prefix levels and therefore their execution speed is the same as the fastest modulo 2^n+1 , modulo 2^n-1 and modulo 2^n+1 diminished-one adders. Translators between the adopted number representation and the modulo 2^n+1 binary system were finally presented.

IX. FUTURE WORK

In the case of addition, for example, the carrypropagation is limited to within a single residue (a few bits). Subtraction and multiplication also have good benefitsfrom RNS, but we will leave them for future works. This design can be more efficient if used in multi-operandarithmetical operations in a cyclic fashion. The wasted time inconversion can be overlapped with the operationitself. Definitely, we have to pipeline this design to achieve this. Moreover, the reverse conversion is carried outonly when the final result available.Finally, fault tolerance implications need further study to quantify the benefits in terms of fault coverage and systemreliability improvement. Extensions in this area include the possibility of encoding DRS pseudoresidues in BSD format.In the context of residue checking, this increases the overhead (2h instead of h+ 1 check bits for a k-bit data word).

ACKNOWLEDGEMENT

We thank Sahyadri college of engineering and management for providing space for doing the project and HOD and E&C department for their support.

REFERENCES

[1] Ashutha K., Ankitha K., "Smart Shopping cart using embedded system and wireless module", Recent Patents on Computer Science (CSENG), UAE, Vol. 8, pp. 1-6, January 2016.

[2] Ashutha K., Shetty Arpitha., et. al "Novel wireless data communication for fisherman", International journal of computer science and mobile computing (IJCSMC),Vol. 5, Issue 4, pp. 511-517, April 2016.

[3] Ashutha K., Ankitha K., "Error Minimization in BCH Codes", International Journal Of Innovative Research In Electrical, Electronics,Instrumentation And Control Engineering (IJIREEICE), Vol. 4, Issue 5, pp. 402-405, May 2016.

[4] Abraham J. A. and Gajski D. D.”Easily testable high-speed realization of register-transfer-level operations”,” Proceedings of the 10th Fault – Tolerant Computing Symposium “(FTCS-10), 339 – 344.

[5] Agrawal D. P. and Rao T.R.N. “Modulo (2n+1) arithmetic logic”.Electronic circuits and systems, Vol. 2, No. 6, 186-188.

[6] Bayoumi M. A., Jullien G.A., and Miller W.C. “A look-up table VLSI design methodology for RNS structures used in DSP applications”. IEEE Transactions on Circuits and Systems, vol. CAS-34, 604-616.

[7] Beaumont-Smith A., and Lim C. C. “Parallel prefix adder design”. Proceedings of the 15th IEEE Symposium on Computer Arithmetic, Los Alamitos, CA, USA, 218 – 225.

[8] Brent R. P. and Kung H. T.”A regular layout for parallel adders. IEEE Transactions on Computers”,1982 Vol. C-31 (3), 260–264.

[9] Curiger A”.VLSI architectures for computations in finite rings and fields.PhD thesis, Swiss Federal Institute of Technology”.1993.

[10] Efstathiou C., Nikolos D., and Kalamatianos J., 1994. Area–time efficient modulo 2n-1 adder design. IEEE Transactions on Circuits and Systems – II, Vol. 41, no. 7, 463 – 467.