



# Design of FFT Processor Using Modified Modulo $2^n+1$ Adder

Fathima Nishah P1, Ruksana Maitheen2

Asst.Professor, ILAHIA college of Engineering and Technology, Muvattupuzha, Ernakulam, Kerala, India<sup>1</sup>

Asst.Professor, ILAHIA college of Engineering and Technology, Muvattupuzha, Ernakulam, Kerala, India<sup>2</sup>

**ABSTRACT:** In this paper we present two different architectures for modulo  $2n+1$  adders and by using this an efficient FFT computation is performed. One of the architecture is based on a sparse carry computation unit in which only some of the carries are computed. In this an inverted circular idempotency property of the parallel prefix carry operator is used and its efficiency is increased by a new prefix operator. The resulting adders will be having less area and power. The second architecture is derived by modifying modulo  $2n-1$  adders with minor hardware overhead. By using this adders we can implement FFT processor with improved performance.

**KEYWORDS:** Modulo addition, parallel prefix carry computation.

## I. INTRODUCTION

Very Large Scale Integration (VLSI) has made a dramatic impact on the growth of integrated circuit technology. The positive improvements have resulted in significant performance/cost advantages in VLSI systems. As we know, to human decimal numbers are easy to comprehend and implement for performing arithmetic. Binary adders are one of the most essential logic elements within a digital system. Therefore, binary addition is essential that any improvement in binary addition can result in a performance boost for any computing system and, help improve the performance of the system. The major problem for binary addition is the carry chain. As the width of the input operand increases, the length of the carry chain increases. In this paper two architectures for modulo addition is designed and is verified using Xilinx. The main goal is to improve the performance of the system in terms of area ,speed, power etc. So that a modified FFT processor can also be designed using the above modified modulo adders.

The concept of the modulo  $2n+ 1$  adder is based on an inverted end around carry(IEAC)  $n$ - bit adder which is an adder that accepts two  $n$ -bit operands and provides a sum increased by one compared to their integer sum if their integer addition does not result in a carry output. Since the carry output depends on the carry input, a direct connection between them forms a combinational loop that may lead to an unwanted race condition .To avoid this Zimmermann [2],[3] proposed IEAC adders that make use of a parallel-prefix carry computation unit along with an extra prefix level that handles the inverted end-around carry.

In [4] it is explained that the recirculation of the inverted end around carry can be performed within the existing prefix levels, that is, in parallel with the carries' computation. In this way, the need of the extra prefix level is canceled and parallel-prefix IEAC adders are derived that can operate fast with a logic depth of  $\log_2 N$  prefix levels. Since this requires more area than [2],[3] a double parallel-prefix computation tree is required in several levels of the carry computation unit. Select-prefix and circular carry select IEAC adders proposed in [5],[6] has less area but only less operating speed

A fast Fourier transform (FFT) is an algorithm to compute the discrete Fourier transform (DFT) and its inverse. Fourier analysis converts time (or space) to frequency and vice versa; an FFT rapidly computes such transformations by factorizing the DFT matrix . As a result, fast Fourier transforms are widely used for many applications in engineering, science, and mathematics. Here an efficient FFT algorithm is also implemented.

# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 5, December 2014

## II. PARALLEL-PREFIX ADDERS

Generally parallel-prefix n-bit adder considered as a three stage circuit. They are pre-processing-stage, carry-computation-unit and post-processing-stage. 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. The preprocessing stage computes the carry-generate bits  $G_i$ , the carry-propagate bits  $P_i$ , and the half-sum bits  $H_i$ , for every  $i; 0 \leq i \leq n-1$ , according to

$$G_i = A_i . B_i ; P_i = A_i + B_i ; H_i = A_i \text{ xor } B_i$$

Where  $.$ ,  $+$ , and  $\text{xor}$  denote logical AND, OR, and exclusive OR, respectively. The second stage of the adder called the carry computation unit, computes the carry signals  $C_i$ , for  $0 \leq i \leq n-1$  using the carry generate and carry propagate bits  $G_i$  and  $P_i$ . The third stage computes the sum bits according to

$$S_i = H_i \text{ xor } C_{i-1}$$

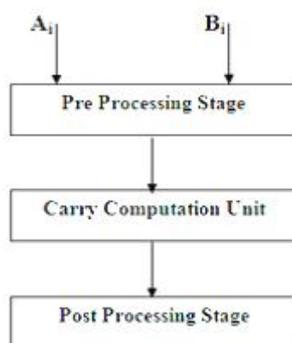


Fig.1 Parallel prefix addition basics

### A. PRE PROCESSING STAGE

The preprocessing stage computes the carry-generate bits  $G_i$ , the carry-propagate bits  $P_i$ , and the half-sum bits  $H_i$ , for every  $i; 0 \leq i \leq n-1$ , according to

$$G_i = A_i . B_i ; P_i = A_i + B_i ; H_i = A_i \text{ xor } B_i$$

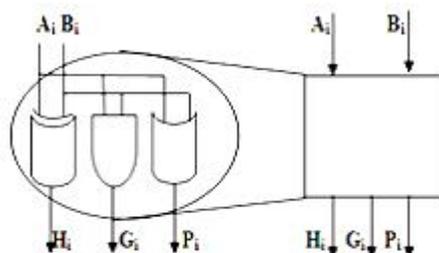


Fig.2 Pre processing stage

### B. CARRY COMPUTATION UNIT

The second stage of the adder, hereafter called the carry computation unit, computes the carry signals  $C_i$ , for  $0 \leq i \leq n-1$  using the carry generate and carry propagate bits  $G_i$  and  $P_i$ . The third stage computes the sum bits according to

$$S_i = H_i \text{ xor } C_{i-1}$$

# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 5, December 2014

Carry computation is done by using an operator called parallel prefix operator i.e, ‘dot’ operator, which associates pairs of generate and propagate signals and was defined as

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

In a series of associations of consecutive generate/propagate pairs(G,P), the notation  $(G_{k:j}; P_{k:j})$ , with  $k > j$ , is used to denote the group generate/propagate term produced out of bits  $k; k -1; \dots ; j$

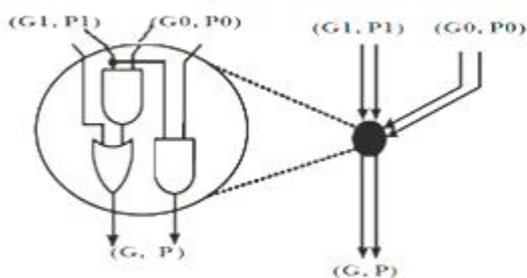


Fig.3. Carry computation unit

## C. POST PROCESSING STAGE

The third stage computes the sum bits according to

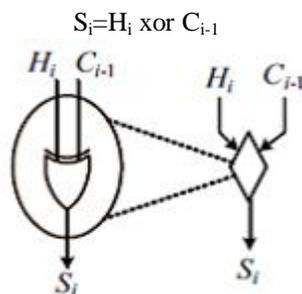


Fig.4 Post processing stage

Based on these concepts three types of parallel prefix adders are designed

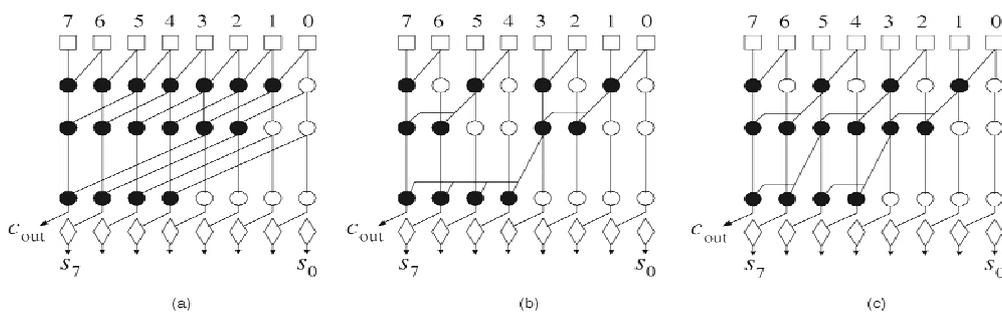


Fig.5 Examples of 8-bit parallel-prefix adders. (a) Kogge-Stone [7], (b) Ladner-Fischer [8] and (c) Knowles [9] family of adders.



# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 5, December 2014

## B. Modulo $2^n+1$ adders

There are two methods for finding modulo  $2^n + 1$  sum. First one is from simple modifications of the modulo  $2^n - 1$  adders. The second one is by the sparse approach and the introduction of a new prefix operator.

### FIRST METHOD

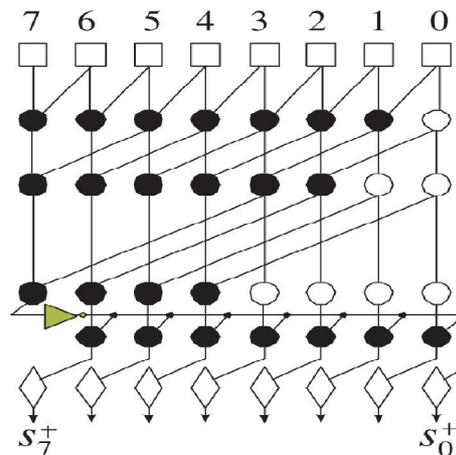


Fig .7. Parallel prefix modulo  $2^8 + 1$  adders

Similar to modulo  $2^n-1$  case, the carry  $C_i^+$  at the  $i^{\text{th}}$  bit position of an IEAC adder, when feeding the carry input  $C_{in} = C_{-1}^+$  with the inverted carry out  $\overline{C_{n-1}} = \overline{G_{n-1}:0}$  can be computed more simply by

$$C_i^+ = G_{i:0} + P_{i:0} \cdot \overline{G_{n-1}:i+1}$$

Which can be expressed as

$$C_i^+ = (G_i, P_i) \circ \dots \circ (G_0, P_0) \circ \overline{(G_{n-1}, P_{n-1}) \circ \dots \circ (G_{i+1}, P_{i+1})}$$

where  $(\overline{g}, \overline{p})$  is equal to  $(\overline{g}, p)$ , and the final sum bits are equal to  $H_i \text{ xor } C_{i-1}^+$

### SECOND METHOD

In this section, we focus on the design of diminished-1 modulo adders with a sparse parallel-prefix carry computation stage [1] that can use the same carry-select blocks as the sparse integer 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. The sum of a diminished-1 modulo adder is derived according to the following cases:

1. When none of the input operands is zero their number parts  $A^*$  and  $B^*$  are added modulo  $2^n + 1$ . This operation can be handled by an IEAC adder.
2. When one of the two inputs is zero the result is equal to the nonzero operand.
3. When both operands are zero, the result is zero.

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 number part of the diminished-1 sum is derived by the number parts  $A^*$  and  $B^*$  of the input operands as follows:

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

# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 5, December 2014

Based on [1] different architectures for modulo  $2^{16}+1$ adders are designed .According to the inverted circular idem potency property

$$(G_{i:0}, P_{i:0}) \circ \overline{(G_{n-1:i+1}, P_{n-1:i+1}) \circ (G_{i:0}, P_{i:0})} = (G_{i:0}, P_{i:0}) \circ \overline{(G_{n-1:i+1}, P_{n-1:i+1})} \dots(4)$$

With  $\overline{(G, P)} = (\bar{G}, \bar{P})$ .

Figure (8) shows sparse-4 modulo  $2^{16}+1$  adders (a)using doubled up operators and (b) using the sparse approach which is enabled by the inverted circular idem potency property. In first case the computation can be performed within  $\log_2 N$  logic levels. Here some prefix operators are doubled up, since two 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 operand's width is not a power of two. Although the sparse version of the parallel-prefix adders has a lot of regularity and the area-overhead problem, as it can be verified , there is still a lot of space for improvement.

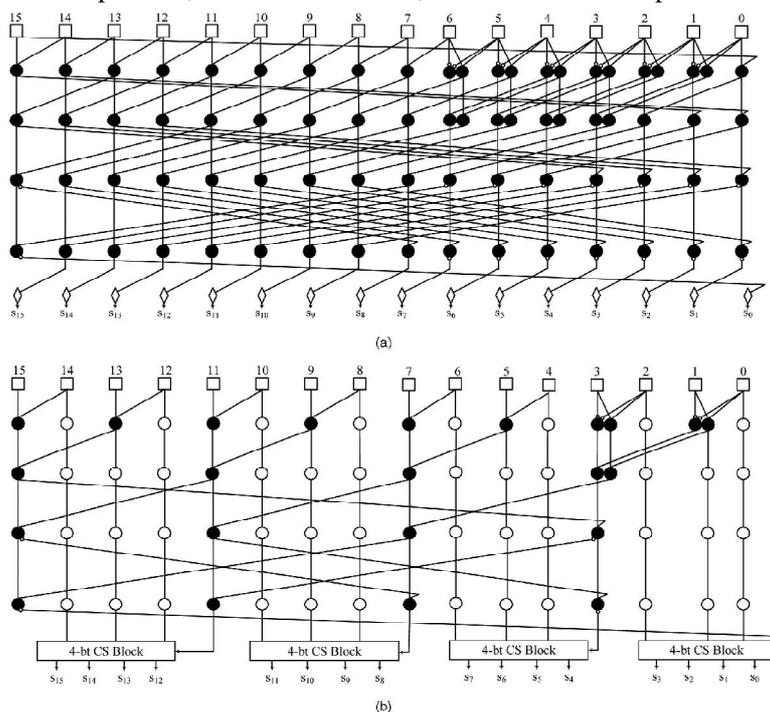


Fig.8. Modulo  $2^{16} + 1$  diminished-1 adders(a) existing and (b) using a sparse carry computation unit

To avoid this problem a new prefix operator called gray operator is introduced. The gray operator accepts five inputs and produces four outputs. Three of the inputs of a gray operator residing at prefix level  $j - 1$ , namely,  $G^{j-1}_v$ ,  $P^{j-1}_v$  and  $T^{j-1}_v$  form the operator's vertical input bus, while the rest two  $G^{j-1}_L$  and  $P^{j-1}_L$  form its lateral input bus. The lateral bus signals are driven inverted to the operator. The gray operator produces three signals for its vertical successor of prefix level  $j$  ( $G^j_v$ ,  $P^j_v$  and  $T^j_v$ ) and one ( $c_j$ ) for its lateral successor.

Based on [1],it is clear that starting from a sparse architecture with doubled up operators, it suffices to

1. remove the doubled up operators that associate inverted signals,
2. replace the top operator of every column excluding the leftmost that accepts a feedback signal with a gray one.
3. replace every vertical successor of a gray operator introduced by the previous step with a gray one.

# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 5, December 2014

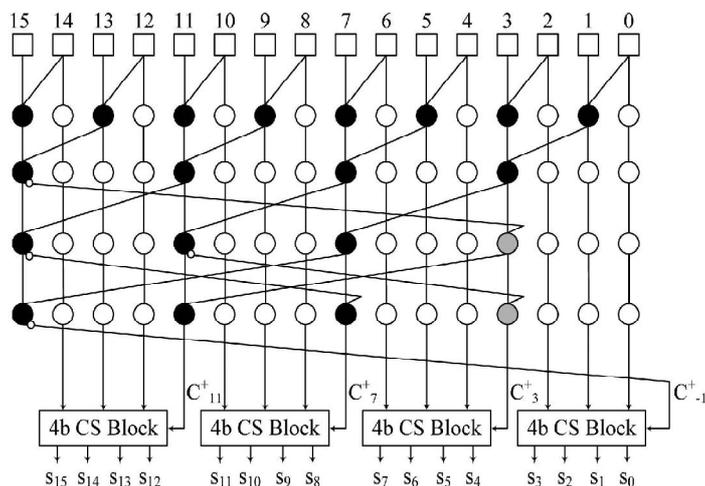


Fig.9. Proposed [1] sparse-4 modulo  $2^{16} + 1$  diminished-1 adder

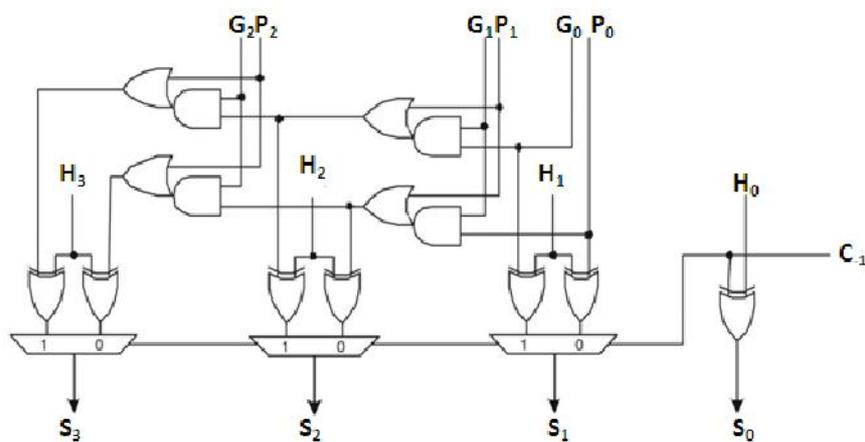


Fig. 10. Design of a 4-bit Carry Select Block

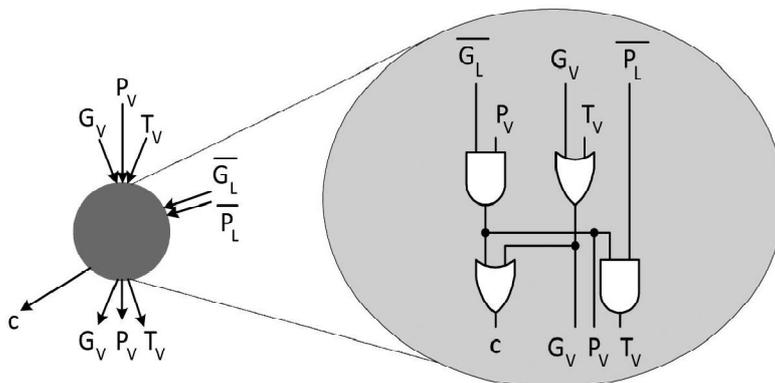


Fig. 11. Design of a gray operator

# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 5, December 2014

By using this proposed sparse-4 modulo Diminished-1 adder an efficient FFT algorithm is also implemented. A fast Fourier transform (FFT) is an algorithm to compute the discrete Fourier transform (DFT) and its inverse. Fourier analysis converts time (or space) to frequency and vice versa; an FFT rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors.

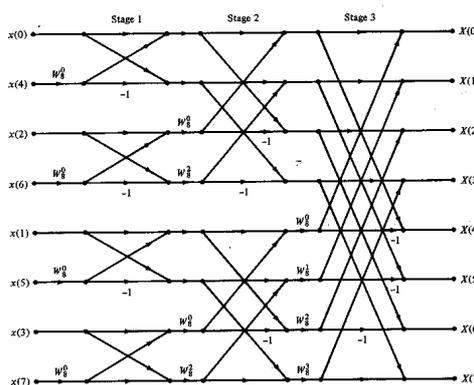


Fig .12. 8-point dft fft algorithm

Let us consider the computation of the  $N = 2^v$  point DFT. We split the  $N$ -point data sequence into two  $N/2$ -point data sequences  $f_1(n)$  and  $f_2(n)$ , corresponding to the even-numbered and odd-numbered samples of  $x(n)$ , that is,

$$F_1(n) = x(2n)$$

$$F_2(n) = x(2n+1), \quad n=0,1,\dots,N/2-1$$

Now the  $N$ -point DFT can be expressed in terms of the DFT's of the decimated sequences as follows

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad k = 0,1,\dots,N-1$$

$$= \sum_{n \text{ even}} x(n) W_N^{kn} + \sum_{n \text{ odd}} x(n) W_N^{kn}$$

$$= \sum_{m=0}^{(N/2)-1} x(2m) W_N^{2mk} + \sum_{m=0}^{(N/2)-1} x(2m+1) W_N^{k(2m+1)}$$

Hence the sequence  $X(k)$  will be obtained. Here we are considering the 8-point dft using the modified modulo  $2^n+1$  adder. The Inputs are given in parallel. The advantage of using this is power and area reduction. But the hardware requirement is high. So in order to reduce this pipelining concept is introduced in which the critical path is reduced by placing delay elements between the registers. It also helps to increase the speed.

## IV. RESULT ANALYSIS

The simulation is performed using XILINX in verilog HDL. The figure below shows the experimental results after the simulation.

TABLE 1  
Experimental Results for Parallel Prefix Adders

Performance parameters	Koggestone adder	Ladner-Fischer adder	Knowles adder
Delay(ns)	11.071	10.023	12.103
Power(mw)	0.159	0.081	0.052



# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 5, December 2014

**TABLE II**  
**Experimental Results for modulo  $2^{16}+1$  adders**

Performance parameters	Modulo $2^{16}+1$ adder existing	Sparse-4 Modulo $2^{16}+1$ adder
Delay(ns)	11.206	11.206
Power(mw)	0.20	0.159

## A. RTL SCHEMATIC of 8-POINT FFT PROCESSOR

Here 8 inputs are given and correspondingly there will be 8 outputs also. Corresponding RTL schematic is given below.



Fig.13. RTL schematic

## B. Output waveform

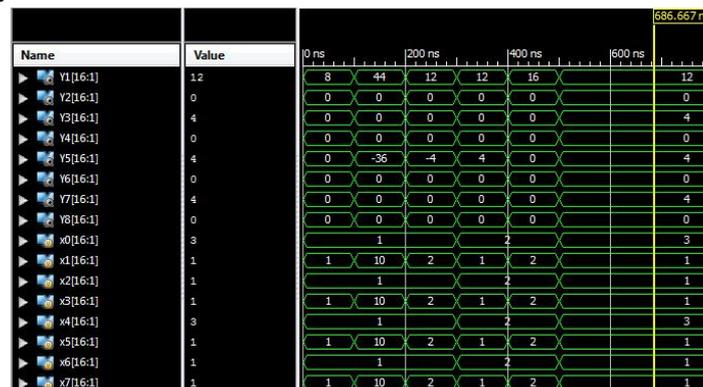


Fig .14. output waveform

## V. CONCLUSION

In this paper, two modified power efficient modulo  $2^n + 1$  adders are presented. A novel architecture has been proposed that uses the inverted circular idempotency property of the parallel-prefix carry operator in modulo  $2^n+1$  addition 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. Also an efficient 8-point FFT is designed using the modified modulo adders which has performance advantages in terms of power and area.



ISSN (Print) : 2320 – 3765  
ISSN (Online): 2278 – 8875

# International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 5, December 2014

## REFERENCES

- [1] Haridimos T. Vergos and Giorgos Dimitrakopoulos "On Modulo  $2^n+1$  Adder Design", IEEE TRANSACTIONS ON COMPUTERS, VOL. 61, NO. 2, FEBRUARY 2012
- [2] R. Zimmermann, "Binary Adder Architectures for Cell-Based VLSI and Their Synthesis," PhD dissertation, Swiss Fed. Inst. Of Technology, 1997.
- [3] R. Zimmerman, "Efficient VLSI Implementation of Modulo  $2n \pm 1$  Addition and Multiplication," Proc. 14th IEEE Symp. Computer Arithmetic, pp. 158-167, Apr. 1999.
- [4] H.T. Vergos, C. Efstathiou, and D. Nikolos, "Diminished-One Modulo  $2^n+1$  Adder Design," IEEE Trans. Computers, vol. 51, no. 12, pp. 1389-1399, Dec. 2002.
- [5] C. Efstathiou, H.T. Vergos, and D. Nikolos, "Modulo  $2n \pm 1$  Adder Design Using Select Prefix Blocks," IEEE Trans. Computers, vol. 52, no. 11, pp. 1399-1406, Nov. 2003.
- [6] S.-H. Lin and M.-H. Sheu, "VLSI Design of Diminished-One Modulo  $2^n+1$  Adder Using Circular Carry Selection," IEEE Trans. Circuits and Systems II, vol. 55, no. 9, pp. 897-901, Sept. 2008.
- [7] P.M. Kogge and H.S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," IEEE Trans. Computers, vol. C-22, no. 8, pp. 786-792, Aug. 1973.
- [8] R.E. Ladner and M.J. Fischer, "Parallel Prefix Computation," J. ACM, vol. 27, no. 4, pp. 831-838, 1980. [9] S. Knowles, "A Family of Adders," Proc. 14th IEEE Symp. Computer Arithmetic, pp. 30-34, 1999.
- [10] G. Dimtra Kopolous, D.G. Nikolos, D. Nikolos, H.T. Vergos, and C. Efstathiou, 'New Architectures for Modulo  $2n-1$  Adders,' proc. IEEE Int' l Conf. Electronics, Circuits, and Systems, 2008.
- [11] [ijcsi.org/papers/IJCSI-10-2-3-430-440.pdf](http://ijcsi.org/papers/IJCSI-10-2-3-430-440.pdf).

## BIOGRAPHY

Fathima Nishah received degree of B.Tech in Electronics and communication and presently working towards masters in Applied Electronics, now doing master thesis work in VLSI, from ILLAHIA college of engineering and Technology, Muvattupuzha, Ernakulam, Kerala.

Ruksana Maitheen., received degree of B Tech, M.Tech, Assistant Professor at ILLAHIA college of engineering and Technology, Muvattupuzha, Ernakulam, Kerala.