International Journal of Innovation and Scientific Research
ISSN 2351-8014 Vol. 11 No. 1 Oct. 2014, pp. 117-125

http://www.ijisr.issr-journals.org/

Design of FFT Processor using Modified Modulo 2^n+1 Adder

Fathima Nishah P[1] and Ruksana Maitheen[2]

[1]Applied Electronics,
Ilahia college of engineering and Technology,
Ernakulam, Kerala, India

[2]Electronics and Communication,
Ilahia college of engineering and Technology,
Ernakulam, Kerala, India

ABSTRACT: In this paper we present two different architectures for modulo 2^n+1 adder 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 2^n-1 adders with minor hardware overhead. By using this adder we can implement FFT processor with improved performance.

KEYWORDS: Parallel prefix carry computation, Modulo addition, Diminished-1 addition, inverted circular idempotency, IEAC adder.

1 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 implement for performing arithmetic operations. Binary adders are one of the most essential logic elements in a digital system. Therefore, binary addition is essential and any improvement in binary addition can result in improved 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. Using this a modified FFT processor is also designed using the above modified modulo adders.

The concept of the modulo 2^n+ 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 input and output forms a combinational loop which leads 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.

Corresponding Author: Fathima Nishah P

A fast Fourier transform (FFT) is an algorithm to compute the discrete Fourier transform (DFT) and it's inverse. Fourier analysis converts time (or space) to frequency and vice versa, 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.

2 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 Pi, and the half-sum bits H_i, for every i; 0 <= i <=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 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 <=i <=n -1 using the carry generate and carry propagate bits G_i and P_i. The third stage computes the sum bits according to

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

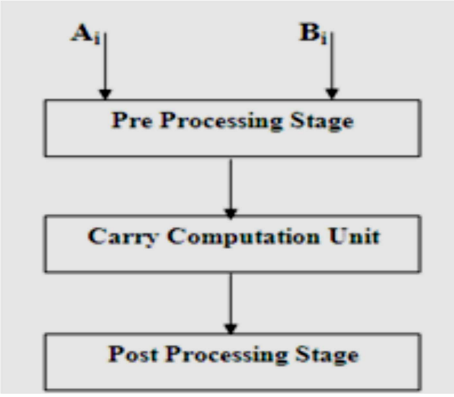


Fig. 1. Parallel prefix addition basics

2.1 PRE-PROCESSING STAGE

The preprocessing stage computes the carry-generate bits G_i, the carry-propagate bits Pi, and the half-sum bits H_i, for every i; 0 <= i <=n-1, according to

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

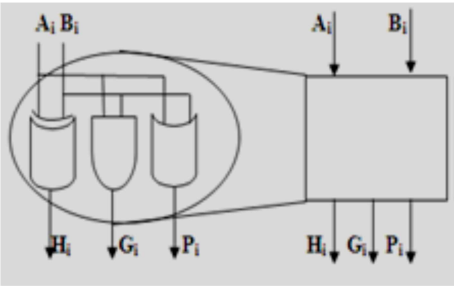


Fig.2. Pre processing stage

2.2 CARRY COMPUTATION UNIT

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

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

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 is defined as

$$(G,P)o(G',P')=(G+P.G',P.P').$$

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; . . ; J.

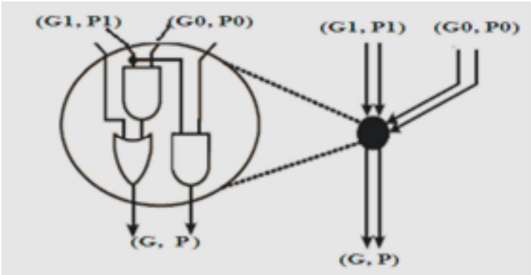


Fig.3. Carry computation unit

2.3 POST PROCESSING STAGE

The third stage computes the sum bits according to

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

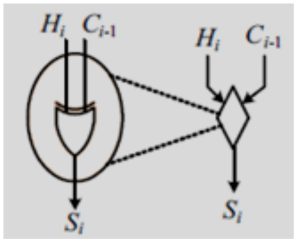


Fig.4. Post processing stage

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

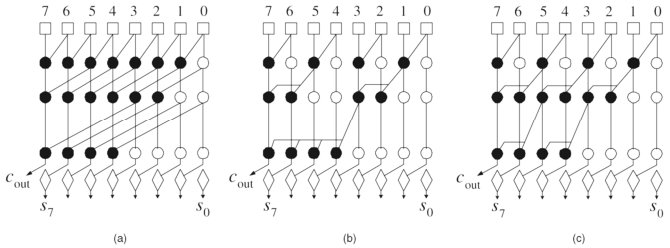


Fig.5. Examples of 8-bit parallel-prefix adders. (a) Kogge-Stone [7], (b) Ladner-Fischer [8] and (c) Knowles [9] family of adders

3 MODULO 2^N±1 ADDERS

3.1 MODULO 2^N–1 ADDERS

The modulo 2^n- 1 addition is defined as

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

A modulo $2^n - 1$ adder can be implemented using an integer adder that increments also its sum when the carry output is one. (when $A + B \geq 2^n$). The conditional increment can be implemented by an additional carry increment stage as shown in fig. 6. In this case, one extra level of '•' cells driven by the carry output of the adder, is required. When $A + B = 2^n + 1$, the adder may produce an all 1s output vector, in place of zero. In most applications, this is the second representation for zero.

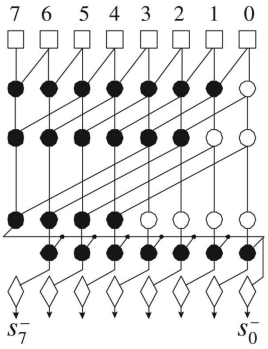


Fig .6. Parallel prefix modulo 2^8 -1 adder

The implementation of a modulo $2^n - 1$ adder requires the connection of the carry output $C_{n-1} = G_{n-1:0}$ of an integer adder to its carry-input port. The carries of the modulo $2^n - 1$ adder is given by $C_i^- = G_{i:0} + P_{i:0} . C_{in}$. Therefore, connecting the carry output to the carry input leads to $C_i^- = G_{i:0} + P_{i:0} . G_{n-1:0}$. This relation can be simplified to

$$C_i^- = G_{i:0} + P_{i:0} . G_{n-1:i+1} \quad (2)$$

The simpler equation can be equivalently expressed using the ◦ operator as follows

$$C_i^- = (G_i, P_i) \circ \ldots (G_0, P_0) \circ (G_{n-1}, P_{n-1}) \circ \ldots \circ (G_{i+1}, P_{i+1}) \quad (3)$$

The above equation (3) that computes the modulo $2^n - 1$ carries has a cyclic form and the number of generate and propagate pairs (G_i, P_i) associated for each carry is n. This means that the parallel-prefix carry computation unit of a modulo 2^n -1 adder has significantly increased area complexity than that of a corresponding integer adder. In terms of delay, the carries C- can be computed in $\log_2$ n levels using regular parallel-prefix structures using end around technique. The final sum bits

S_i^- are equal to Hi xor C_{i-1}^-.

3.2 MODULO 2^N+1 ADDERS

- First method

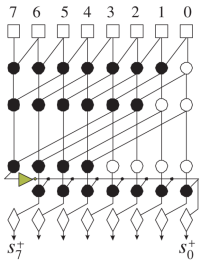


Fig .7. Parallel prefix modulo 2^8 +1 adders

First method is obtained by giving slight modification to modulo 2^n-1 adders. Similar to modulo 2^n-1 case, the carry C_i^+ at the i^{th} bit position of an IEAC adder, when feeding the carry input $C_{in} = C_{-1}^+$ with the inverted carry out $\overline{Cn-1} = \overline{Gn-1:0}$ can be computed as

$$C_i^+ = G_{i:0} + P_{i:0} \,.\, \overline{Gn-1:i+1}$$

Which is expressed as

$$C_i^+ = (G_i, P_i) \circ \ldots (G_0, P_0) \circ \overline{(Gn-1, Pn-1)\ o \ldots o\ (Gi+1, Pi+1)}$$

where $(\overline{g,p})$ is equal to $\overline{(g,p)}$, and the final sum bits are equal to H_i 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. 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.
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.

So 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. The result is given as

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

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

$$\begin{aligned} (G_{i:0}, P_{i:0}) \circ \overline{(Gn-1:i+1, Pn-1:i+1)o(Gi:0, Pi:0)} \\ = (G_{i:0}, P_{i:0}) \circ \overline{(Gn-1:i+1, Pn-1:i+1)} \end{aligned} \quad (4)$$

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

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. Although the sparse version of the parallel-prefix adders has a lot of regularity and the area-overhead problem there is still a lot of space for improvement.

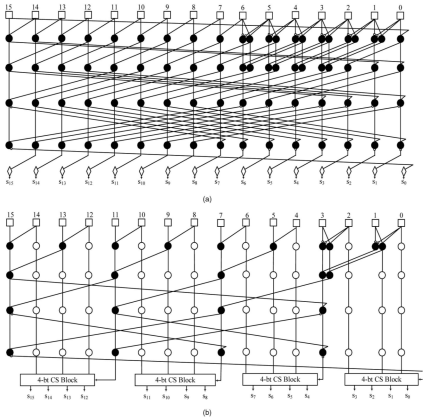


Fig.8. Modulo 2^{16} + 1 diminished-1 adders (a) existing and (b) using a sparse carry computation unit

To avoid this problem a new prefix operator called gray operator is introduced. 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, 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 level j ($G^{j}{}_{V}$; $P^{j}{}_{V}$ and $T^{j}{}_{V}$) and one (c_j) for its lateral successor. Compared to the 'o' prefix operator, the gray operator requires one extra gate but no extra logic levels.

Based on [1], by using gray operator,

1. Doubled up operators that associate inverted signals can be removed,
2. We can replace the top operator of every column excluding the leftmost that accepts a feedback signal with a gray operator, where T_V input is tied to zero
3. Replace every vertical successor of a gray operator in the previous step with a gray one.

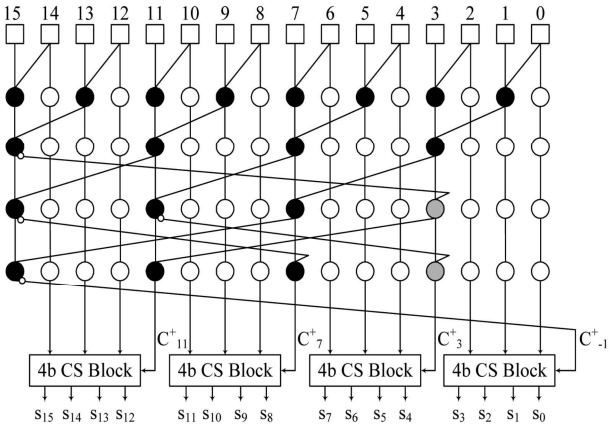


Fig.9. Proposed [1] sparse-4 modulo 2^{16} +1 diminished-1 adder.

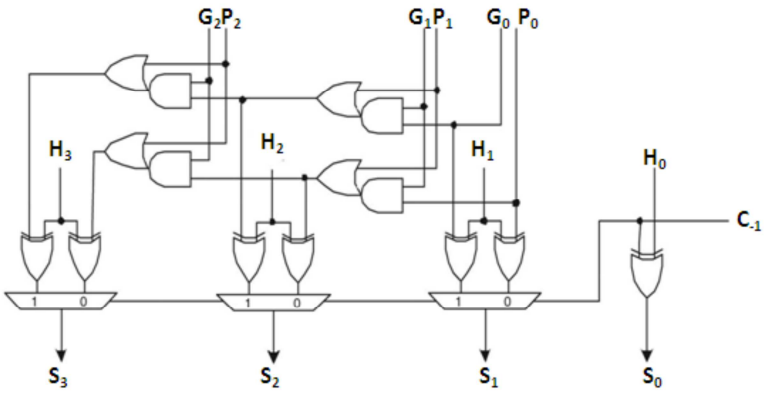


Fig.10. Design of a 4-bit Carry Select Block

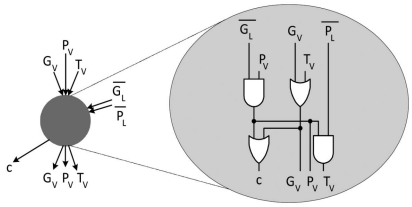


Fig.11. Design of a gray operator

By using this proposed sparse-4 modulo Diminished-1 adder an efficient radix-2 FFT algorithm is also implemented. A fast Fourier transform (FFT) is an algorithm to compute the discrete Fourier transform (DFT) and it's 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.

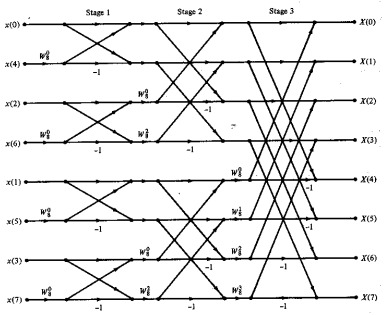


Fig.12. 8-point DFT FFT algorithm

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,

$$F1(n)=x(2n)$$

$$F2(n)=x(2n+1), \quad n=0,1,.....N/2 -1$$

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

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad k = 0,1,\ldots, N-1$$
$$= \sum_{n \text{ even}} x(n) W_N^{kn} + \sum_{n \text{ odd}} x(n) W_N^{kn}$$
$$= \sum_{m=0}^{(N/2)-1} x(2m) W_N^{2mk} + \sum_{m=0}^{(N/2)-1} x(2m+1) W_N^{k(2m+1)}$$

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.

4 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

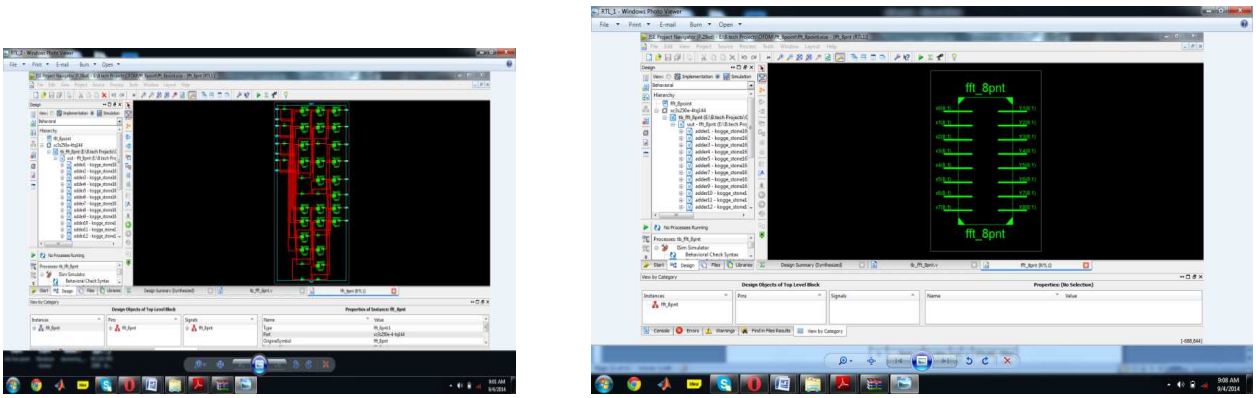
	Koggestone adder	Ladner-fischer adder	Knowles adder
Delay(ns)	2.951	2.396	13.572
Memory Usage(kb)	316132	317092	199496
Power(mw)	0.067	0.066	0.52

Table 2. Experimental Results for modulo 2^{16}+ 1 adders

	Existing modulo 2^{16}+1 adder	Modulo 2^{16}+ 1 adder using sparse carry computation	Proposed sparse-4 modulo 2^{16} +1 diminished-1 adder
Delay(ns)	4.387	3.868	3.301
Memory Usage(kb)	318372	234916	144816
Power(mw)	0.068	0.066	0.042

4.1 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.



(a) (b)

Fig.13. RTL schematic

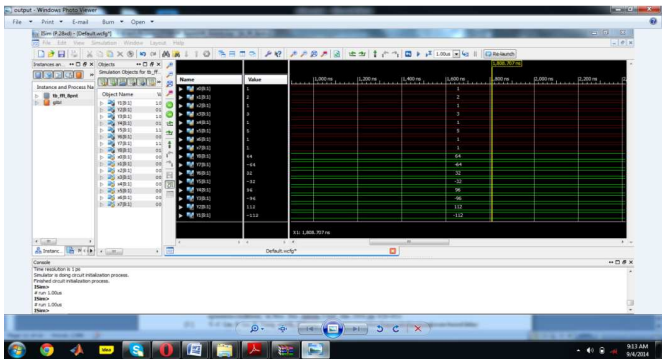


Fig .14. output waveform

5 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.

ACKNOWLEDGMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to Mrs. Ruksana Maitheen for her guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards my parents & project coordinator Mrs. Lekshmi M S (Ilahia college of engineering and Technology) for their kind co-operation and encouragement which help me in completion of this project.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

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 2^n ±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 2^n±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. DimtraKopulous, D.G. Nikolos, D. Nikolos, H.T. Vergos, and C. Efstathiou,' New Architectures for Modulo 2^n-1 Adders," *proc. IEEE Int'1 Conf. Electronics, Circuits, and Systems*, 2008.