ISSN 2319 – 2518 www.ijeetc.com
Vol. 2, No. 4, October 2013
© 2013 IJEETC. All Rights Reserved

Research Paper

MODULO $2^n + 1$ MAC UNIT

Sithara Sha[1]* and Shajimon K John[1]

*Corresponding Author: Sithara Sha, ✉ sitharash786@gmail.com

Modulo $2^n + 1$ arithmetic has a variety of applications in several fields like cryptography, pseudorandom number generation, eliminating round-off errors in convolution computations, etc. Residue Number System (RNS) represents a number in the range of $[0, 2^n]$ using n + 1 bits. 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 speed up 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. The complexity of modulo $2^n + 1$ arithmetic operation such as addition and multiplication can be reduced by designing an efficient modulo $2^n + 1$ adder and modulo $2^n + 1$ multiplier. Both the addition and multiplication are combined together to form a multiply and accumulate (MAC) unit. Diminished-one number representation is employed in this MAC unit. In the diminished-one number system, each number X is represented by $X^* = X - 1$ and the representation of 0 is treated in a special way. Therefore, diminished-one modulo $2^n + 1$ circuits require only n bits for their number representations. An efficient modulo $2^n + 1$ MAC unit is presented in this paper. A diminished-one modulo $2^n + 1$ adder using parallel-prefix tree for carry computation and modulo $2^n + 1$ multiplier based on dadda tree architecture is used for MAC architecture. The proposed MAC unit is analysed using various tools like ModelSim 10.1b for logical verification and for synthesizing Leonardo Spectrum LS 2009 a_6.

Keywords: RNS, Modulo $2^n + 1$ arithmetic; Diminished-one number representation, Modulo adder, Modulo multiplier

INTRODUCTION

Modulo arithmetic has been used in digital computing systems for various purposes for many years. In particular, modulo $2^n + 1$ arithmetic appears to play an important role in many algorithms. A first application field is in Residue Number Systems (RNS). In an RNS based application, every number X is represented by a sequence of residues (X_1, X_2, X_3, X_4, ..., X_M). The RNS is an arithmetic

[1] VLSI & Embedded Systems, ECE Department, SAINTGITS College of Engineering, Kottayam.

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 (Costas and Haridimos, 2005).

Since a number in the range of [0, 2^n] requires n + 1 bits for its representation, the weighted representation of an operand modulo $2^n + 1$ is a problem in an RNS that uses the three moduli set {$2^n – 1; 2^n; 2^n + 1$}, given that the other two channels operate on n-bit quantities. To overcome this problem, Leibowitz introduced the diminished-1 representation. In diminished-1 representation, each number is represented decremented by 1 modulo $2^n + 1$ and all arithmetic operations are inhibited for a zero operand (Haridimos *et al.*, 2002; and Costas and Haridimos, 2005). Zero is represented using a separate zero indication bit. This representation has the advantage that the numbers are represented by n bits and simplifies the basic operations of addition, multiplication, and scaling modulo $2^n + 1$.

The main objective of this paper is to design and implement a modulo $2^n + 1$ MAC (Multiply and Accumulate) unit with improved speed and less implementation area. Here diminished-1 representation of the number is used. Multiplication is performed by an efficient diminished-1 multiplier which uses dadda tree architecture to reduce the number of partial products. Since the number of partial product is reduced the speed of MAC gets increased. This speedily generated partial product are added using a diminished-1 modulo $2^n + 1$ adder which performs the addition in a faster way using parallel-prefix logic.

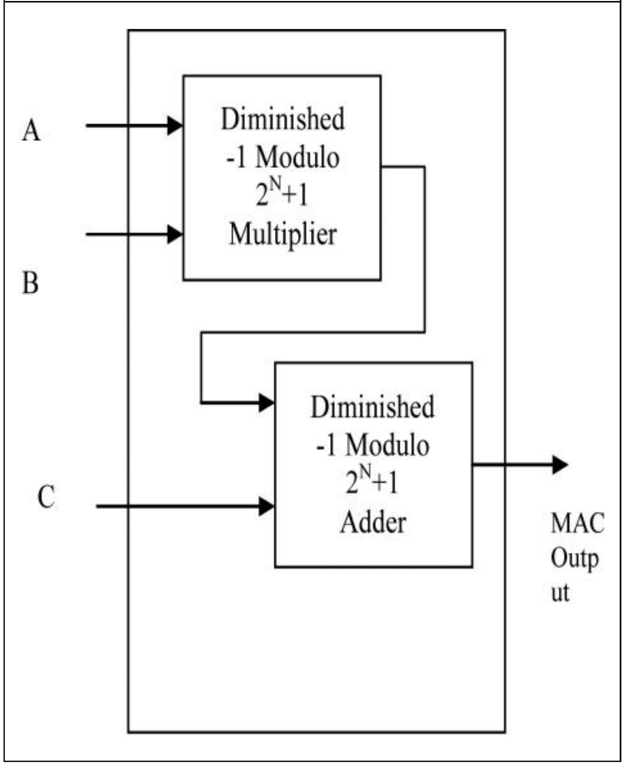
DESIGN OF MAC

For real-time signal processing, a high speed and high throughput Multiplier-Accumulator (MAC) is always a key to achieve a high performance digital signal processing system. In the last few years, the main consideration of MAC design is to enhance its speed. This is because speed and throughput rate is always the concern of digital signal processing system. A conventional MAC unit consists of (fast multiplier) multiplier and an accumulator that contains the sum of the previous consecutive products.

Arithmetic modulo $2^n + 1$ has found applicability in a variety of fields ranging from pseudorandom number generation and cryptography, up to convolution computations without round-off errors (Haridimos, 2012). The complexity of modulo $2^n + 1$ arithmetic unit is determined by the representation chosen for the input operands. Recently, the benefits of diminished-1 arithmetic have been utilized for the design of low-power convolution architectures and for high speed implementation of the IDEA cryptographic algorithm (Vergos, 2008; and Somayeh and Keivan, 2008). In every case, when performing arithmetic operations modulo $2^n + 1$ the input operands and the results are limited between 0 and 2^n. A more dense encoding of the input operands and simplified arithmetic operations modulo $2^n + 1$ are offered by the diminished-1 representation.

The proposed work mainly focuses to develop an efficient modulo MAC unit with high speed and better performance. Diminished-1 representations of the numbers are used here. The block diagram of proposed MAC unit is depicted in Figure 1.

Figure 1: Block Diagram of MAC Unit


A
B
Diminished -1 Modulo 2^N+1 Multiplier
C
Diminished -1 Modulo 2^N+1 Adder
MAC Outp ut

Diminished-1 Modulo $2^n + 1$ Multiplier

Fast multipliers are essential parts of digital signal processing systems. The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today, especially since the media processing took off. In the past multiplication was generally implemented via a sequence of addition, subtraction, and shift operations. Multiplication can be considered as a series of repeated additions. Modulo $2^n + 1$ multiplier is one of the critical components in the area of data security applications such as International Data Encryption Algorithm (IDEA), digital signal processing, and fault-tolerant systems that demand high reliability and fault tolerance. Here modulo $2^n + 1$ multiplier use diminished-1 representation because this representation is specific for the IDEA implementation and imposes all operands to be in weighted form, except the operand 2^n, which is represented as an all zeros operand (Costas and Haridimos, 2005; and Wonhak *et al.*, 2011).

The basic equations for modulo multiplication of two numbers *X* and *Y* are:

$X^* = X - 1$

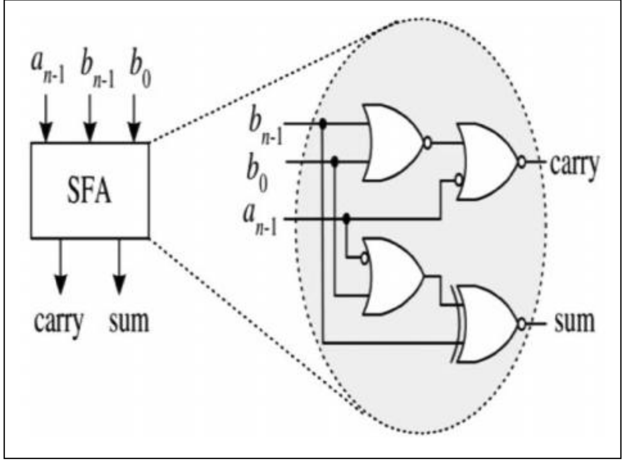
$Y^* = Y - 1$

Product $Q = Q - 1 = (X.Y) - 1$

A new architecture for modulo $2^n + 1$ multiplication for diminished-1 operands are introduced here. The derivation of the partial products and their reduction in two summands is done (Costas and Haridimos, 2005).

Additional simplifications are possible to the Dadda reduction tree. Consider the partial products $PP_0 = a_{n-1}b_0, a_{n-2}b_0, ..., a_1b_0, a_0b_0$, $PP_n = A_{-1}$, and $PP_{n+1} = B_{-1}$. If these three partial products are driven to the same FA row of the array, then each FA can be simplified significantly. Figure 2 presents a possible implementation of a block that accepts a_{n-1}, b_{n-1}, and b_0 and performs the addition of the bits $a_{n-1}b_0$, a_{n-1}, and b_{n-1}. The simplified FA is

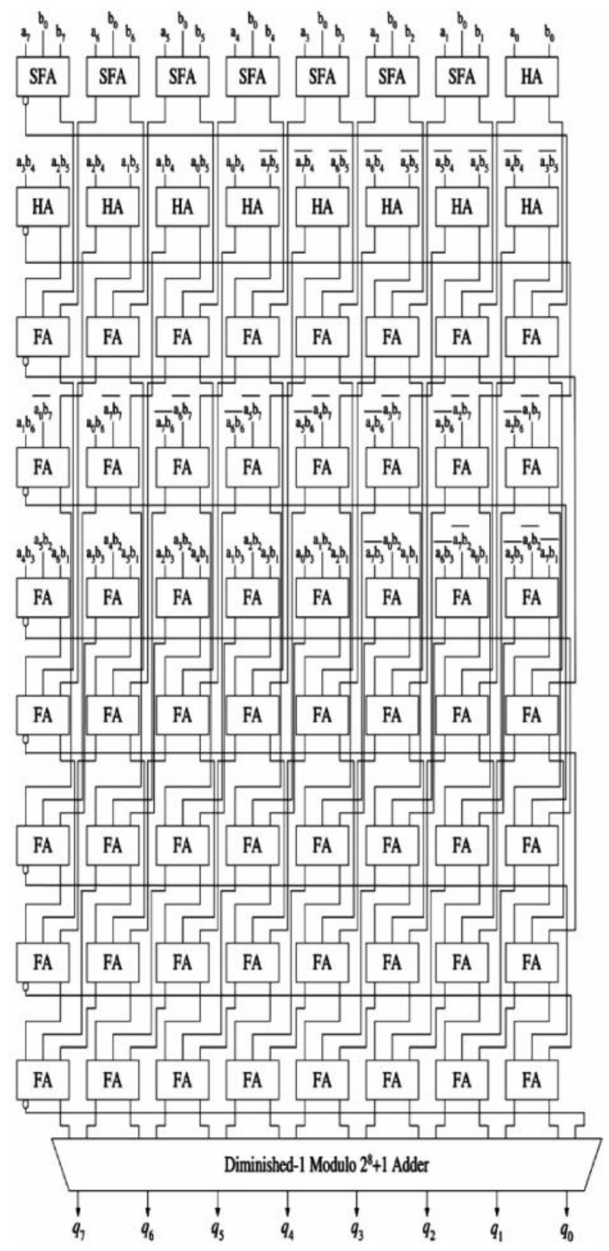
Figure 2: Simplified Full Adder (SFA) Implementation



denoted as SFA. The FA of the same row that accepts a_0b_0, a_0, and b_0 can be further simplified to an HA. Furthermore, since C-1 is the all 0s vector, the row of FAs that accepts this operand can be simplified to a row of Half-Adders (HA).

After all the analysis, we reduce multiplication into a series of reduced partial products. The partial products are reduced using dadda tree architecture. Here Full Adder (FA) architecture is regular. Since an FA row reduces the number of partial products by one, n + 1 FA rows are required in order to derive the two final summands from n + 3 partial products. The FAs at the most significant bit position will then produce n + 1 carries of weight 2^n. An implementation of the proposed architecture is composed of AND or NAND gates that form a bit of each partial product, a Dadda tree that reduces the n + 3 partial products into two summands, and a modulo $2^n + 1$ adder for diminished-1 operands that accepts these two summands and produces the required product. The final multiplier structure (Costas and Haridimos, 2005; and Reto, 1999) is depicted in Figure 3.

Figure 3: Diminished One Modulo $2^n + 1$ Multiplier for n = 8


SFA
HA
FA
Diminished-1 Modulo 2^8+1 Adder
q_7 q_6 q_5 q_4 q_3 q_2 q_1 q_0

Diminished-1 Modulo $2^n + 1$ Adder

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 (1 00 . . . 0). The sum of a diminished-1 modulo adder is derived according to the following cases:

- When none of the input operands is zero (az, $bz \neq 0$), their number parts A^* and B^* are added modulo $2^n + 1$.
- 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 (Haridimos, 2012). 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 (Ghassem, 2006; and Haridimos, 2012). When none of the input operands is zero, az, $bz \neq 1$, the number part of the diminished-1 sum S^* is derived by the number parts A^* and B^* of the input operands as follows:

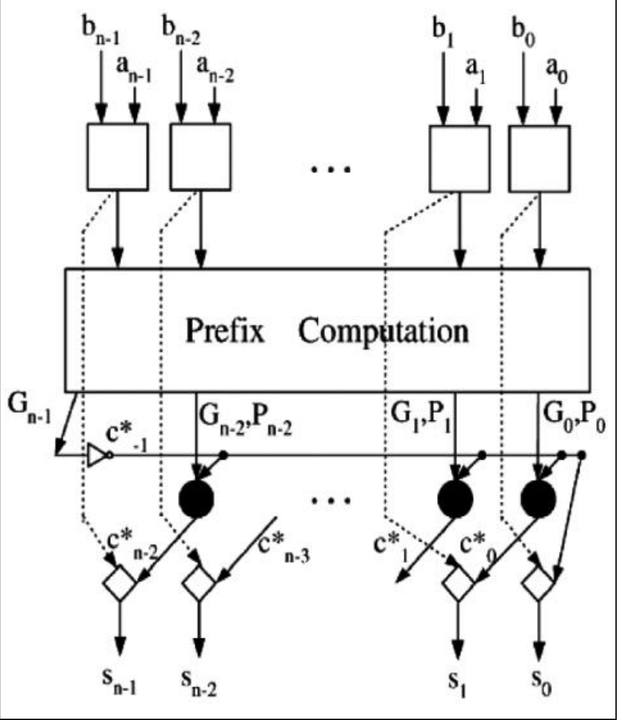
$$S^* = (A^* + B^*) \bmod (2^n + 1)$$

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

The general architecture of a modulo $2^n + 1$ diminished one adder is given in Figure 4. The carry computation is performed by parallel-prefix structure (Haridimos *et al.*, 2002; and Haridimos, 2012).

In order to speed up the addition operation, the latency of carry computation should be minimized. One solution is to use Carry Look-Ahead (CLA) adders. Unfortunately, when the operand length becomes large, a single level of CLA is insufficient since the number of inputs to the high order gates in the carry generation logic also becomes large. Since,

Figure 4: General Modulo $2^n + 1$ Diminished-1 Arithmetic Adder

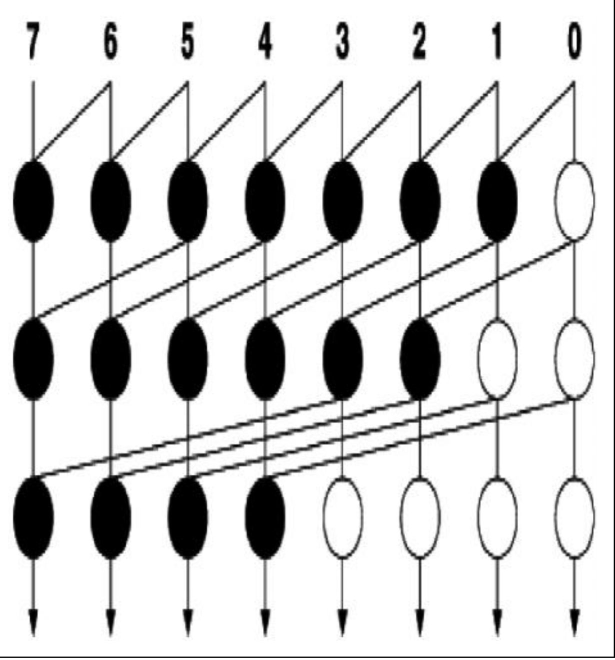


in current VLSI technology, gates with a large number of inputs are either not available or too slow, a gate with a large number of inputs is usually implemented as two or more logic levels of gates with fewer inputs. Increasing the logic levels of a circuit, however, always results in a slower circuit. This problem can be attacked by designing the adders with two or more levels of CLA (Lampros, 2000). Another approach for speeding up the carry computation problem stems from the observation that carry propagation in binary addition is a prefix problem (Lampros, 2000; Haridimos, 2002; and Costas and Haridimos, 2003),. In general, we can construct a multilevel tree of look-ahead structures to achieve delay that grows with log N. Such adders are variously referred to as tree adders or parallel prefix adders. Carry computation is transformed into a prefix computation.

The 'o' operation on a pair of (g_x, p_x) terms is usually represented as a node ● and a whole carry computation unit is represented as a tree structured interconnection of such nodes. Several tree structures have been proposed in past such as Ladner- Fischer, Kogge-Stone, etc. (Lampros, 2000; Haridimos *et al.*, 2002; Somayeh and Keivan, 2008; and Haridimos, 2012). The insertion of the buffering nodes ○ is not mandatory. The adders that result following one of the proposed tree structures feature regular layout, but each structure has distinct implementation area, speed, and fan-out characteristics.

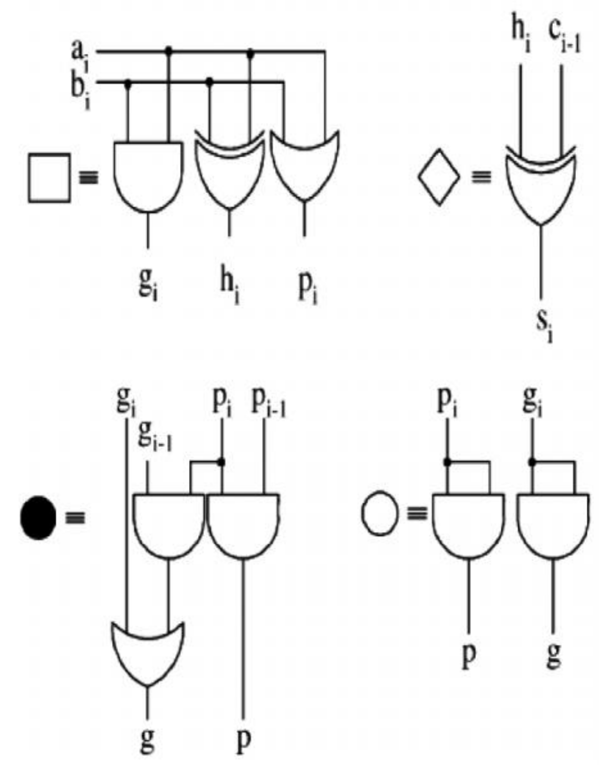
Here Kogge-Stone prefix tree structure is used to implement the carry computation unit of our modulo $2^n + 1$ diminished-1 adder. Adders with a Kogge-Stone prefix structure are faster and require only minimum implementation area. Figure 5 represent the kogge-stone prefix structure.

Figure 5: Kogge-Stone Prefix Structure for n = 8



In this paper, Diminished one modulo $2^n + 1$ adder is implemented by using the general architecture along with kogge-stone parallel prefix structure for performing carry computation. The gate level implementation of each node in the final adder is given in Figure 6.

Figure 6: Gate Level Implementation of Different Nodes



SIMULATION AND SYNTHESIS RESULTS

Simulation

Modulo $2^n + 1$ adder, Modulo $2^n + 1$ multiplier and proposed Modulo $2^n + 1$ MAC unit in this paper are simulated using Modelsim. ModelSim provides an Integrated Debug Environment that facilitates efficient design debug for SoC and FPGA based designs. For reference 8 bit adder, 8 bit multiplier and 8×8 MAC unit are considered here.

Figure 7 depicts the simulation result of an 8 bit Modulo $2^n + 1$ adder. Parallel prefix tree structure used for carry computation enhances the speed of addition.

The simulation result of Modulo $2^n + 1$ multiplier Figure 8. Dadda tree structure is used for partial product reduction and thereby reduces the implementation area.

The simulation result of Modulo $2^n + 1$ MAC unit is given in Figure 9. Here diminished-one representation of numbers is taken.

Figure 7: Simulation Result of Diminished-1 Modulo $2^n + 1$ Adder for n = 8

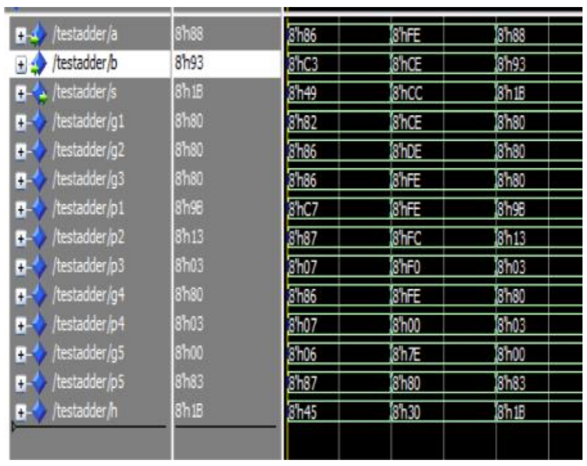

/testadder/a 8'h88 8'h86 8'hFE 8'h88
/testadder/b 8'h93 8'hC3 8'hCE 8'h93
/testadder/s 8'h1B 8'h49 8'hCC 8'h1B
/testadder/g1 8'h80 8'h82 8'hCE 8'h80
/testadder/g2 8'h80 8'h86 8'hDE 8'h80
/testadder/g3 8'h80 8'h86 8'hFE 8'h80
/testadder/p1 8'h9B 8'hC7 8'hFE 8'h9B
/testadder/p2 8'h13 8'h87 8'hFC 8'h13
/testadder/p3 8'h03 8'h07 8'hF0 8'h03
/testadder/g4 8'h80 8'h86 8'hFE 8'h80
/testadder/p4 8'h03 8'h07 8'h00 8'h03
/testadder/g5 8'h00 8'h06 8'h7E 8'h00
/testadder/p5 8'h83 8'h87 8'h80 8'h83
/testadder/h 8'h1B 8'h45 8'h30 8'h1B

Figure 8: Simulation Result of Diminished-1 Modulo $2^n + 1$ Multiplier for n = 8

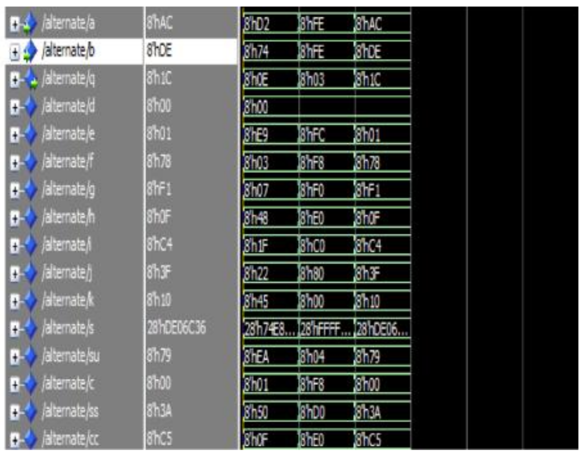
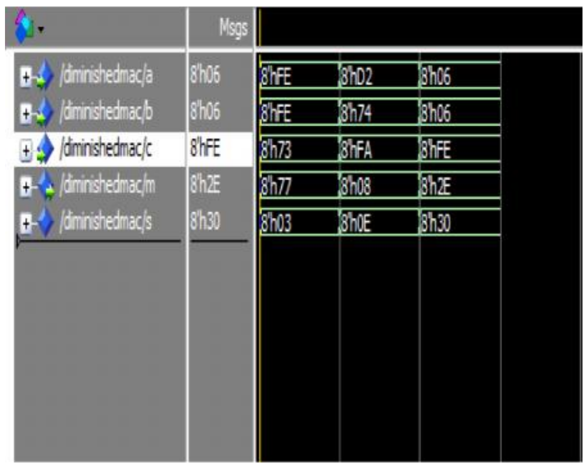

/alternate/a 8'hAC 8'hD2 8'hFE 8'hAC
/alternate/b 8'hDE 8'h74 8'hFE 8'hDE
/alternate/q 8'h1C 8'h0E 8'h03 8'h1C
/alternate/d 8'h00 8'h00
/alternate/e 8'h01 8'hE9 8'hFC 8'h01
/alternate/f 8'h78 8'h03 8'hF8 8'h78
/alternate/g 8'hF1 8'h07 8'hF0 8'hF1
/alternate/h 8'h0F 8'h48 8'hE0 8'h0F
/alternate/i 8'hC4 8'h1F 8'hC0 8'hC4
/alternate/j 8'h3F 8'h22 8'h80 8'h3F
/alternate/k 8'h10 8'h45 8'h00 8'h10
/alternate/s 28'hDE06C36 28'h74E8... 28'hFFFF... 28'hDE06...
/alternate/su 8'h79 8'hEA 8'h04 8'h79
/alternate/c 8'h00 8'h01 8'hF8 8'h00
/alternate/ss 8'h3A 8'h50 8'hD0 8'h3A
/alternate/cc 8'hC5 8'h0F 8'hE0 8'hC5

Figure 9: Simulation Result of Diminished-1 Modulo $2^n + 1$ MAC Unit for n = 8


Msgs
/diminishedmac/a 8'h06 8'hFE 8'hD2 8'h06
/diminishedmac/b 8'h06 8'hFE 8'h74 8'h06
/diminishedmac/c 8'hFE 8'h73 8'hFA 8'hFE
/diminishedmac/m 8'h2E 8'h77 8'h08 8'h2E
/diminishedmac/s 8'h30 8'h03 8'h0E 8'h30

Synthesis

Synthesis of modulo adder, multiplier and MAC unit is performed by using Leonardo Spectrum. Leonardo Spectrum synthesizes all levels of abstraction, and minimizes the amount of logic needed, resulting in a final net list description in the technology of your choice. Area in terms of total gate count and Delay of all the circuits are analyzed. For reference 8 bit adder, multiplier and 8×8 MAC is taken. The analysis report is given in Table 1.

Table 1: Stnthesis Results Type Area Delay

Modulo Adder	253	1.74
Modulo Multiplier	2109	7.86
Modulo Mac Unit	2362	8.75

CONCLUSION

The conventional arithmetic carries propagation based on a weighted number system is the reason for performance degradation in hardware computing systems. An 8x8 modulo $2^n + 1$ multiplier-accumulator (MAC) is presented in this work. A diminished-one modulo $2^n + 1$ adder using parallel-prefix tree for carry computation and modulo $2^n + 1$

multiplier based on dadda tree architecture is used for MAC architecture. Prefix tree structure enhances the speed of addition by reducing the time required for carry computation. Here we use Kogge-Stone prefix parallel tree for carry computation. It also features regular layout. Dadda tree architecture used in the multiplier circuit consists of a series of full adders and half adders. Hence it facilitates the realization of high-speed and low-power arithmetic units. It is widely used in applications such as digital filtering, digital signal processing, communications and cryptography.

REFERENCES

1. Costas Efstathiou and Haridimos T Vergos (2003), "Modulo $2^n \pm 1$ Adder Design Using Select-Prefix Blocks", *IEEE Transactions on Computers*, Vol. 52, No. 11, November.
2. Costas Efstathiou and Haridimos T Vergos (2005), "Efficient Diminished-1 Modulo $2^n + 1$ Multipliers", *IEEE Transactions on Computers*, Vol. 54, No. 4, April.
3. Ghassem Jaberipur (2006), "A One-Step Modulo $2^n + 1$ Adder Based on Double-lsb Representation of Residues", *The CSI Journal on Computer Science and Engineering*, Vol. 4, Nos. 2 & 4, pp. 10-16.
4. Giorgos Dimitrakopoulos and Dimitris Nikolos (2005), "High-Speed Parallel-Prefix VLSI Ling Adders", *IEEE Transactions on Computers*, Vol. 54, No. 2, February.
5. Haridimos T Vergos (2012), "On Modulo $2^n + 1$ Adder Design", *IEEE Transactions on Computers*, Vol. 61, No. 2, February.
6. Haridimos T Vergos, Costas Efstathiou and Dimitris Nikolos (2002), "Diminished-One Modulo $2^n + 1$ Adder Design", *IEEE Transactions on Computers*, Vol. 51, No. 12, December.
7. Lampros Kalampoukas, Dimitris Nikolos, Costas Efstathiou, Haridimos T Vergos and John Kalamatianos (2000), "High-Speed Parallel-Prefix Modulo $2^n – 1$ Adders", *IEEE Transactions on Computers*, Vol. 49, No. 7, July.
8. Reto Zimmermann (1999), "Efficient VLSI Implementation of Modulo ($2^n \pm 1$) Addition and Multiplication", 14th IEEE Symposium on Computer Arithmetic (ARITH 14), Adelaide, Australia.
9. Ruchi Singh and Mishra R A (2011), "Design and Simulation of Diminished-One Modulo $2^n + 1$ Adder Using Circular Carry Selection", Proceedings of the World Congress on Engineering, Vol. II, July 6-8, WCE, London, UK.
10. Somayeh Timarchi and Keivan Navi (2008), "Improved Modulo $2^n + 1$ Adder Design", *International Journal of Computer and Information Engineering*.
11. Somayeh Timarchi, Mahmood Fazlali and Sorin D Cotofana (2010), "A Unified Addition Structure for Moduli Set $\{2^n – 1, 2^n, 2^n + 1\}$ Based on a Novel RNS Representation", Computer Design (ICCD), 2010 IEEE International Conference.
12. Vergos H T and Efstathiou C (2008), "A Unifying Approach for Weighted and

Diminished-1 Modulo $2^n + 1$ Addition", *IEEE Transactions on Circuits and Systems—II: Express Briefs*, Vol. 55, No. 10, October.

13. Wonhak Hong, Rajashekhar Modugu and Minsu Choi (2011), "Efficient Online Self-Checking Modulo $2^n + 1$ Multiplier Design", *IEEE Transactions on Computers*, Vol. 60, No. 9, September.