Design and Implementation of On Modulo 2^n+1 Adder Design Using FPGA

CH Srinivasa Raoand P Pavan Kumar

Abstract: 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. Also, modulo 2^n +1 operator 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 depending on the propagation of previous carry bits, which leads to decrease the delay of leading significant bits for binary operations. RNS is well appropriate to applications that are rich of arithmetic operations and has been adopted in the design of digital signal processors, FIR filters and communication devices offering in several cases apart from enhanced operation speed, low-power characteristics. The complexity of a modulo 2^n +1 arithmetic unit will be determine by 3 depictions, they are the normal weighted one, the diminished-1 and the signed-LSB representations .Only consider the first two representations in the subsequent, since the implementation of the signed-LSB representation does not lead to more efficient circuits in delay or area terms.

By using two architectures, on modulo 2^n +1 adder has been developed. The first one is developed by using a sparse carry computation unit that computes only some of the carries of the on modulo 2^n +1 addition. This sparse approach is facilitated by the preface of the inverted circular idem potency property of the parallel-prefix carry operator and its regularity and area efficiency are improved by adding a new prefix operator. The ensuing diminished-1 adders can be developed in less area and use less power compared to all former proposal, while maintaining a high operation speed. The second one coalesce the design of modulo $2^n \pm 1$ adders. It is shown that modulo 2^n +1 adder can be easily derived by straightforward modifications of modulo 2^n- 1 adder with minor hardware overhead

Keywords: Parallel Prefix adders, Sparse Diminished-1 Adders, FPGA, Xilinx

I. INTRODUCTION

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. Also, modulo 2^n +1 operator 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 carry's, leading to significant speedup over the corresponding binary operations. RNS is well appropriate 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 devices, offering in several cases apart from enhanced operation speed, reduced power characteristics

The complexity of modulo 2^n+1 arithmetic unit is determined by the representation of the input operands. Three illustrations have been considered, the normal weighted one, the diminished-1 and the signed-LSB representations. We only consider the first two representations in the following, since the implementation of the signed-LSB representation does not lead to more efficient circuits in delay or area terms. In all the cases, when implementing 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 representation 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_zA* where a_z is a single bit, frequently called the zero indication bit, and A* is an n-bit vector, frequently 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 = 5 modulo 17 is 001002.

II. PARALLEL PREFIX ADDER

The parallel adder is the most important element used in arithmetic operations of many processors. With the rising popularity of mobile devices, low power consumption and high performance integrated circuits has been the target of recent research. However, the two design criteria are often in conflict and that improving one particular aspect of the design constrains the other. Extends from the idea of carry look ahead calculation, a class of parallel carry look-ahead schemes are formed targeting at high-performance applications. A Parallel Prefix Adder (PPA) is equivalent to the CLA adder.

The two different in the way their carry generation block is implemented. The parallel prefix carry look ahead adder was first proposed some twenty years ago as a means of accelerating n-bit addition in VLSI technology. It widely considered as the fastest adder and used for high performance arithmetic circuits in the industries. A three step process is generally involved in the construction of a Parallel Prefix Adder. The first step involves the creation of generate and propagate signals for the input operand bits. The second step involves the generation of carry signals. In the final step, the sum bits of the adder are generated with the propagate signals

of the operand bits and the preceding stage carry bit using a xor gate.

For large word lengths, the design of sparse parallel prefix adders is preferred, because the wiring and area of the design are considerably reduced without give up 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, reduction significant amount of area in the carry-computation unit

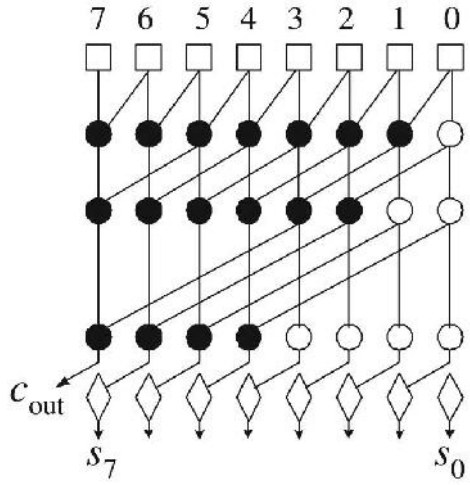


Fig 1: Parallel prefix adder structure [Kogge-Stone]

A. The Implementation of each cell in the parallel prefix adder structure

I. Square Cell

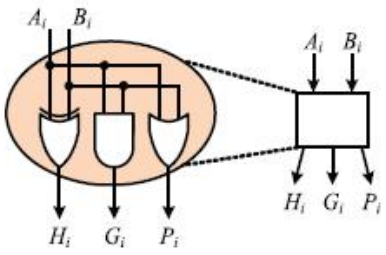


Figure 2: Square Cell

$G_i = A_i \& B_i$
$P_i = A_i + B_i$
$H_i = A_i \oplus B_i$

Here three types of signals will be generated namely carry generate (G_i), Carry propagate (P_i) and Half sum (H_i).

II. Black Cell

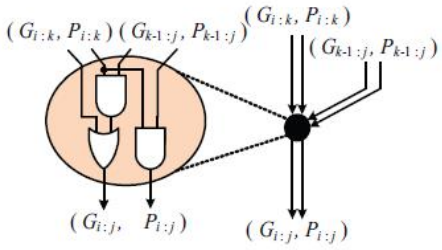


Fig 3: Black Cell

By Using G_i, P_i and H_i carries will be calculated

III. Diamond Cell

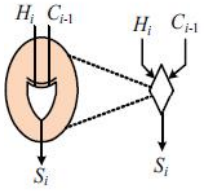


Fig 4: Diamond Cell

$S_i = H_i \oplus C_i$

The sum of each stage will be calculated by using diamond cell. If number of input binary bits increases automatically, number of carry calculations will be increased will be increased, hence area, propagation delay will increases.

To overcome that the proposed system is developed & implemented

Here carry will calculated by using

$C_{i+1}=G_i+P_iC_i$

III. MODULO 2^n+1 ADDER

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:

1. When none of the input operands is zero (a_z, b_z /=0) their number parts A and B are added modulo 2^n + 1. This operation as discussed in the following can be handled by an adder.
2. When one of the two inputs is zero the result is equal to the none-zero operand.
3. When both operands are zero, the result is zero.

In the cases 1 or 3 the result is zero, 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. As per the above, a true modulo addition in a diminished-1 adder is needed only in case 1, where as in other cases the sum is known in advance. When none of the input operands is zero (a_z, b_z /=1), the number part of the diminished-1 sum is derived by the number parts A and B of the input operands as follows:

$$s = (a^* + b^*) \text{ modulo } 2^n + 1$$
$$= \begin{cases} (a^* + b^*+1) \text{ modulo } 2^n & (a^* + b^*) < 2^n \\ (a^* + b^*) \text{ modulo } 2^n & (a^* + b^*) >= 2^n \end{cases}$$

Implementation of Parallel prefix modulo 2^8 + 1 adder

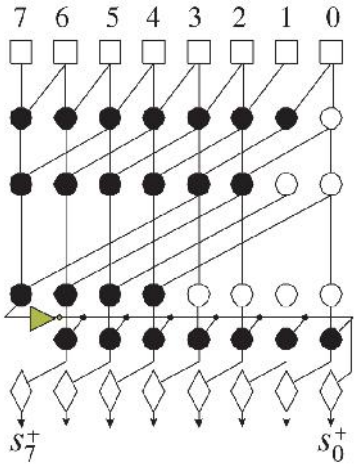


Fig 5: Parallel Prefix Modulo $2^8 + 1$ adder

IV. IMPLEMENTATION OF MODULO 2^n+1 ADDERS BY USING SPARSE CARRY COMPUTATION

The devise of diminished modulo adders with a sparse parallel-prefix carry computation stage that can use the same carry-select blocks as the sparse integer adders.

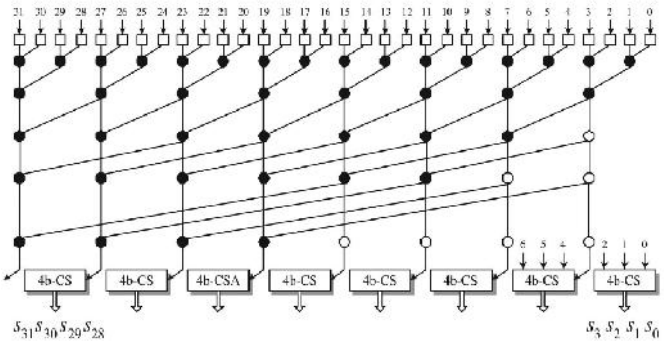


Fig 6: Sparse-4 parallel-prefix structure for a 32-bit integer adder

The logical level implementation of each CS block for computing the sums

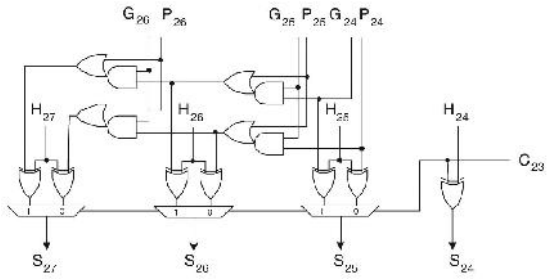


Fig 7: CS Block

The G_i, P_i and H_i in CS block are implemented by using

$G_i = A_i \,\&\, B_i$
$P_i = A_i + B_i$
$H_i = A_i \oplus B_i$

By using above architecture, concurrently sum and carries are calculating which reduces the delay, but the number of CS blocks is more.

Further we can reduce the CS blocks, which causes the area to be decrease by using the proposed sparse architecture with the introduction of gray cells

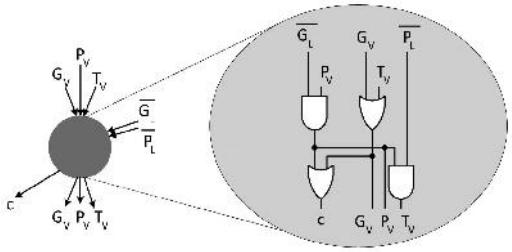


Fig 8: Gray Cell

The logic equations performed by a gray operator residing at prefix level j - 1 are

$$G_V^j = G_V^{j-1} + T_V^{j-1},$$
$$P_V^j = P_V^{j-1} \cdot \overline{G_L^{j-1}},$$
$$T_V^j = P_V^j \cdot \overline{P_L^{j-1}},$$
$$c^j = G_V^j + P_V^j.$$

.

V. IMPLEMENTATION OF MODULO 2^n+1 ADDERS BY USING PROPOSED SPARSE CARRY COMPUTATION

The implementation of proposed Sparse-4 modulo 2^{16}+1 diminished – 1 adder using gray cell, which reduces the area and time delay when compared with existing adders

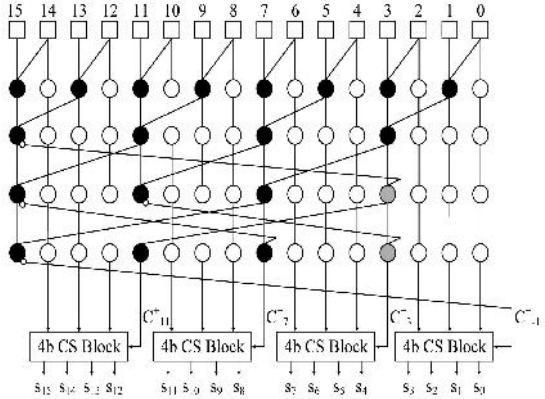


Fig 9: ProposeedSparse-4 modulo 2^{16}+1 diminished – 1 adder

Mainly in Network Security Related Applications The normal adders will not be useful because same number of information bits which are transmitted by doing encryption should be received at receiver when decrypted, but in normal adders addition of bits may be occurred so On Modulo 2^n+1 adders are used to produce the correct Information

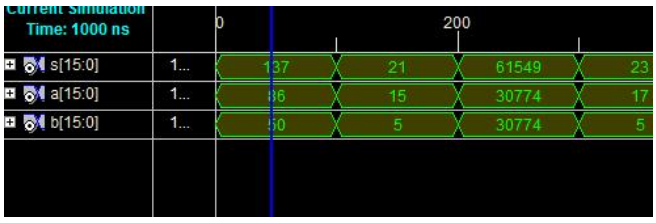
VI. SIMULATION RESULTS

Parallel prefix adder for Kogge-Stone

On modulo 2^n+1 adder

Modulo 2^n+1 adder by using Sparse carry computation

Modulo 2^n+1 adder by using Proposed Sparse carry computation



VII. SYNTHESIS RESULTS

Synthesis Results

	Area *Luts* *Slices*	*Delay (ns)*
Parallel Prefix Adder [Kogge]	15 out of 9312 9 out of 4656	13.912
Modulo_2^n +1 Adder	29 out of 9312 16 out of 4656	17.686
Modulo 2^n +1 adder sparse carry	123 out of 9312 68 out of 4656	19.415
Proposed Sparse-4 modulo 2^{16} +1 Adder	69 out of 9312 41 out of 4656	15.236

VIII. CONCLUSION

The proficient modulo 2^n + 1 adder are acceptable in a variety of computer applications including all RNS implementations. In modulo 2^n + 1 adder two hand-outs are offered to modulo 2^n + 1 problem. A narrative architecture has been proposed that uses a sparse totally regular parallel-prefix carry computation unit. This planning was plagiaristic by proving the upturned circular idem potency properties 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 calculation tree in the earlier application. The experimental results indicate that the projected architecture heavily outperforms the earlier solutions in implementation area and power consumption, while offering a high execution rate. Modulo 2^n + 1 addition problem was also shown to be related to modulo 2^n - 1 addition problem. The coalesce theory presented in this paper revealed that a simple post processing stage composed of an XOR gate for each output bit needs to be added to a modulo 2^n - 1 adder for attaining a modulo 2^n + 1 adder. As a result, every architecture that has been and more importantly that will be proposed for designing modulo 2^n - 1adders, can be reused for the design of modulo 2^n + 1 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.

CH Srinivasa Rao, M.tech VLSI, CMR IT
Pavan Kumar P, Associate Professor CMR IT