Fast Parallel Prefix Modulo 2n+1 Adder

G.Chandana[1], P.Rajini[2]
M.Tech

Abstract- **Two architectures for modulo 2n+1 adders are introduced in this paper. The first one is built around a sparse carry computation unit that computes only some of the carries of the modulo 2n+1 addition. This sparse approach is enabled by the introduction of the inverted circular idem potency property of the parallel-prefix carry operator and its regularity and area efficiency are further enhanced by the introduction of a new prefix operator. The resulting diminished-1 adders can be implemented in smaller area and consume less power compared to all earlier proposals, while maintaining a high Operation speed. The second architecture unifies the design of modulo 2n-1 adders. It is shown that modulo 2n-1 adders can be easily derived by straightforward modifications of modulo 2n-1 adders with minor hardware overhead.**

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 operators 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.

II.PARALLEL-PREFIX ADDITION BASICS

Suppose that A =An-1,An-2 . . .A0 and B=Bn-1,Bn-2 . . .B0 represent the two numbers to be added and S= Sn-1, Sn-2 . . . S0 denotes their sum. An adder can be considered as a three-stage circuit. The pre processing stage computes the carry-generate bits Gi, the carry-propagate bits Pi, and the half-sum bits Hi, for every i, $0 \leq i \leq n-1$, according to

$$Gi=Ai\ .Bi,\quad Pi=Ai+Bi\ ,\quad Hi=Ai\oplus Bi$$

Where . , + and ⊕denote logical AND, OR, and exclusive-OR, respectively. The second stage of the adder, here after called the carry computation unit, computes the carry signals Ci, for $0\leq i \leq n-1$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 'o' 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/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, that is,

$$(G_{K:J},P_{K:J)} = (GK\ ,PK)\ o\ (GK\text{-}1,PK\text{-}1)\ o(GJ,PJ).$$

Since every carry Ci=Gi:0, a number of algorithms have been introduced for computing all the carries using only operators. Fig. 1 presents the most well-known approaches for the design of an 8-bit adder, while Fig. 2 depicts the logic-level implementation of the basic cells used throughout the paper.

For large word lengths, the design of sparse parallel prefix adders is preferred, since the wiring and area of the design are significantly reduced without sacrificing 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, saving considerable amount of area in the carry-computation unit. The 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.

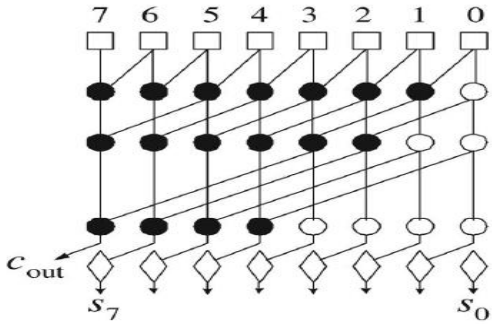

Fig1.Kogge-Stone Adder

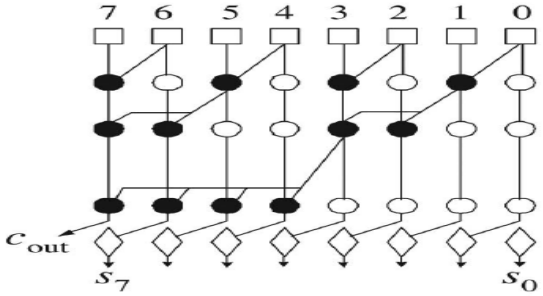

Fig2: Ladner-Fischer Adder

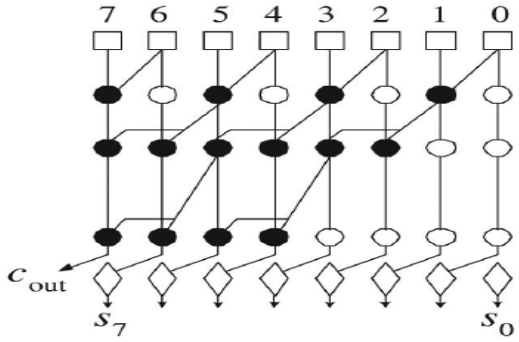

Fig3. Knowels Adder

SYNTHESIS RESULTS:

	AREA(μm^2)	DELAY (ns)
KOGGE-STONE	821	305
LADNER-FISCHER	829	307
KNOWLES	847	278

CONCLUSION

All the Parallel-Prefix adders gives less delay and area specifications by synthesis.

REFERENCES

[1] X. Lai and J.L. Massey, "A Proposal for a New Block EncryptionStandard,"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.

[6] V.K. Zadiraka and E.A. Melekhina, "Computer Implementation of Efficient Discrete-Convolution Algorithms," Cybernetics and Systems Analysis, vol. 30, no. 1, pp. 106-114, Jan. 1994.

[7] M.A. Soderstrand et al., Residue Number System Arithmetic: Modern Applications in Digital Signal Processing. IEEE Press, 1986.

[8] P.V.A. Mohan, Residue Number Systems: Algorithms and Architectures. Springer-Verlag, 2002.