Design of High Speed Modulo 2^n+1 Adder

M. Varun
M. Tech, Student
Department of ECE
Vardhaman College of Engineering

M. Nagarjuna
Assistant Professor
Department of ECE
Vardhaman College of Engineering

M. Vasavi
Assistant Professor
Department of ECE
Vazir Sultan College of Engineering

ABSTRACT
The two different architectures for 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 modulo 2^n+1 addition. This sparse approach is enabled by the introduction of 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 adder can be implemented in a smaller area and consume less power compared to all earlier proposals, maintaining a high operation speed. The second adder architecture unifies the design of modulo 2^n+1 adder. Both the adders are derived and compared by using the simulation results.

General Terms
Modulo adder, parallel-prefix computation, VLSI design.

Keywords
IEAC adder, Sparse-4 adder, RNS

1. INTRODUCTION
The modulo 2n+1 adder has the applications in many fields, say pseudorandom number generation, cryptography, convolution computations without round-off errors. It has the applications in residue number system (RNS) also. 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 between them, which leads 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, FIR filters and communication components, offering in several cases apart from enhanced operation speed and low power characteristics [1].

There are three input representations chosen for the input operands namely, the normal weighted one [2], the diminished-1 and the signed-LSB representations [3]. But, only the first two representations in the following are considered, since the adoption of the signed-LSB representation does not lead to more efficient circuits in delay or area terms. The input operands and results are limited between 0 and 2n when performing arithmetic operations modulo 2n + 1.

In normal weighted representation, each operand requires n+ 1 bit for its representation but only utilizes 2n+1 representation out of 2n+1 that these can provide. The diminished-1 representation offers a denser encoding of the input operands. In the diminished-1 representation, A is represented as azA*, where az is a single bit, often called the zero indication bit and A* is an n-bit vector, often called the number part. If A>0, then az=0 and A*=A-1, whereas for A=0, az=1 and A*=0. For example, the diminished-1 representation of A=5 modulo 17 is 001002.

Considering that the most common operations required in modulo 2n+1 arithmetic are negation, multiplication by a power of two and addition [4], the adoption of the diminished-1 representation, allows to limit these operations to n bits. Specifically, negation is performed by complementing every bit of A*, if az=0 and inhibiting any change when az=1. Multiplication by 2i is performed by an i-bit left rotation of the bits of A*, if az=0 and inhibiting any change when az=1. Finally, the addition of azA* and bzB* boils down to an n-bit modular addition of A* and B* with some minor modifications.

1.1 Related Work:
Several papers have attacked the problem of designing efficient diminished adders. The majority of them rely on the use of 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. Although an IEAC adder can be implemented by using an integer adder in which its carry output is connected back to its carry input via an inverter, but such a direct feedback is not a good solution. 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 [4]. To this end, a number of custom solutions have been proposed for the design of efficient IEAC adders.

Considering the diminished-1 representation for modulo 2^n+1 addition, [4], [5] used an IEAC adder which is based on an integer adder along with an extra carry look ahead (CLA) unit. The CLA unit computes the carry output which is then inverted, used as the carry input of the integer adder. Solutions that rely on a single carry computation unit have also been proposed. Zimmermann [5], [6] 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.

Although these architectures are faster than the carry look-ahead ones proposed in [7], for sufficiently wide operands, they are slower than the corresponding parallel-prefix integer adders because of the need for the extra prefix level. In [7], it has been shown 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 as fast as their integer counterparts, that is, they offer a logic depth of $\log_2 n$ prefix levels. Unfortunately, this level of performance requires significantly more area than the solutions of [5], [6] since a double parallel-prefix computation tree is required in several levels of the carry computation unit.

For reducing the area complexity of the parallel-prefix solutions, select-prefix [8] and circular carry select [9] IEAC adders have been proposed. Unfortunately, both these proposals achieve a smaller operating speed than the parallel-prefix ones of [7]. Recently, very fast IEAC adders that use the Ling carry formulation of parallel-prefix addition [10] have appeared in [11], which also suffer from the requirement of a double parallel-prefix computation tree.

Although a modulo 2^n+1 adder that follows the (n+1)-bit weighted representation can be designed following the principles of generic modulo adder design [12], specialized architectures for it have appeared in [13], [14]. However, it has been recently shown [15] that a weighted adder can be designed efficiently by using an IEAC one and a carry save adder (CSA) stage. As a result, improving the design for an IEAC adder would improve the weighted adder design as well.

2. PARALLEL-PREFIX ADDERS:

Suppose that $A= A_{n-1}A_{n-2}\ldots A_0$ and $B= B_{n-1}B_{n-2}\ldots B_0$ represent the two numbers to be added and $S= S_{n-1}S_{n-2}\ldots S_0$ denotes their sum. An adder can be considered as a three-stage circuit. The preprocessing stage computes the carry-generate bits G_i, the carry-propagate bits P_i, and half-sum bits H, for every i, $0 \le i \le n-1$, according to

$$G_i = A_i . B_i \quad Pi = A_i + B_i \quad Hi = A_i \oplus B_i,$$

Where, +, $\oplus$ denote logical AND, OR, and exclusive-OR respectively. The second stage of the adder, hereafter called the carry computation unit, computes the carry signals C_i, for $0 \le i \le n-1$ using the carry generate and carry propagate bits G_i 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 ∘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, ….., j, that is,

$$G_{k:j} = (G_k, P_k) o (G_{k-1}, P_{k-1}) o \ldots o (G_j, P_j)$$

Since every carry $C_i = G_{i:0}$, 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 in the paper.

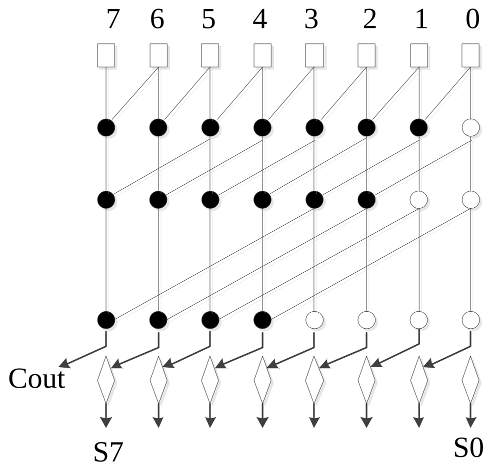


Fig 1: Kogge-Stone adder

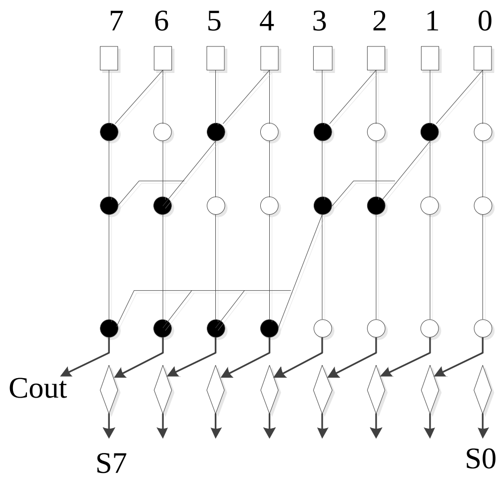


Fig 2: Ladner-Fischer adder

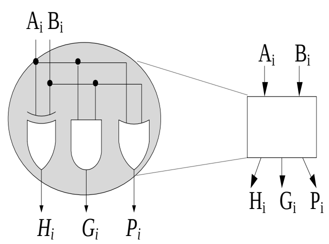


Fig 2 (a):

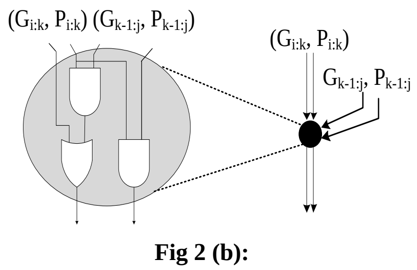


Fig 2 (b):

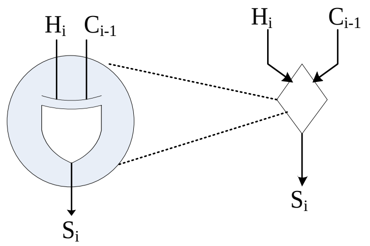


Fig 2 (c):
The logic level implementations of the basic cells used in the parallel prefix adders.

3. MODULO ADDERS:

Modulo adders are important for several applications including residue number system, digital signal processors and cryptography algorithms. 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:

- **a.** When none of the input operands is zero (a_z, $b_z \neq 0$) their number parts A* and B* are added modulo 2^n+1. This operation is discussed in the following, can be handled by an IEAC adder.
- **b.** When one of the two inputs is zero, the result is equal to the nonzero operand.
- **c.** 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. 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.

4. SPARSE-4 PARALLEL-PREFIX STRUCTURE FOR 16-BIT:

Parallel Prefix Adder (PPA) is very useful in today's world of technology because of its implementation in Very Large Scale Integration (VLSI) chips. The VLSI chips rely heavily on fast and reliable arithmetic computation. These contributions can be provided by PPA. For larger 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.

A 32-bit adder with 4-bit sparseness is shown in fig. 3a. 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. A possible logic-level implementation of a 4-bit carry-select block is shown in Fig 4 (b). The following architecture shows the sparse-4 parallel-prefix adder structure for 16-bit and its CS-block logic level implementation.

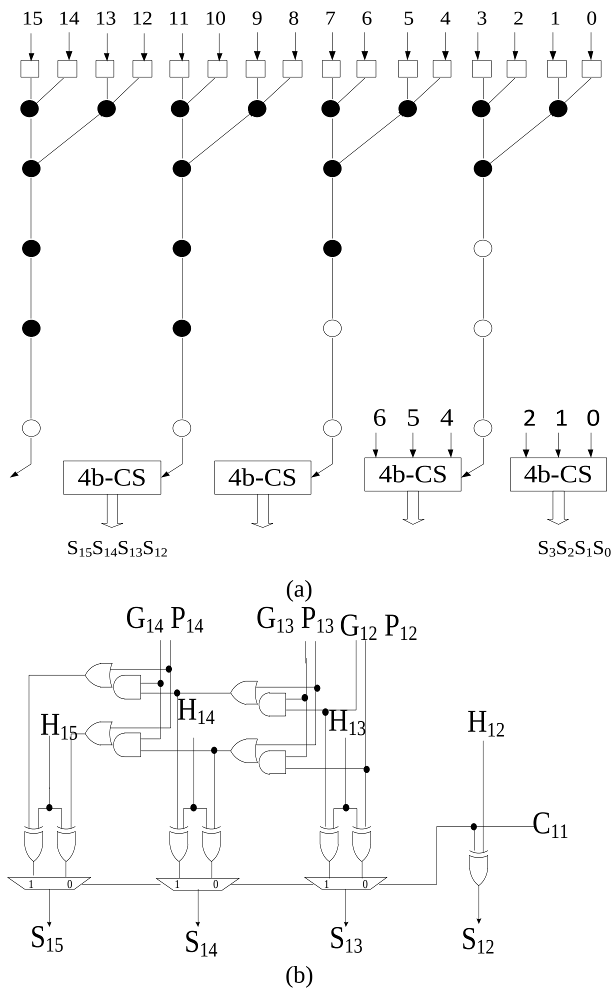


Fig4: (a) Sparse-4 parallel-prefix structure for 16-bit integer adder and (b) the logic level implementation of the CS block

5. NEW SPARSE MODULO 2^n+1 ADDER:

In this section, it is to be focussed on the design of diminished modulo adder with a sparse parallel-prefix carry computation stage that can use the same carry-select blocks as the sparse carry-select blocks as the sparse integer adder [19]. In the previous sections, partially regular and totally regular sparse parallel-prefix units are introduced [17].In this paper, by making small changes to the proposed architecture, there will be a reduce in the delay and thus gets an improved operational speed [18].Here the carry kill concept is used.

$$K_i = A_i + B_i$$

$$P_i = A_i \oplus B_i$$

Parallel Prefix addition is a technique for improving the speed of binary addition. Due to continuing integrating intensity and the growing needs of portable devices, low-power and high performance designs are of prime importance.

The following architecture introduces four different computation nodes for achieving improved performance namely odd dot, even dot, odd semi-dot and even semi-dot.

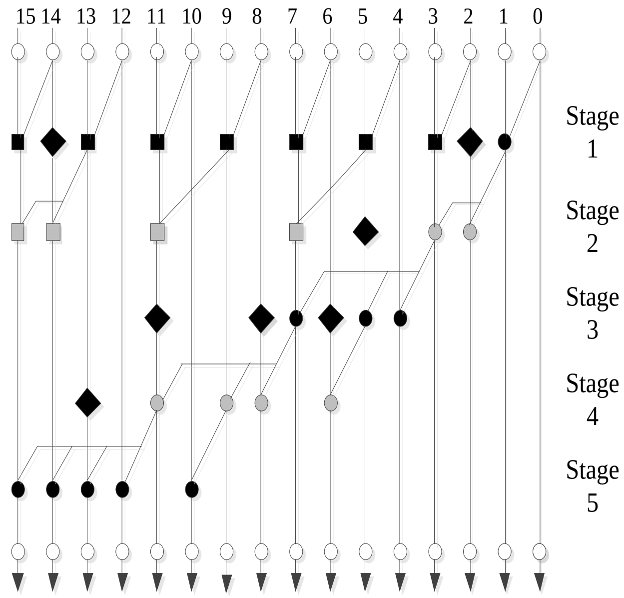


Fig 5: Proposed sparse-4 modulo $2^{16} + 1$ adder

The black-cell representation is given below:

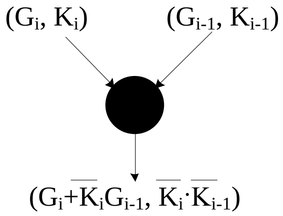


Fig 6: Black cell

$$(G_i, K_i)o(G_{i-1}, K_{i-1}) = (G_i + \overline{K_i}\cdot G_{i-1}, \overline{K_i}\cdot\overline{K_{i-1}})$$

The representations chosen for the network are as follows:

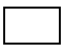
	Even dot
	Odd dot
	Even semi-dot
	Odd semi-dot

Table 1: Cells representation

The even semi-dot and odd semi-dot are used at the last stage of network and the output representations are given as $\overline{G}$ and *G*.

$$\overline{G} = \overline{G}_i + \overline{K}_i \bullet G_{i-1}$$
$$G = \overline{G}_i \bullet (\overline{K}_i + \overline{G}_{i-1})$$

The output of the odd-semi-dot cells gives the value of the carry signal in that corresponding bit position. The output of the even-semi-dot cell gives the complemented value of carry signal in that corresponding bit position.

The even dot and odd dot representations are as follows:

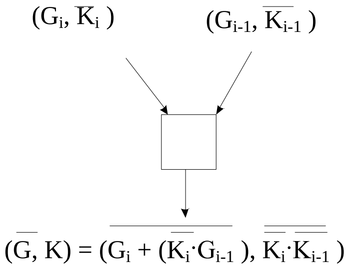


Fig 7: Even dot

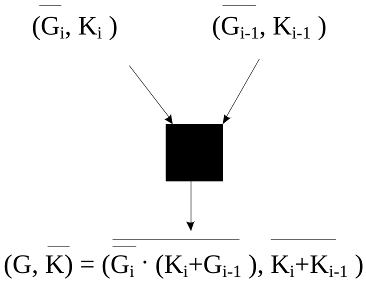


Fig 8: Odd dot

Therefore,

$$(\overline{G}, K) = ((\overline{G_i + (\overline{K}\bullet G_{i-1})}, \overline{\overline{K}_i\bullet\overline{K}_{i-1}})$$

And

$$(G, \overline{K}) = (\overline{\overline{G}_i\bullet(K_i + G_{i-1})}, \overline{(K_i + K_{i-1})})$$

, and the inverter representation is:

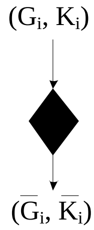


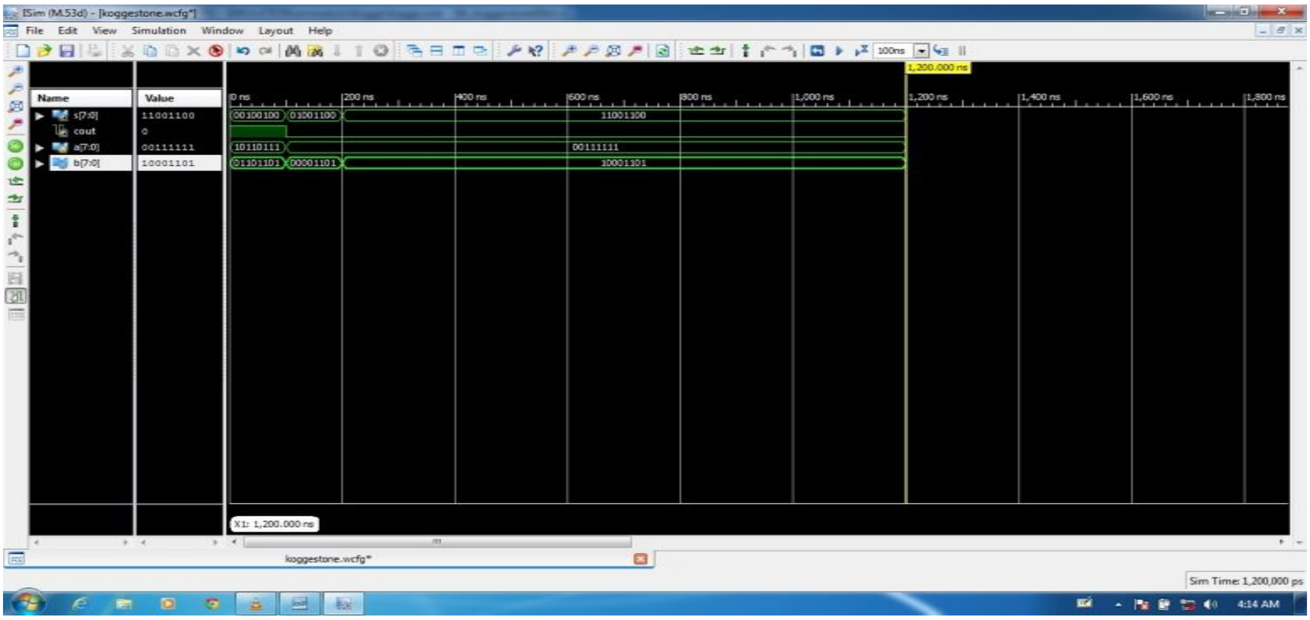
Fig 9: Inverter

'Nor' operation is used instead of 'or' to reduce the number of transistors used (for nor 4 transistors are used whereas for or 6 are used).

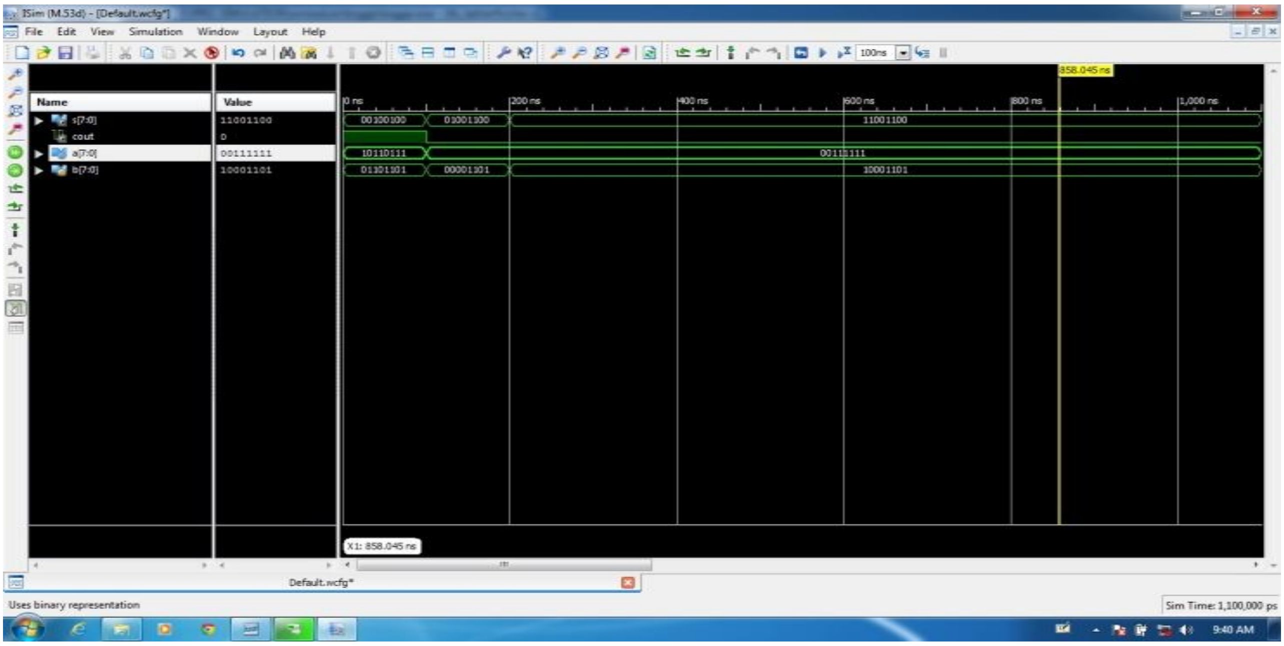
6. COMPARISIONS:

In this section, first compare all the diminished-1 adders that use the totally regular parallel-prefix IEAC adders presented in the previous sections against the diminished-1 adders proposed and those that use the IEAC proposed in [6], [7], [11]. It will be considered that all the diminished-1 adders can handle true operands and indicate true zero results. For the High-Speed Fermat Number Transform Based adders, consider the carry output computed by the CLA unit is used as a late increment carry signal in the successor integer adder. For the latter, it is then considered that it follows the Ladner-Fischer (LF) proposal augmented by a carry increment prefix level. For the IEAC adders of [6], it is considered that the first $\log_2 n$ prefix levels may either follow the Ladner-Fischer (LF) or the Kogge-Stone (KS) proposal. Finally, both the reduced area parallel prefix (RAPP) and the full parallel prefix (FPP) architectures of the IEAC adders that use Ling carries [11] are examined. The stages with odd indexes use odd-dot and odd-semi-dot cells where as the stages with even indexes use even-dot and even-semi-dot cells. Cascading odd cells and even cells alternatively gives the benefit of elimination of two inverters between them, if a dot or a semi-dot computation node in an odd stage receives both of its input edges from any of the even stages and vice-versa. But it is essential to introduce two inverters in a path, if a dot or a semi-dot computation node in an even stage receives any of its edges from any of the even stages and vice-versa.

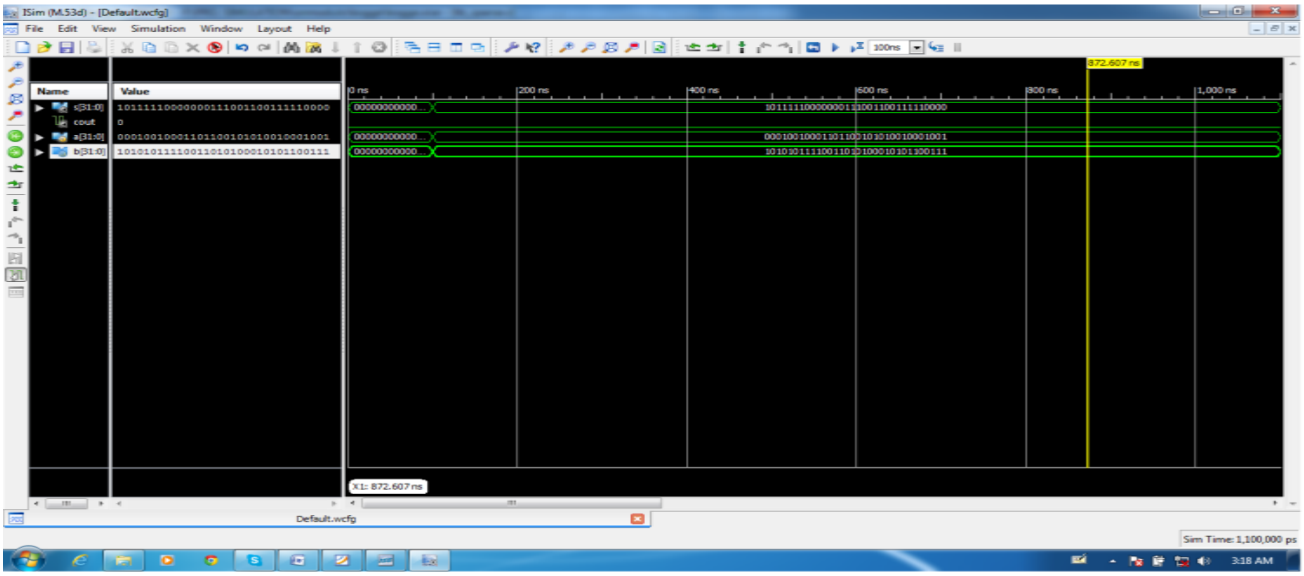
7. SIMULATION RESULTS:



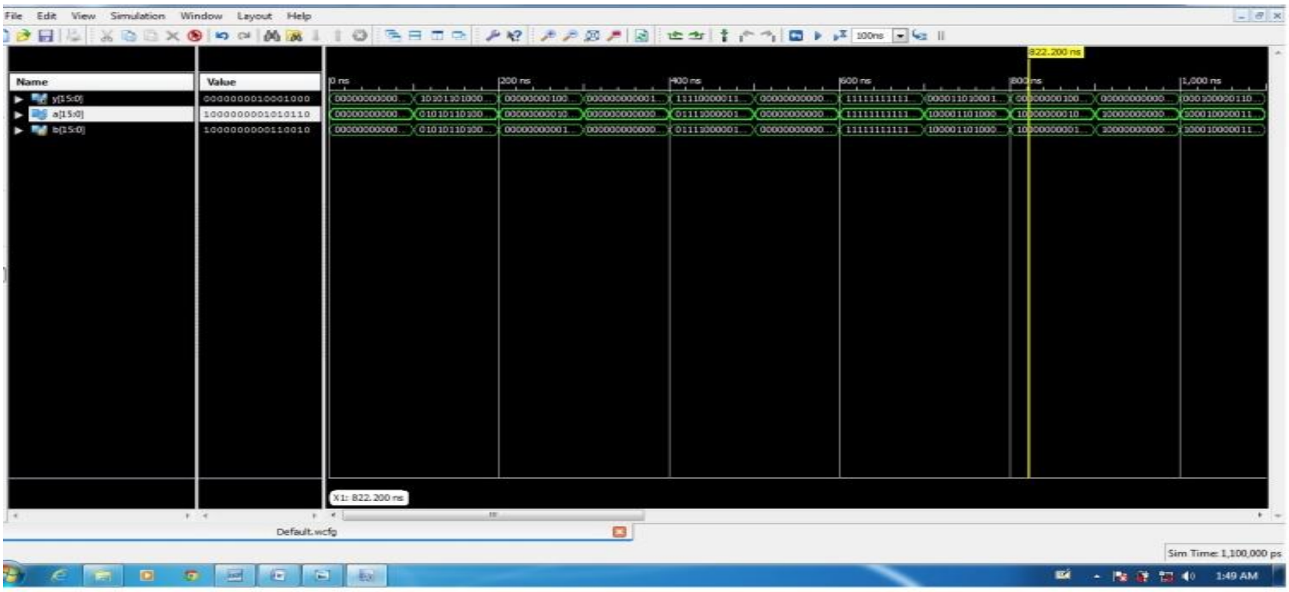
Output 1: Kogge-Stone adder



Output 2: Ladner-Fischer adder



Output 3: Sparse parallel-prefix adder



Output 4: Sparse-4 modulo 2^n+1 diminished-1 adder

Table 1: Synthesis Results

	Sparse-4 parallel-prefix adder	Sparse-4 modulo 2^n+1 adder
No. of 4 input LUTs	57	39
No. of slices	30	24
No. of bonded IOBs	48	48
Fan-out	2.94	2.24

8. CONCLUSION:

The parallel prefix formulation of binary addition is a very convenient way to formally describe an entire family of parallel binary adders. A novel architecture has been proposed that uses a sparse totally regular parallel-prefix carry computation unit. This architecture was derived by introducing even and odd dot and semi-dot operators and an inverter. The architecture has five stages of implementation. The output of the odd semi-dot cells gives the value of the carry signal in that corresponding bit position. The output of even semi-dot cell gives the complemented value of carry signal in that corresponding bit position. By introducing two cells for dot operator and two cells for semi-dot operator, a large number of inverters are eliminated. Due to inverter elimination in paths, the propagation delay in these paths has reduced. The proposed architecture is compared with the sparse-4 parallel-prefix structure and thus proved that proposed one has a reduced delay and high operational speed. Further achieving a benefit in power reduction, since these inverters if not eliminated, would have contributed to significant amount of power dissipation due to switching. It uses 39 LUTs and 24 slices. The results are shown in table. Xilinx 12.1 tool is used for simulation.

9. REFERENCES

[1] R. Chokshi, K.S. Berezowski, A. Shrivastava, and S.J. Piestrak, "Exploiting Residue Number System for Power-Efficient Digital Signal Processing in Embedded Processors," Proc. Int'l Conf. Compilers, Architecture, and Synthesis for Embedded Systems (CASES '09), pp. 19-28, 2009.

[2] L.M. Leibowitz, "A Simplified Binary Arithmetic for the Fermat Number Transform," IEEE Trans. Acoustics, Speech and Signal Processing, vol. ASSP-24, no. 5, pp. 356-359, Oct. 1976.

[3] G. Jaberipur and B. Parhami, "Unified Approach to the Design of Modulo-($2^n \pm 1$) Adders Based on Signed-LSB Representation of Residues," Proc. 19th IEEE Symp. Computer Arithmetic, pp. 57-64, 2009.

[4] J.J. Shedletsky, "Comment on the Sequential and Indeterminate Behavior of an End-Around-Carry Adder," IEEE Trans. Computers, vol. C-26, no. 3, pp. 271-272, Mar. 1977.

[5] R. Zimmermann, "Binary Adder Architectures for Cell-Based VLSI and Their Synthesis," PhD dissertation, Swiss Fed. Inst. Of Technology, 1997.

[6] R. Zimmerman, "Efficient VLSI Implementation of Modulo $2^n \pm 1$ Addition and Multiplication," Proc. 14th IEEE Symp. Computer Arithmetic, pp. 158-167, Apr. 1999.

[7] 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.

[8] C. Efstathiou, H.T. Vergos, and D. Nikolos, "Modulo $2^n \pm 1$ Adder Design Using Select Prefix Blocks," IEEE Trans. Computers, vol. 52, no. 11, pp. 1399-1406, Nov. 2003.

[9] 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.

[10] G. Dimitrakopoulos and D. Nikolos, "High-Speed Parallel-Prefix VLSI Ling Adders," IEEE Trans. Computers, vol. 54, no. 2, pp. 225- 231, Feb. 2005.

[11] H.T. Vergos and C. Efstathiou, "Efficient Modulo $2^n + 1$ Adder Architectures," Integration, the VLSI J., vol. 42, no. 2, pp. 149-157, Feb. 2009.

[12] M. Bayoumi, G. Jullien, and W. Miller, "A VLSI Implementation of Residue Adders," IEEE Trans. Circuits and Systems, vol. CAS-34, no. 3, pp. 284-288, Mar. 1987.

[13] A. Hiasat, "High-Speed and Reduced-Area Modular Adder Structures for RNS," IEEE Trans. Computers, vol. 51, no. 1, pp. 84-89, Jan. 2002.

[14] C. Efstathiou, H.T. Vergos, and D. Nikolos, "Fast Parallel-Prefix Modulo $2^n + 1$ Adders," IEEE Trans. Computers, vol. 53, no. 9, pp. 1211-1216, Sept. 2004.

[15] H.T. Vergos and C. Efstathiou, "A Unifying Approach for Weighted and Diminished-1 Modulo $2^n + 1$ Addition," IEEE Trans. Circuits and Systems II, vol. 55, no. 10, pp. 1041-1045, Oct. 2008.

[16] R.P. Brent and H.T. Kung, "A Regular Layout for Parallel Adders," IEEE Trans. Computers, vol. C-31, no. 3, pp. 260-264, Mar. 1982.

[17] S. Mathew, M. Anders, R.K. Krishnamurthy, and S. Borkar, "A 4- GHz 130-nm Address Generation Unit with 32-bit Sparse-Tree Adder Core," J. Solid-State Circuits, vol. 38, no. 5, pp. 689-695, May 2003.

[18] Haridimos T. Vergos and Giorgos Dimitrakopoulos, "On Modulo $2^n + 1$ adder design", IEEE transactions on computers, vol. 61, No. 2, Feb, 2012.

[19] K. Nehru, A. Shanmugam and S. Vadivel, "Design of 64-Bit Low Power Parallel Prefix VLSI Adder for High Speed Arithmetic Circuits".