International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

ASIC Implementation and Comparison of Diminished-one Modulo 2^n+1 Adder

Raj Kishore Kumar[1], Vikram Kumar[2]

[1]Shivalik Institute of Engineering & Technology

[2]Assistant Professor, Shivalik Institute of Engineering & Technology

Abstract: ***Modulo 2^n+1 adder is one of the most common adder which have frequent use in RNS operations that has significant critical path, and often encountered in applications like pseudo-random number generation and cryptography as well. In this paper we have presented various types of diminished-one modulo 2^n+1 adder using globular carry selection(GCS) and several parallel prefix adders. The adder has been simulated using verilog HDL codes and mapped this design to the TSMC (90 nm) and calculated the area, power dissipation and time for n= 8,16,32 & 64 and shown in the graph that which adder will have better power efficiency and time delay. Area occupied by several bits is presented in the table.***

Keyword:- RNS, Parallel Prefix Adder, GCS, ASIC

1. Introduction

Arithmetic modulo 2^n+1 has wide range of applicability ranging from pseudorandom number generation and cryptography without round-off errors [1], [2], [3]. Also, modulo 2^n+1 arithmetic operators are frequently included in residue number system (RNS) application. Most of the case, RNS is extremely good for many applications-such as computer security, speech processing, communication engineering, digital signal processing, image processing, and transformation in which the critical arithmetic operations are addition and multiplication. RNS offers in several cases apart from enhanced operation speed, low-power characteristics. RNS are more complex than the conventional arithmetic, and, consequently, cannot be implemented directly with the same efficiency. Therefore, in practice, residue arithmetic is frequently realized in terms of lookup-tables (to avoid the complex combinational-logic circuits) and conventional arithmetic(i.e. arithmetic in some standard notation). For example, the sign-and-magnitude approach may be convenient for representing signed numbers in RNS, but actual arithmetic operations might be best realized in terms of radix-complement arithmetic. We shall also see that certain choices of representational parameters in RNs naturally lead to diminished-radix complement arithmetic.

The difficulty of a modulo 2^n+1 arithmetic unit is defined by the representation chosen for the input operands. There are three representations which have been considered: namely, the signed-LSB representation, the normal weighted one and the Diminished-one. The adoption of the signed-LSB representation does not lead to more efficient circuits in delay or area terms so that we only consider the last two representations in the following. when performing arithmetic operations modulo 2^n+1 in all cases, the input operands and the results are limited between 0 and 2n. In the case of the normal weighted representation, each input operand requires n+1 bits for its representation but only utilizes C representations out of the 2^n+1 that these can provide. Other way to simplify arithmetic operations modulo 2^n+1 are offered by the diminished-one representation. In the diminished-one representation, A is represented as a_z A^*, where a_z is a single bit, frequently called the zero indication bit, and A^* is an n-bit vector, often called the number part. a =0 and A^*=A-1 only if A>0, whereas for A=0, a_z=1 and A^*=0. For example, the diminished-one representation of A=6modulo17 is 00101_2 considering that the most common operation required in modulo 2^n+1 arithmetic are multiplication by a power of two, negation and addition, so in the case of diminished-one representation, allows to limit these operations to n bits. Specifically negation is performed by complementing every bit of A^*, if a_z=0 and prevent any change when a_z=1. In this paper we have presented a paper on diminished-one modulo 2 +1 arithmetic adder using GCS which is based on the paper[5]. This adder consists of a DS-CLA, a GCS and a multiplexer. The DS-CLA adder produces two sets of modulo results in parallel and the GCS computes the carry out bit and globularly control the multiplexer to get the correct modulo sum from DS-CLA adder.

The paper has been organized in six section. In section 2, the details of the basics of parallel-prefix addition is presented. All operations involved in modulo 2^n+1 addition for diminished-one operands are discussed in details in section 3. Section 4 gives the details of the diminished-one modulo adder using GCS is presented. Section 5 gives the performance parameter comparison and the conclusion part is given in section 6.

2. Parallel-Prefix addition basics

Consider 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 $S=S_{n-1}S_{n-2}.....S_0$ shows there sum. An adder can be divided as a three stage circuit. The preprocessing stage computes the

carry-generate bits G_i, the carry-propagate bits P_i, and the half-sum bits H_i, for all i, $0 \le i \le n-1$, according to

$$G_i = A_i \bullet B_i \qquad P_i = A_i B_i \qquad H_i = A_i + B_i,$$

Where the sign ●, **+, denotes** logical AND,OR, and exclusive-OR, respectively. After this stage, the second stage of the adder called the carry signals C_i, for $0 \le i \le n-1$ using the carry propagate and carry generate bits G_i and P_i. The third stage computes the sum bits as

$$S_i = H_i \, C_{i-1}$$

Carry computation get changed into a parallel prefix problem by using the ○ operator, which includes pair of generate and propagate signals and was defined in [10] as

$$(G, P) \circ (G', P') = (G + P.G' \;\; P.P')$$

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

$$(G_{K:j}, P_{K:j}) = (G_K, P_K) \circ (G_{K-1}, P_{K-1}) \circ \ldots\ldots \circ (G_j, P_j)$$

Since every carry $C_i = G_{i:0}$, various types of algorithms have been introduced for computing all the carries using only ○ operators. Fig.1 presents the well-known approaches for the design of an 8-bit adder, while Fig.2 shows the logic-level implementation of the basic cells used throughout the paper. When we use large word lengths, the design of sparse parallel prefix adders is preferred, because the wiring and area of the design are significantly reduced without sacrificing delay. The basic design of sparse adders depends on the use of a sparse parallel-prefix carry computation unit and carry select blocks. It saves considerable amount of area in the carry-computation unit[4]. Carry selection block work with the two sets of sum bits corresponding to the two possible values of the incoming carry and select the proper sum without any delay overhead.

3. Modulo $2^n \pm 1$ addition basics

3.1 Modulo 2^n-1 adders:-

The modulo 2^n-1 addition depends on a conditional operation defined as

$$(A+B) mod (2^n+1) = \begin{cases} (A+B), & A+B < 2^n \\ (A+B+1) mod\ 2^n & A+B \ge 2^n \end{cases}.$$

When $A+B \ge 2^n$, A modulo 2^n-1 adder can be implemented using an integer adder that increments also its sum when the carry output is one, which is equivalent to feed the carry-input of the adder with the carry-output of the first addition. Additional carry increment stage can implement the conditional increment as shown in Fig.4a. 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 the expected result which is equal to zero. To implement 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 equal to $= G_{i:0} + P_{i:0} . C_{in}$. So that connecting the carry output to the carry input leads to $= G_{i:0} + P_{i:0} . C_{n-1:0}$.

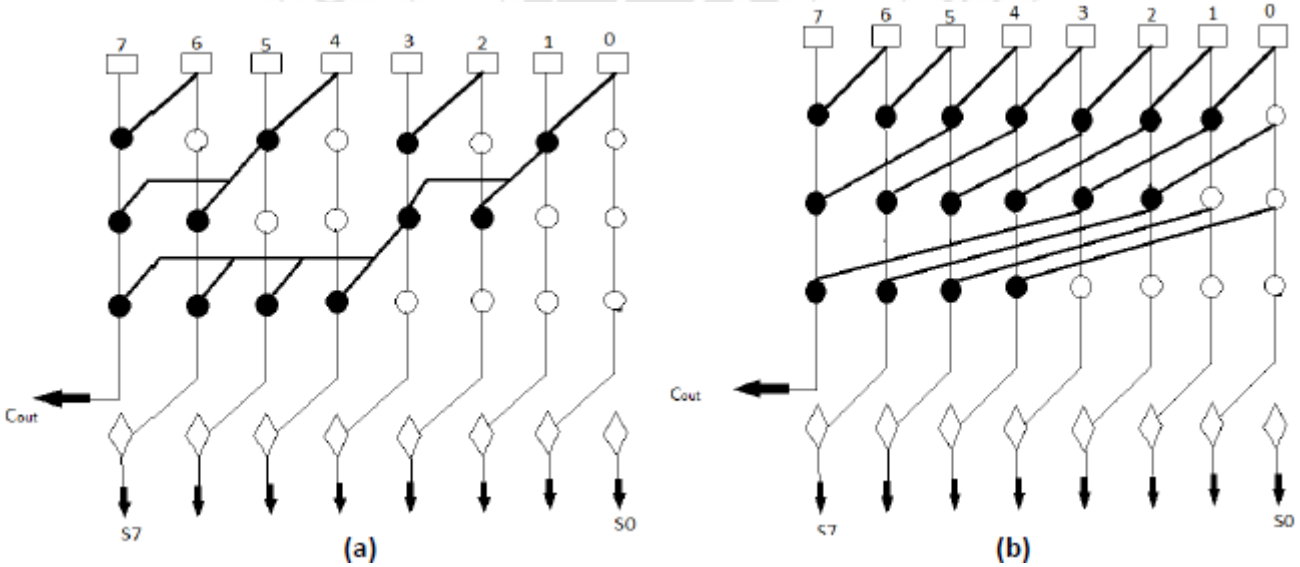

Figure 1: Examples of 8-bit parallel-prefix structures for integer adders (a) Kogge-stone[7] (b) Ladner-Fischer[8]

This relation can be simplified [] to

$$C_i^- = G_{i:0} + P_{i:0} \cdot G_{n-1:i+1} \ldots\ldots\ldots\ldots (1)$$

This equation is equivalent to

$$C_i^- \leftrightarrow (G_i, P_i) \circ \ldots\ldots \circ (G_0, P_0) \circ (G_{n-1}, P_{n-1}) \circ \ldots\ldots \circ (G_{i+1}, P_{i+1}) \qquad \ldots\ldots\ldots\ldots\ldots (2)$$

Equation (2) computes the modulo 2^n-1 carries has a cyclic form and, in terms of the number of generate and propagate pairs (G_i , P_i) is equal to n. Therefore the parallel-prefix carry computation unit of a modulo 2^n-1 adder has increased area complexity than that of a corresponding integer adder.

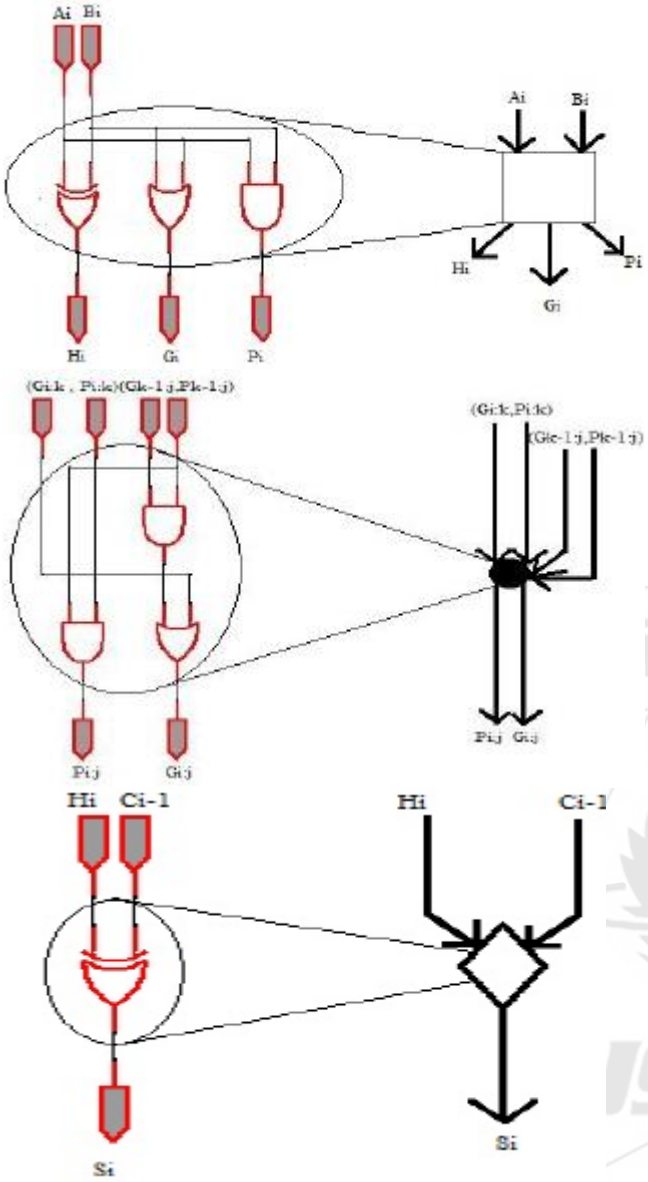

Figure 2: The logic level implementation of the basic cells used in parallel-prefix adders

3.2 Modulo 2^n+1 adders

In terms of complexity, Diminished-1 modulo 2^n+1 addition is more complex so that special care is required when at least one of the input operand is zero (100...0). When none of the input operands is zero their number parts A^* and B^* are added modulo 2^n+1. When ant one of the two inputs is zero the result is nonzero operand and when both operands are zero, the result is zero. The Diminished-one sum is derived by the number parts A^* and B^* of the input operands, when none of the input operands is zero, as follows:-

$$S^+ = (A^* + B^*) \bmod (2^n+1) = \begin{cases} (A^* + B^* + 1) mod 2^n, & A^* + B^* < 2^n \\ (A^* + B^*) mod 2^n, & A^* + B^* \geq 2^n \end{cases} \quad \text{.....(3)}$$

4. Globular Carry Selection Diminished-one Modulo 2^n+1 Adder

Suppose that the two n-bit diminished-one operands are A^*= A-1= $a^*_{n-1}..\ a^*_0$ and B^*= B-1 = $b^*_{n-1}...b^*_0$. So their sum $S^* = s^*_{n-1}..... s^*_0$ is derived by doing modulo 2^n+1 addition of these two operands. Sum can be changed into the uncomplicated function with performing modulo 2^n addition as:

$$S^* = \langle A^* + B^* + C'_{n-1} \rangle 2^n \text{ .....(4)}$$

Where C_{n-1} denotes as an original carry-out bit of ($A^* + B^*$).

According to the carry look ahead adder function [12], the carry term

$$C^*_i = G^*_i + \sum_{j=0}^{i-1} \left(\prod_{K=j+1}^{i} P^*_K \right) G^*_j + C^*_{-1} \prod_{K=0}^{i} P^*_K \quad for\ i = 0, \ldots, n-1$$

Where denotes the carry-in bit. According to GCS technique, we set $C^*_{-1} = C'_{n-1}$. The boolean function of each sum bit in (4) can be shown as:

$$S^*_i = C^*_{i-1} \oplus P^*_i = \left(G^*_{i-1} + \sum_{j=0}^{i-2} \left(\prod_{K=j+1}^{i-1} P^*_K \right) G^*_j + C'_{n-1} \prod_{K=0}^{i-1} P^*_K \right) \oplus P^*_i \text{ .....(5)}$$

Where,

$$C_{n-1} = G^*_{n-1} + \sum_{j=0}^{n-2} \left(\prod_{K=j+1}^{n-1} P^*_K \right) G^*_j \text{ ..... (6)}$$

Since $C_{n-1} \in \{0, 1\}$, so that

$$S^*_i = \begin{cases} S^*_{i,1} = (G^*_{i-1} + \sum_{j=0}^{i-2} (\prod_{K=j+1}^{i-1} P^*_K) G^*_j) \oplus P^*_i + \prod_{K=0}^{i-1} P^*_K \oplus P^*_i, & if\ C_{n-1} = 0 \\ S^*_{i,0} = (G^*_{i-1} + \sum_{j=0}^{i-2} (\prod_{K=j+1}^{i-1} P^*_K) G^*_j) \oplus P^*_i, & if\ C_{n-1} = 1 \end{cases} \text{ .....(7)}$$

In (7), We can clearly see that they have the same term $(G^*_{i-1} + \sum_{j=0}^{i-2} (\prod_{K=j+1}^{i-1} P^*_K) G^*_j) \oplus P^*_i$, so that it is easy to design a DS-CLA adder to produce two sums $S^*_{i,1}$ and $S^*_{i,0}$. Meanwhile, the carry C_{n-1} generated by the CLA function of (6) is globularly used to control multiplexer for getting the appropriate outputs $S^{*'}_i s$. The block diagram of GCS Diminished-one modulo 2^n+1 adder is shown in Fig.3. At a clear point of view, Fig.4. shows the detailed logic design for GCS Diminished-one modulo 2^4+1 adder.

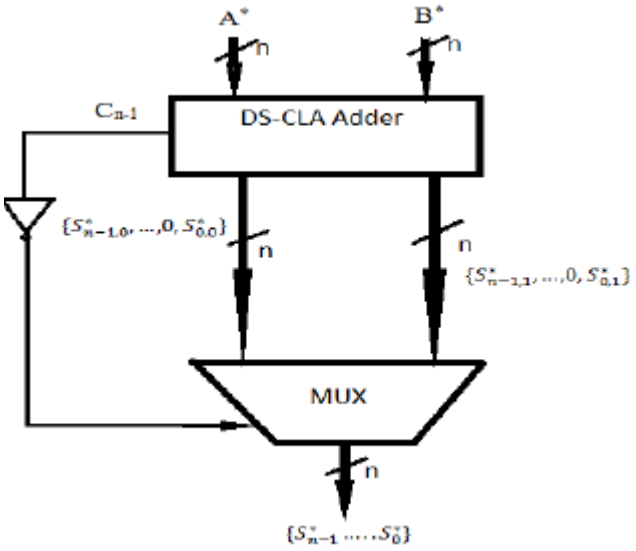

Figure 3: Block diagram of GCS diminished-one modulo 2^n+1 adder

To increase the speed of the GCS modular adder for the large dimension of n, we divide the n-bit GCS modular adder into m r-bit GCS addition blocks and a fast GCS where n=m x r. Fig. 5 illustrates the partitioned GCS modular adder architecture. Both the input operands are divided into m blocks inputs a $A^*=\{A^*_{m-1} \dots \dots A^*_0\}$ and $B^*=\{B^*_{m-1} \dots \dots B^*_0\}$, where $A^*_t=\{a^*_{(t+1)r-1} \dots \dots a^*_{tr+1}a^*_{tr}\}$ and $B^*_t=\{b^*_{(t+1)r-1} \dots \dots b^*_{tr+1}b^*_{tr}\}$ for t=0,.....,(m-1).

The block sum $S^*_t=\{s^*_{(t+1)r-1} \dots .. s^*_{tr+1}s^*_{tr}\}$ is derived

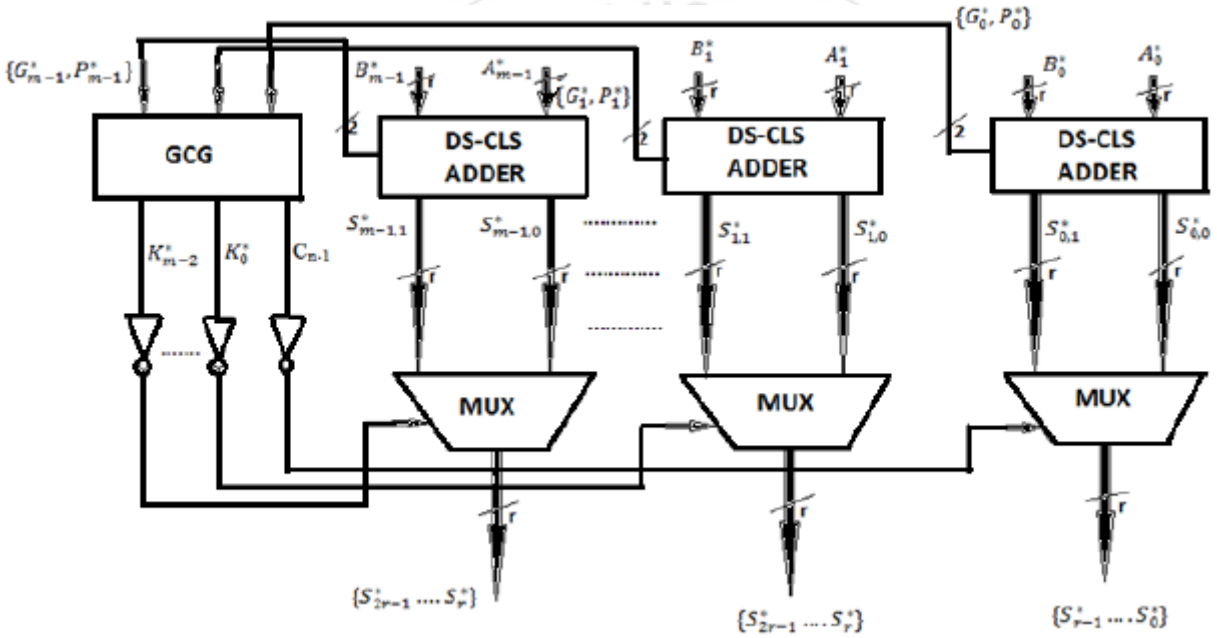

Figure 4: The m x r partitioned GCS modular adder

by the equation $A^*_t + B^*_t + K^*_{t-1}$ Where K^*_{t-1} denotes the carry-out bit of the (t-1)[th] addition block. In each GCS addition block, the DS-CLA adder generates two block sums $S^*_{t,1}=S^*_t$ for $K^*_{t-1}=1$ and $S^*_{t,0}=S^*_t$ for $K^*_{t-1}=0$ in parallel. Same as, the carry-out bit K^*_{t-1} is used to select the correct block sum. Each carry-out signal K^*_{t-1} for t=1,.....,m-1 can be generated by GCG as follows:

$$K^*_{t-1} = G^*_{t-1} + \sum_{j=0}^{t-2}\left(\prod_{i=j+1}^{t-1} P^*_i\right)G^*_j + C'_{n-1}\prod_{i=0}^{t-1} P^*_i$$

$$\begin{cases} K^*_{t-1,1}=G^*_{t-1}+\sum_{j=0}^{t-2}\left(\prod_{i=j+1}^{t-1} P^*_i\right)G^*_j+\prod_{i=0}^{t-1} P^*_i, if\ C_{n-1}=0 \\ K^*_{t-1,0}=G^*_{t-1}+\sum_{j=0}^{t-2}\left(\prod_{i=j+1}^{t-1} P^*_i\right)G^*_j\ if\ C_{n-1}=0 \end{cases} \quad \dots\dots\dots(8)$$

Where the block generate term $G^*_t = G^*_{tr+(r-1)} + \sum_{j=tr}^{tr+(r-2)}\left(\prod_{K=j+1}^{tr+(r-1)} P^*_K\right)G^*_j$ and the block propagate term $P^*_t = \prod_{K=tr}^{tr+(r-1)} P^*_K$ are given by the t[th] GCS addition block. So that the carry-out bit

$$C_{n-1} = G^*_{m-1} + \sum_{j=0}^{m-2}\left(\prod_{i=j+1}^{m-1} P^*_i\right)G^*_j \quad \dots\dots\dots\dots(9)$$

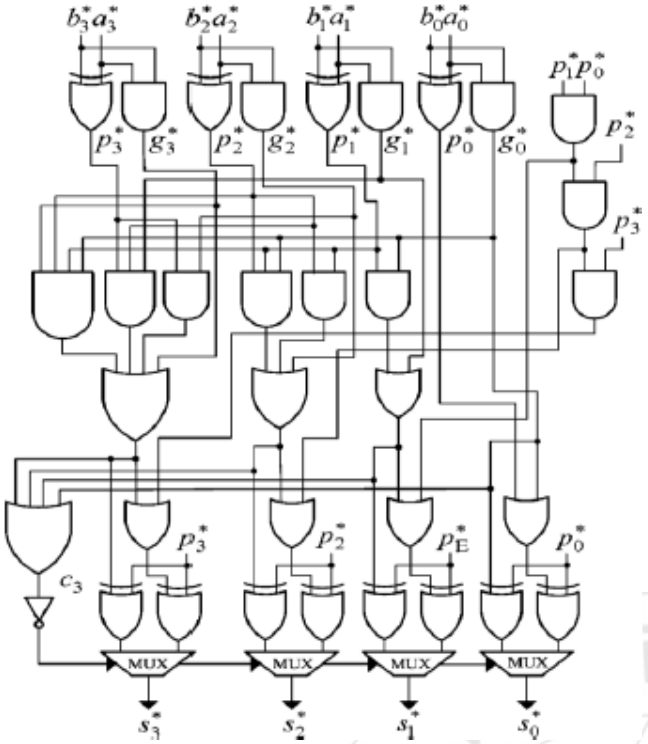

Figure 5: Logic circuit of CCS diminished-one modulo 2^4+1 adder[5]

5. Comparison and VLSI Implementation

We compare the GCS diminished-one modulo 2^n+1 adder against three parallel-prefix modular adder[7] [8] [10] which are regarded as the fast and the more area efficient design for arithmetic addition. In order to make an accurate comparison, we use TSMC 90 nm design kit with Cadence's rc tool to implement the designs for n= 8, 16, 32, 64. Then calculated the area, power dissipation and time delay and shows the AD (Area x Delay) and DP (Delay x Power) products. All the area results are calculated in μm^2, delay results are given in ns and power dissipation results are expressed in µw. Table1 shows the comparison in terms of area, delay and power consumption and its product for various bits. The shaded parts in the table shows the best results for the specific dimension of n. In the graph, we can see that for n > 8, the GCS modular adder has less power delay product than [7], [8]. [10]. This result shows that GCS modular adder to be profitable for many real application when to require a good compromise in area, delay and power.

Table 1: Comparison of the Synthesized Adders

n	Design	Area(µm²)	Delay(ns)	Power(µw)	Area x Delay	Delay x Power
8	KS 8x1	58	3.184	2.04	185	6.49
	LF	209	2.34	8.64	489	20.21
	BK	196	2.21	8.21	433	18.14
	GCS 2X4	98	2.39	3.28	234	7.83
16	KS 8X2	112	4.75	5.14	532	24.41
	KS 16X1	120	5.2	4.37	624	22.7
	LF	463	2.48	24.65	1148	61.13
	BK	454	2.62	24.29	1189	63.63
	GCS 4X4	138	3.21	4.34	443	13.93
	GCS 1X16	116	3.98	3.67	461	14.6
32	KS 8X4	231	10.78	7.65	2490	82.46
	KS 32X1	244	9.23	9.59	2252	88.51
	LF	993	2.72	54.83	2701	149.13
	BK	943	2.96	52.65	2791	155.84
	GCS 8X4	203	3.67	13.76	745	50.49
	GCS 2X16	192	3.82	11.23	733	42.89
64	KS 8X8	584	5.37	26.68	3136	143.27
	KS 64X1	633	10.29	27.7	6513	285.03
	LF	1568	3.81	83.56	5974	318.36
	BK	1523	3.96	81.03	6031	320.87
	GCS 16X4	429	4.01	23.01	1720	92.27
	GCS 4X16	403	4.21	19.56	1696	82.34

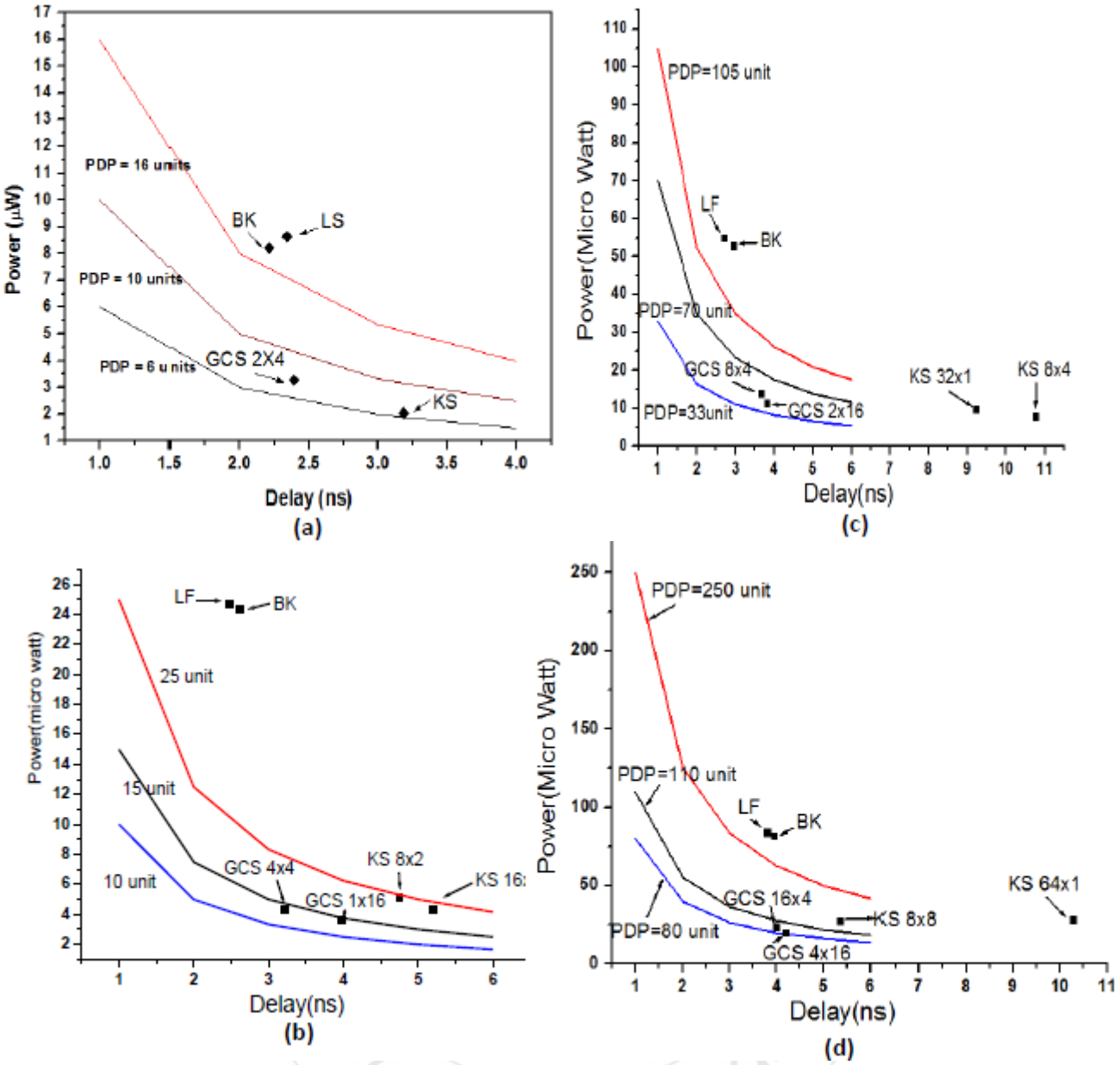

Figure 6: Illustration of the power delay product (a) 8 bit (b) 16 bit (c) 32 bit (d) 64 bit

6. Conclusion

A GCS Diminished-one modulo 2^n+1 adder has been implemented and compared with various parallel prefix adder to drive the most compromising design in terms of area, delay, power. Based on the 90 nm CMOS technology, the ASIC implementation of GCS modular adder has better area-delay and delay-power performance over those of the parallel-prefix adder.

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] 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.

[5] Su-Hon Lin and Ming-Hwa Sheu,"VLSI Design of Diminished-One Modulo 2^n+1 Adder Using Circular Carry Selection," IEEE Trans on Circuits and Systems-ll, Exp Briefs, Vol. 55, no. 9, Sep 2008.

[6] Haridimos T. vergos and Giorgos Dimitrakopoulos, Member, IEEE," On Modulo 2^n+1 Adder Design," IEEE Trans on Computers, Vol. 61, No. 2, Feb. 2012.

[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. 831-838, 1980.
[8] R. E. Ladner and M.J. Fischer ," Parallel Prefix Computation," J. ACM, Vol. 27, no. 4, pp. 30-34, 1999.
[9] 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.
[10] 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.
[11] S. Knowles, "A Family of Adders," Proc. 14th IEEE Symp. Computer Arithmetic, pp. 30-34, 1999.
[12] H. T. Vergos, C. Efstathiou, and D. Nikolos, "Diminished-one modulo 2^{n+1} adder design," IEEE Trans. Comput., vol. 51, no. 12, pp.1389–1399, Dec. 2002.