

Low Power Modulo 2^n+1 Adder Based on Carry Save Diminished-One Number System

Somayeh Timarchi, Omid Kavehei, and Keivan Navi
Department of Electrical and Computer Engineering,
Shahid Beheshti University, Tehran, Iran

Abstract: Modulo 2^n+1 adders find great applicability in several applications including RNS implementations. This paper presents a new number system called *Carry Save Diminished-one* for modulo 2^n+1 addition and a novel addition algorithm for its operands. In this paper, we also present a novel architectures for designing modulo 2^n+1 adders, based on parallel-prefix carry computation units. CMOS implementations reveal the superiority of the resulting adders against previously reported solutions in terms of implementation area and delay.

Keywords: Modulo 2^n+1 addition, carry save diminished-one number system, parallel-prefix adders, residue number system, computer arithmetic, VLSI circuits.

INTRODUCTION

The Residue Number System (RNS) is a non-weighted number system [1] that can map large numbers to smaller residues, without any need for carry propagations [2]. Arithmetic operations like addition, subtraction and multiplication can be performed on residue digits concurrently and independently. Thus, using residue arithmetic, would in principle, increase the speed of computations [3,4,5].

RNS has shown high efficiency in realizing special purpose applications such as digital filters [6,7,8,9], image processing [10], RSA cryptography [11] and specific applications for which only additions, subtractions and multiplications are used and the number dynamic range is specific.

Special moduli sets have been used extensively to reduce the hardware complexity in the implementation of converters and arithmetic operations [12,13]. Among which the triple moduli set $\{2^n-1, 2, 2^n+1\}$ have some benefits [14]. Because of operand lengths of these moduli, the operation delay is determined by the modulo 2^n+1 channel. Therefore, the design of efficient modulo 2^n+1 adders is critical [15]. Modulo 2^n+1 operations are used in many applications such as DSP algorithms [16], Fermat Number Transform for elimination of round off errors in convolution computations [17,18,19], cryptography [20] and in pseudorandom number generation [21]. Modulo 2^n+1 adders are also utilized as the last stage adder of modulo 2^n+1 multipliers.

In the last few years, several algorithms and architectures have been proposed for designing modulo

2^n+1 adders. These algorithms are based on two number systems :

- To overcome the problem of $(n+1)$ -bit wide circuits for the modulo 2^n+1 channel, the diminished-one number system [17] has been proposed. In this system, efficient adders have been reported in [14,22-25]. But these adders need a special treatment for zero operands.

For this problem, a new number representation called "Carry Save Diminished-one" (CSD-1) is proposed in this paper. With this system, the addition with zero operand doesn't need a special treatment, which reduces the adder chip area.

- Modulo 2^n+1 adders can be designed as a special case of general modulo m adders. The most efficient circuits for generalized modulo adders are reported in [15,26-28]. In [15], the proposed adder is more efficient than the ones proposed by [26-28].

However, the corresponding structure [15] uses a 3-operand adder which is eliminated in our method. In the paper, we derive a new methodology for modulo 2^n+1 adder that leads to a parallel-prefix adder architecture. Using implementation in a CMOS technology, we show that the proposed parallel-prefix design methodology uses considerably less chip area than that reported in [23] (diminished-one number system) and less chip area and propagation delay than the approach reported in [15] (normal number system).

FOUNDATION

Modulo 2^n+1 Reduction Basics : Let A be a $2n$ bit word and A^h (resp. A^l) the corresponding high (resp. low) n bit words:

Corresponding Author: Somayeh Timarchi, Faculty of Electrical and Computer Engineering, Shahid Beheshti University, Tehran, Iran, Tel: +98 21 29902286, Fax: +98 21 2417940, E-mail: s_timarchi@sbu.ac.ir

$$A = A^h 2^n + A^l$$

$$A \bmod (2^n + 1) = (A^h 2^n + A^l) \bmod (2^n + 1)$$

$$= (A^l - A^h) \bmod (2^n + 1).$$

Therefore, the reduction modulo $2^n + 1$ is computed by subtracting the high n -bit word from the low n -bit word and then conditionally adding $2^n + 1$ if the subtraction yields a negative result.

Diminished-One Number System: In the diminished-one number system, the number A is represented by $A' = A - 1$ and the value zero is treated separately, i.e., it requires an additional zero indication bit. In this system, the ordinary addition can be implemented by an end-around-carry parallel-prefix adder with $c_{in} = c_{out}$ [17,25].

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

$$= [(A'+1) + (B'+1) - 1] \bmod (2^n+1)$$

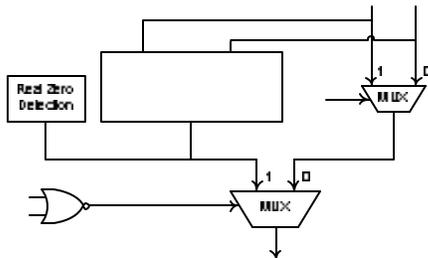
$$= (A' + B' + 1) \bmod (2^n+1)$$

$$= (A' + B' + c_{out}) \bmod 2^n \tag{1}$$

Algorithm 1: (Modulo $2^n + 1$ addition in diminished-1 number system): A number in diminished-one is represented by $n+1$ bits in which the $(n+1)^{th}$ bit is used to indicate '0'. In [17], the modulo $2^n + 1$ addition algorithm has been presented for zero and non zero operands:

- 1) If the most significant bit of one addend is '1', inhibit the addition and the other addend is the sum (Fig. 1).
- 2) If the msb of both addends are '0', ignore the msb, add the n lsb's, complement the carry and add it to the n lsb's of the sum.

The modulo $2^n + 1$ adder in Fig. 1 can be designed in different ways. To increase the modulo addition speed, the delay of carry computation should be minimized. In many papers, parallel-prefix adders have been proposed for this purpose.



1 is extendable to any other modulo when diminished-one is only defined for modulo $2^n + 1$.

CSD-1 Addition Algorithm: In this section, we present the CSD-1 addition algorithm for modulo $2^n + 1$ (Algorithm 2).

Algorithm 2: (CSD-1 addition): This algorithm is decomposed into 2 steps.

Step 1. The first step is based on the following theorem:

Theorem 1: Let A and B be two CSD-1 numbers in the range $[0, 2^n + 1]$. Then,

$$|A+B|_{2^{n+1}} = \begin{cases} \|A+B-1\|_{2^n} + 1 & \text{if } A+B-1 < 2^n \\ \|A+B-1\|_{2^n} & \text{otherwise} \end{cases}$$

Proof:

$$|A+B|_{2^{n+1}} = \begin{cases} A+B & \text{if } A+B < 2^n + 1 \\ A+B-(2^n + 1) & \text{otherwise} \end{cases} \quad (2)$$

$$= \begin{cases} (A+B-1)+1 & \text{if } A+B-1 < 2^n \\ (A+B-1)-2^n & \text{otherwise} \end{cases}$$

When $A+B-1 < 2^n$ then

$$(A+B-1)+1 = \|A+B-1\|_{2^n} + 1 \quad (3)$$

The maximal value of (3) is 2^n . In CSD-1, this value can be represented by $(n+1)$ bits in n positions. In other words, the output carry resulting from $\|A+B-1\|_{2^n} + 1$ is 0. Thus, the term (3) is transformed into:

$$\|A+B-1\|_{2^n} + 1 = \| \|A+B-1\|_{2^n} + 1 \|_{2^n}$$

Since $A, B \in [0, 2^n]$, the second case of equation (2) leads to the following inequalities:

$$A+B-1-2^n \leq 2^n + 2^n - 1 - 2^n < 2^n$$

So,

$$A+B-1-2^n = \|A+B-1-2^n\|_{2^n} = \|A+B-1\|_{2^n}$$

Therefore, from (2) we get:

$$|A+B|_{2^{n+1}} = \begin{cases} \|A+B-1\|_{2^n} + 1 & \text{if } A+B-1 < 2^n \\ \|A+B-1\|_{2^n} & \text{otherwise} \end{cases} \quad (4)$$

The equation (4) outlines the impact of the output carry of $(A+B-1)$. In the CSD-1 number system, this carry is produced when the sum is larger than 2^n . The carry generation indicates that the sum is equal or greater than the modulo. Let assume C_{out} is the output carry of $(A+B-1)$. Thus the carry of $(A+B-1)$ will be generated when:

$$c_{out} = '1' \Leftrightarrow (A+B-1) \geq 2^n + 1 \Leftrightarrow A+B \geq 2^n + 2$$

$$\Leftrightarrow A+B > 2^n + 1 \quad (5)$$

Thus, if the sum of two numbers is greater than the modulo, the output carry of $(A+B-1)$ is '1' and the sum is correct according to theorem 1: there is no need to increment the result.

The output carry is zero in the following cases:

$$c_{out} = '0' \Leftrightarrow (A+B-1) < 2^n + 1$$

$$\Leftrightarrow A+B \leq 2^n + 1 \quad \begin{cases} A+B < 2^n + 1 & (*) \\ A+B = 2^n + 1 & (**) \end{cases} \quad (6)$$

In condition (*), since $A+B$ is less than the modulo, the output carry of $(A+B-1)$ is '0'. According to equation (4), the sum should be incremented in the second stage. Therefore from (4), (5) and (*) we have: if $A+B < 2^n + 1$ or $A+B > 2^n + 1$ then:

$$S = (A+B) \bmod (2^n + 1) = (A+B-1) + \overline{c_{out}} \quad (7)$$

But in condition (**) of equation (6), when $A+B = 2^n + 1$, equation (7) leads to $S=1$, which is not true. To correct this case, we introduce step 2 of Algorithm 2 that will be presented later.

In our method, $(A+B-1)$ is computed without any extra hardware and only by ignoring a'_0 in above sum. As mentioned earlier, if $A \neq 0$ then $a'_0 = 1$; thus $(A+B-1)$ will be achieved by eliminating a'_0 . If $A = 0$ then $A+B$ will be computed by removing a'_0 . In this case, we have always $c_{out} = '0'$ and the sum will be incremented according to equation (7). But incrementing shouldn't be done to obtain the correct result.

The first step of Algorithm 2 reveals that a two-stage combinational circuit is required for modulo addition (adder and incremter). The first stage computes an intermediate sum M .

$$M = A+B-1 \quad \text{if } A \neq 0 \quad (a'_0 = 0)$$

$$M = A+B \quad \text{if } A = 0 \quad (a'_0 = 1)$$

Therefore, we adjust equation (7) as below:

$$(A+B) \bmod (2^n + 1) = M + \overline{c_{out}} + \overline{a'_0} \quad (8)$$

M in equation (8) is achieved by addition of A and B excluding a'_0 . The range of M is given by theorem 2.

Theorem 2 (The range of M): M has a $(n+1)$ -digit binary representation in CSD-1, i.e., $M \in [0, 2^{n+1}]$. (Note that $m_n = C_{out}$ in this theorem.)

Proof: If $A=0$ then $M=A+B=B$. Since $B \in [0, 2^n]$ then the theorem is established. The situation is the same if $B=0$.

If $A \neq 0$ and $B \neq 0$ then $M = A + B - 1$. Since

$$A, B < 2^n + 1, A > 0 \text{ and } B > 0 \text{ then,} \\ 0 < A + B - 1 < (2^n + 1) + (2^n + 1) - 1 \Leftrightarrow 0 < M \leq 2^{n+1} \quad (9)$$

The maximal value of M is 2^{n+1} which can be presented by $n+2$ bits or $n+1$ digits in CSD-1 (for this maximal value, all bits are '1').

In the second stage, the least n posibits of M is incremented according to (8).

Step 2: As described before, if $A + B \neq 2^n + 1$ then theorem 1 leads to equation (8). But if $A + B = 2^n + 1$ then the correct output of $S=0$ should be produced. In this case, A and B are non zero and $M = A + B - 1 = 2^n$.

According to theorem 2 and equation (8), if the msb of M , $m_n=0$ ($C_{out} = 0$) then M should be incremented in the second stage. Thus the final output is 2^n+1 . In CSD-1, each number is in the range of $[0, 2^n]$ and can be represented by n digits. Therefore the output carry can be ignored and the output sum is "0...01" that can be corrected by inverting s'_0 .

In the second step of Algorithm 2, we introduce two methods to detect zero output and to correct it.

- a) The correct output zero occurs when two inputs are complementary, i.e. their sum is equal to modulo 2^n+1 . One method to recognize complementary numbers is the logical AND of the outputs of a_i XOR b_i (for any i except $i=0$). A similar method has been mentioned in [23].
- b) Another method is based on the following theorem.

Theorem 3: (Complementary of two inputs): Two inputs are complementary when (and only when) the input and output carries of the incrementer are '1'.

Proof:

- o First, we prove that if A and B are complementary numbers, the input and output carries of the incrementer are '1'. When A and B are complementary, both of them are non zero.

Therefore, $M = A + B - 1 = (2^n + 1) - 1 = 2^n$. In

CSD-1, 2^n has n digits. Thus, $c_{out} = 0 \Rightarrow \overline{c_{out}} = 1$.

The input carry of the incrementer is $c_{in}^* = \overline{c_{out}} \cdot a'_0 = 1$.

The output carry of the incrementer is equivalent to the output carry of the following addition:

$$S = (A + B - 1) + 1 = A + B = 2^n + 1$$

Obviously, the output carry is '1'.

- o Now we prove that if input and output carries of the incrementer are '1' then A and B are complementary numbers.

If $c_{in}^* = 1$ then $\overline{(c_{out} \cdot a'_0)} = 1$. Therefore $A \neq 1$ and $c_{out} = 0$. In other words, $M = A + B - 1$ and we have:

$$M = A + B - 1 \leq 2^n \Rightarrow A + B \leq 2^n + 1 \quad (10)$$

The output carry of the incrementer is '1' when the sum is equal or more than 2^n+1 . That is:

$$(A + B - 1) + 1 \geq 2^n + 1 \Rightarrow A + B \geq 2^n + 1 \quad (11)$$

Equations (10) and (11) are simultaneous verified when $A+B = 2^n+1$, which shows that A and B are complementary.

Method (a) has been used in [23]. However the method (b) for zero detection and correction consumes less area than method (a). Then, we implemented method (b). As described earlier and according to example 1, s'_0 can be transformed to '0' in the condition of zero detection.

THE PROPOSED CSD-1 PARALLEL-PREFIX ADDER (CSD-PP)

One way for implementing the CSD-PP adder is based on the adder architecture of Fig. 2. But instead of having a dedicated single stage for reentering the carry, [23] has proposed to perform carry recirculation at each existing prefix level. Then, there is no need for the extra carry increment stage. As a result, a dedicated CSD-PP adder architecture is derived with one less prefix level compared to those derived from Fig. 2 architecture. In the CSD-1 system, it requires several modifications. These modifications will be introduced by the 3 following theorems.

Theorem 4: Let assume that $(\overline{G}, P) = (\overline{G}, P)$ and

$G_{a,b}$ and $P_{a,b}$, with $a > b$, are respectively the group generate and propagate signals for the group $a, a-1, a-2, \dots, b-1, b$, computed by:

$$(G_{a,b}, P_{a,b}) = (g_a, p_a) \circ (g_{a-1}, p_{a-1}) \circ \dots \circ (g_b, p_b)$$

In our case, in which the reentering carry is given by the expression $a'_0 \overline{G_{n-1}}$, the carries c_i^* of the addition modulo 2^i+1 are equal to G_i^* , where G_i^* is computed by the prefix equations:

$$(G_i^*, P_i^*) = \begin{cases} (\overline{G_{n-1}}, P_{n-1}) & i = -1 \\ (G_i, P_i) \circ (a'_0 \cdot \overline{G_{n-1,i+1}}, P_{n-1,i+1}) & \text{if } 0 \leq i \leq n-2 \end{cases}$$

Proof:

$$\begin{aligned} (G_i^*, P_i^*) &= (G_i, P_i) \circ (\overline{G_{n-1}} \cdot a'_0, P_{n-1}) \\ &= (G_i + P_i \overline{G_{n-1,i+1}} + P_{n-1,i+1} \cdot G_i, a'_0, P_{n-1}) \\ &= (G_i + P_i \overline{G_{n-1,i+1}} (\overline{P_{n-1,i+1}} + \overline{G_i}), a'_0, P_{n-1}) \\ &= (G_i + P_i \overline{G_{n-1,i+1}} \overline{P_{n-1,i+1}} \cdot a'_0 + P_i \overline{G_{n-1,i+1}} \overline{G_i} \cdot a'_0, P_{n-1}) \\ &= (G_i + P_i \overline{G_{n-1,i+1}} \cdot a'_0 + P_i \overline{G_{n-1,i+1}} \overline{G_i} \cdot a'_0, P_{n-1}) \\ &= (G_i + P_i \overline{G_{n-1,i+1}} \cdot a'_0, P_{n-1,i+1}) \\ &= (G_i + P_i \overline{G_{n-1,i+1}} \cdot a'_0, P_{n-1,i+1}) \\ &= (G_i, P_i) \circ (\overline{G_{n-1,i+1}} \cdot a'_0, P_{n-1,i+1}) \end{aligned}$$

Y

Theorem 5 will derive expressions leading to faster circuits.

Theorem 5: Defining $p_0^* = p_0 \cdot a'_0$ leads to

$$(G_i^*, P_i^*) = (g_i, p_i) \circ \dots \circ (g_0, p_0^*) \circ (\overline{G_{n-1,i+1}}, P_{n-1,i+1})$$

Proof:

$$\begin{aligned} (G_i^*, P_i^*) &= (g_i, p_i) \circ \dots \circ (g_0, p_0) \circ (\overline{G_{n-1,i+1}} \cdot a'_0, P_{n-1,i+1}) \\ &= \left(\begin{aligned} &(g_i + p_i g_{i-1} + \dots + p_i p_{i-1} \dots p_1 p_0 \overline{G_{n-1,i+1}} \cdot a'_0) \\ &(p_i p_{i-1} \dots p_1 p_0 P_{n-1,i+1}) \end{aligned} \right) \end{aligned}$$

When computing G_i^* , only the last term includes p_0 and a'_0 . Therefore, we can define $p_0^* = p_0 \cdot a'_0$ and replace p_0 by p_0^*

$$\begin{aligned} (G_i^*, P_i^*) &= (g_i, p_i) \circ \dots \circ (g_0, p_0^*) \circ (\overline{G_{n-1,i+1}}, P_{n-1,i+1}) \\ &= \left(\begin{aligned} &(g_i + p_i g_{i-1} + \dots + p_i p_{i-1} \dots p_1 p_0^* \overline{G_{n-1,i+1}}) \\ &(p_i p_{i-1} \dots p_1 p_0^* P_{n-1,i+1}) \end{aligned} \right) \end{aligned} \tag{12}$$

The final P_i^* are never used and the intermediate P_i don't have p_0 . The above equations are thus correct. Y

In several cases, the equations (12) require more than $\log_2 n$ prefix levels for their implementation. These equations can be transformed into equivalent ones that can be implemented within $\log_2 n$ prefix levels. The required transformation uses Theorem 2 of [23], as well as the Theorem 6 that will be introduced below. Theorem 2 of [23] says that,

$$(g, p) \circ (\overline{G, P}) = (\overline{p, g}), (\overline{G, P})$$

This implies that a carry equal to the generate term which is expressed by a prefix equation of the form $(g, p) \circ (\overline{G, P})$ is also equal to the generate term of an equation of the form $(\overline{p, g}), (\overline{G, P})$.

The above formula is true when $g \cdot p = p$.

The following theorem is also required to derive the term that has the form $(g_0, p_0^*) \circ (\overline{G, P}) = (\overline{p_0^*, g_0}), (\overline{G, P})$ in prefix notation:

Theorem 6: If $(G_x, P_x) = (g_0, p_0^*) \circ (\overline{G, P})$ and

$$(G_y, P_y) = (\overline{p_0^*, g_0}), (\overline{G, P}) \text{ then } G_x = G_y.$$

Proof: First, we proof the following expression:

$$\begin{aligned} g_0 \cdot p_0^* &= (g_0 + p_0^*) = \overline{a_0'' b_0'' + (a_0'' b_0' + b_0'' a_0')} \\ &= \overline{a_0'' b_0' + b_0'' (a_0' + a_0')} = \overline{a_0'' b_0' + b_0'' a_0'} = \overline{p_0^*} \end{aligned} \tag{13}$$

Using this formula, we get:

$$\begin{aligned} g_0 + p_0^* \overline{G} &= \overline{(g_0 + p_0^* \overline{G})} = \overline{(g_0 \cdot (\overline{p_0^*} + \overline{G}))} \\ &= \overline{(g_0 \cdot \overline{p_0^*} + g_0 \cdot \overline{G})} = \overline{(p_0^* + g_0 \cdot \overline{G})} \end{aligned}$$

Y

The carry equations resulting from theorem 2 of [23] and theorem 6 can be implemented by a prefix structure that has $\log_2 n$ levels. As mentioned earlier, we use the modifications introduced by theorems 4 to 6. Our proposed adder is similar to [23] modulo adder architecture but its first cells of preprocessing and post processing stages are designed differently.

In the CSD-1 number system, if $x'_0 = 0$ then $x''_0 = 0$.

This is a special property of CSD-1. Using this property to simplify truth tables of these two cells leads to the following equations:

$$s_0'' = a_0' b_0' \overline{b_0''} + \overline{a_0''} b_0' b_0'' = b_0' (a_0'' \oplus b_0'')$$

$$p_0^* = p_0 \cdot a_0' = a_0'' b_0' + a_0' b_0''$$

$$g_0 = a_0' b_0''$$

$$s_0'' = s_0'' c_{in}^* + s_0^* \overline{s_0''} c_{in}^*$$

Theorem 3 is used for the detection and generation of a correct zero. The term r indicates the condition of theorem 3:

$$c_{-1}^* = 1 \rightarrow \overline{G_{n-1}} \cdot a'_0 = 1$$

$$c_{out}^* = 1 \rightarrow h_{n-1} \cdot c_{n-2}^* + g_{n-1} = 1$$

$$r = \overline{c_{-1}^* \cdot c_{out}^*} = \overline{(\overline{G_{n-1}} \cdot a'_0)(h_{n-1} \cdot c_{n-2}^* + g_{n-1})}$$

RESULTS AND COMPARISONS

In this section, we compare the proposed CSD-PP adder to those proposed in [15] and [23]. As previously mentioned, the architecture proposed in [23] outperforms those presented in [24] and [25], and the architecture proposed in [15] outperforms those presented in [26-28] in terms of implementation area and execution delay. Thus, the architecture of [23] is the best diminished-one architecture, and the architecture of [15] is the best architecture using normal binary representation.

All architectures were described in HSPICE and mapped to the 0.18 implementation technology (0.18 μm , Vdd=1.8 v). We use VLSI implementations and a simple model to compare the proposed adder architectures to those proposed in [15] and [23]. We use the notation PPREF for the diminished-one modulo 2^8+1 adder proposed in [23] and TPP for the normal binary one in [15]. The CSD-PP implementation for the modulo 2^n+1 adder is given in Fig. 2.

Analytical Comparisons and Results: First, we use the analytical model used in [15] and [23], under the notation “unit-gate model”. This model assumes that each gate, except the exclusive-OR gate, counts as one elementary gate for both area and delay. An exclusive-OR gate counts for two elementary gates for both area and delay. According to this model, the latencies of the modulo 2^n and modulo $2^n - 1$ adders are equal to $2 \cdot \log_2 n + 3$. The PPREF modulo adder has an execution latency of $2 \cdot \log_2 n + 3$.

However, according to Fig. 1, the overall delay of PPREF is the modulo adder latency plus the multiplexer delay. The multiplexer is a 2-level circuit in unit-gate

model. The overall delay is $2 \cdot \log_2 n + 5$. The TPP adder has a latency equal to $2 \cdot \log_2 n + 6$ and the proposed CSD-PP adder has a latency equal to $2 \cdot \log_2 n + 4$. The CSD-PP architecture is faster than PPREF and TPP. Therefore, the CSD-PP adder offers the fastest designs reported in the open literature. The CSD-PP adder has also the same prefix levels as the PPREF adder, without requiring any circuits for treating zero operands as shown in Fig. 1, which reduce both the execution time and the implementation area. Therefore, the proposed CSD-PP adders are more efficient than the fastest modulo 2^n+1 adder which handle operands in diminished-one representation. The normal binary system can be easily converted to the normal binary RNS. The representation of odd numbers in CSD-PP adders is the same as in TPP adders.

According to the unit-gate model, the hardware overheads of the fastest reported modulo 2^n and modulo $2^n - 1$ adders are respectively equal to $1.5 n \cdot \log_2 n + 5n$ and $3 n \cdot \log_2 n + 5n$. The PPREF modulo adder has an area of $4.5 n \cdot \log_2 n + 0.5n + 6$. However, according to Fig. 1, the final area of PPREF includes the modulo adder area and the area of circuit for the treatment of zero operands. The zero operand circuit area is $2n+5$. Thus, the final area is $4.5 n \cdot \log_2 n + 2.5n + 11$. The area of the TPP adder is equal to $4.5 n \cdot \log_2 n + 3.5n + 13$ and the proposed CSD-PP adder area is equal to $4.5 n \cdot \log_2 n + 0.5n + 15$.

Real Comparisons and Results: For evaluating the speed, area and power consumption efficiencies of each architecture, every adder is implemented by CMOS technology. The obtained results are listed in Table 2. As we can see proposed architecture leads to far faster implementations than that of [15] and [23]. This is due to the fact that the architecture of [15] requires a delay of one CSA unit and the design of PPREF in [23] uses some multiplexers to treat zero operands. The proposed architecture, on the other hand, relies on a 2-operand addition (in adverse of TPP that adds two inputs and $2^n - 1$) and requires unique circuit for zero and non zero operands based on CSD-1 number system.

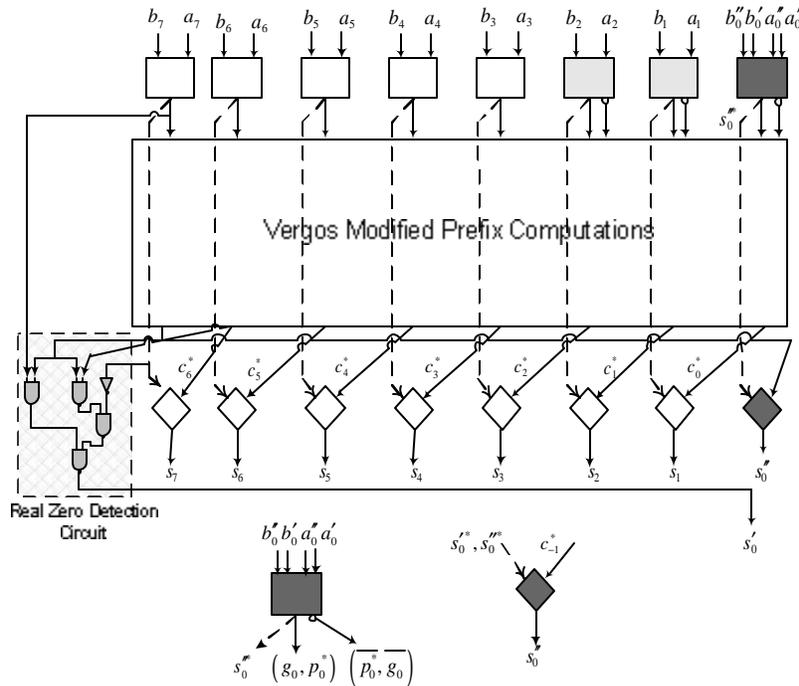


Fig. 2: Proposed modulo 2^n+1 parallel-prefix carry save diminished-one adder

Table 2: Real Comparison Results

Adder Architecture	Transistor Count	Average Power Consumption (μW)	Delay (ps)	Power-Delay Product (fJ)
PPREF ^[23]	1036	293.56	434.44	127.54
TPP ^[15]	844	278.64	562.74	156.80
CSD-PP	838	214.51	235.59	50.53
Improvement CSD-PP vs. PREF	> 19%	< 27%	> 45%	> 60%
Improvement CSD-PP vs. TPP	> 0.7%	> 23%	> 58%	> 67%

Finally, we study power consumption of compared architectures. The simulation results are shown in Table 2. It is obvious that the proposed CSD-PP adder has the lowest consumption of all. It improves TPP and PPREF power consumptions above 23% and 26% respectively.

CONCLUSIONS

In this paper, a new number system has been presented. This paper also presents a new architecture for modulo 2^n+1 adders that uses parallel-prefix computation units based on mentioned number system. The proposed architecture has better performance than

the conventional modulo 2^n+1 adders. The main points of the paper are summarized below:

1. The special treatment required for zero operands in the diminished-one number system has been removed.
2. The proposed architecture removes the 3-operand adder issue in the fastest modulo 2^n+1 adders with the normal binary system.
3. The proposed architecture leads to the fastest reported modulo 2^n+1 adders, with execution latencies close to the execution latency of the fastest modulo 2^n and modulo 2^n+1 adders, which means that the proposed architecture is suitable for RNS applications.

REFERENCES

1. Behrooz Parhami., 2000. Computer arithmetic: algorithms and hardware designs, Oxford.
2. Garner, H., 1959. The residue number system, IRE Trans. On Electronic Computer, vol. EC-8, pp.140-147.
3. Kouretas I., and V. Paliouras, 2005. High-radix redundant circuits for RNS modulo r^n-1 , r^n , or r^n+1 , The International Symposium on Circuits and Systems (ISCAS '03), vol. 5.
4. Hosseinzade M., S. Timarchi and K. Navi, 2007. Multi Level Residue Number System with Moduli Set of $(2^n, 2^n-1, 2^{n-1}-1)$, 12th International CSI Computer Conference (CSICC'2007).
5. Timarchi S., K. Navi and M. Hosseinzade, 2006. New Design of RNS Subtractor for modulo $(2^n + 1)$, 2nd IEEE International Conference on Information & Communication Technologies: From Theory To Application.
6. Freking W.L. and K.K. Parhi, 1997. Low-power FIR digital filters using residue arithmetic, 31st Asimolar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, vol. 1, pp. 739-43.
7. Fernandez P. G., et al., 2000. A RNS-Based Matrix-Vector-Multiply FCT Architecture for DCT Computation, Proc. of the 43rd IEEE Midwest Symposium on Circuits and Systems, pp. 350-353.
8. Parhami B., 1996. A Note on Digital Filter Implementation Using Hybrid RNS-Binary Arithmetic, Signal Processing, vol. 51, pp. 65-67.
9. Re A. D., A. Nannareli and M. Re, 2004. A Tools for Arithmetic Generation of RTL-Level VHDL Description of RNS FIR Filters, IEEE Proceeding of the Design, Automation and Test in Europe Conference and Exhibition (DATE).
10. Bhardwaj, M. and B. Ljusanin, 1998. The Renaissance – A Residue Number System Based Vector Co-Processor for DSP Dominated Embedded ASICs, Proc. Asimolar Conference on Signals, Systems, and Computers, pp. 202-207.
11. Yen S., S. Kim, S. Lim and S. Moon, 2003. RSA Speedup with Chinese Remainder Theorem Immune against Hardware Fault Cryptanalysis, IEEE Trans. On Computers, vol. 52, no. 4, pp. 461-472.
12. Hariri A, K. Navi, Rastegarpanah, 2007. A new High Dynamic Range Moduli Set with Efficient Reverse Converter, to appear in Computers & Mathematics with Applications journal (Elsevier).
13. Sabbagh A., K. Navi, 2007. An improved Residue to Binary converter for the RNS with Pairs of conjugate moduli, International Conference on Electrical engineering and Informatics.
14. Efstathiou C. et al., 2001. On the Design of Modulo 2^n-1 Adders, Proc. of the Eighth IEEE Int'l Conference on Electronics, Circuits & Systems, pp. 517-520.
15. Efstathiou C., H. T. Vergos and D. Nikolos, 2004. Fast Parallel-Prefix 2^n+1 Adder, IEEE Trans. On Computers, vol. 53, no. 9.
16. Taylor F., 1985. A Single Modulus ALU for Signal Processing, IEEE Trans. on Acoustics, Speech, Signal Processing, vol. 33, pp. 1302-1315.
17. Leibowitz L. M., 1976. A Simplified Binary Arithmetic for the Fermat Number Transform, IEEE Trans. on Acoustics, Speech, Signal Processing, vol. 24, pp. 356-359.
18. Sunder S. et al., 1993. Area-Efficient Diminished-1 Multiplier for Fermat Number-Theoretic Transform, IEE Proc. G, vol. 140, pp. 211-215.
19. Truong T. K. et al., 1986. Techniques for Computing the Discrete Fourier Transform Using the Quadratic Residue Fermat Number Systems, IEEE Trans. On Computers, vol. 35, pp 1008-1012.
20. Zimmermann R. et al., 1994. A 177 Mb/s VLSI Implementation of the International Data Encryption Algorithm, IEEE Journal of Solid-State Circuits, vol. 29, no. 3, pp. 303-307.
21. Lehmer D.H., 1951. Proc. of the Second Symposium on Large-Scale Digital Calculating Machinery (Cambridge, MA: Harvard University Press), pp. 141-146.
22. Vergos H.T., C. Efstathiou, and D. Nikolos, 2001. High Speed Parallel-Prefix Modulo 2^n+1 Adders for Diminished-One Operands, IEEE Proce. Of 15th IEEE Symposium on Computer Arithmetic, pp. 211 – 217.
23. Vergos H.T., C. Efstathiou, and D. Nikolos, 2002. Diminished-One Modulo 2^n+1 Adder Design, IEEE Trans. On Computers, vol. 51, pp. 1389-1399.
24. Zimmermann R., 1997. Binary Adder Architectures for Cell-Based VLSI and Their Synthesis, PhD thesis, Swiss Federal Institute of Technology.
25. Zimmermann R., 1999. Efficient VLSI Implementation of Modulo $(2^n \pm 1)$ Addition and Multiplication, Proc. of the 14th IEEE Symposium on Computer Arithmetic (ARITH-14), Adelaide, Australia, pp. 158-167.
26. Bayoumi M. and Jullien G., 1987. A VLSI Implementation of Residue Adders, IEEE Trans. on Circuits and Systems, vol. 34, pp. 284-288.
27. Dugdale M., 1992. VLSI Implementation of Residue Adders Based on Binary Adders, IEEE Trans. on Circuits and Systems II, vol. 39, pp. 325-329.
28. Hiasat A.A., 2002. High-Speed and Reduced Area Modular Adder Structures for RNS, IEEE Trans. on Computers, pp. 84-89.
29. Brent R.P. and H.T. Kung, 1982. A Regular Layout for Parallel Adders, IEEE Trans. on Computers, vol. 31, no. 3, pp. 260-264.
30. Kogge P.M. and H.S. Stone, 1973. A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations, IEEE Trans. On Computers, vol. 22, no. 8, pp. 783-791.