International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org || PP.17-24

Design of Novel Digital Adder Design Based On Residue Number System

1,P. Rajender , 2,R.Srinivas
1,Pg Scholar, Vlsi , Jyothismathi Institute Of Technolgy And Science, Karimnagar, Telangana, India
2Assistant Professor, Electronics & Communication System, Jyothismathi Institute Of Technolgy And Science, Karimnagar, Telangana, India

ABSTRACT: *Modular adder is one of the key components for the application of residue number system (RNS). Module set with the form can offer excellent balance among the RNS channels for multi-channels RNS processing. In this paper, a novel algorithm and its VLSI implementation structure are proposed for modulo2^n -2^k-1 adder. In the proposed algorithm, parallel prefix operation and carry correction techniques are adopted to eliminate the re-computation of carries. Any existing parallel prefix structure can be used in the proposed structure. Thus, we can get flexible tradeoff between area and delay with the proposed structure. Compared with same type modular adder with traditional structures, the proposed modulo 2^n -2^k-1 adder offers better performance in delay and area.*

INDEX TERMS: *Carry correction, modular adder, parallel prefix, residue number system (RNS), VLSI*

I. INTRODUCTION

Reduce number system (RNS) is an ancient numerical representation system. It is recorded in one of Chinese arithmetical masterpieces, the Sun Tzu Suan Jing, in the 4th century and transferred to European known as Chinese Remainder Theorem (CRT) in the 12th century. RNS is a non-weighted numerical representation system and has carry-free property in multiplication and addition operations. In recent years, it has been received intensive study in the very large scale integration circuits (VLSI) design for digital signal processing (DSP)systems with high speed and low power consumption [1]–[4].For the general modular adder, Bayoumi proposed a scheme for arbitrary modulus by using two cascaded binary adders [5].However, the delay is the sum of the two binary adders. Several literatures constructed several modular adders with two parallel binary adders to calculate A+B and A+B+T [6], [7]. This method can achieve less delay but needs about twice area of binary adder. Dugdale proposed a method to construct a type of general modular adders with a reused binary adder [9]. The shortage of this structure is that it will use two operation cycles to perform one modular addition. The area or delay of these modular adders mentioned above is twice or more than that of binary adder. In recent studies, a few modular adders with better area and delay performance are presented. Hiasat proposed a class of modular adders in which any regular Carry Look-Ahead(CLA)—based binary adder can be used in the final stage [10].However, it needs an extra CLA unit to get the carry-out bit of A+B+T before the final CLA addition. As a result, the structure does not reduce the delay significantly. The ELMMA algorithm proposed by Platelet al.[11] uses two carry computation modules for A+B and A+B+T in which some carry computation units can be shared. The area reduction of this scheme dominated by the form of T . In the worst case, almost two independent carry generation modules are needed. Patel et al.[12] also proposed several algorithms which can generate carries fast. A new number representation for modulo addition is proposed in [8]. However, its outputs are represented in special format. Thus, the extra area and delay are needed to perform the conversion from the special representation to binary number representation or all operations should be performed in this number representation format in RNS-based systems. On the other hand, the complexity of the special modular adder is much less than that of general modular adder, since the structure of the special modular adder can be further optimized according to the modulus.

II. PROPOSED MODULO 2^N-2^K-1 ADDER

In this paper, a new class of modulo 2^n-2^k-1 adder based on carries correction and parallel prefix algorithm is proposed. The new modular adder can be divided into four units, the pre-processing unit, the prefix computation unit, the carry correction unit, and the sum computation unit. In the proposed scheme, the carry information of A+B+T computed by prefix computation unit is modified twice to obtain the final carries required in the sum computation module. Meanwhile, any existing fast prefix structure of binary adder scan be used in the proposed modular adder structure, which offers superior flexibility in design. In order to evaluate the performance of the proposed modular adder in this paper, the unit-gate model and Design Compiler (DC) of Synopsys Company are used to estimate its complexity and performance.

As shown in Fig. 2, the proposed modulo 2^n-2^k-1 adder is composed of four modules, pre-processing unit, carry generation unit, carry correction unit, and sum computation unit. In Fig. 2, different shade represents different processing units.The proposed modular adder can be divided into two general binary addersA1, and A2 in Fig. 2, with carry correction and sum computation module according to the characteristics T of correction for modulus 2^n-2^k-1 . We can get the carries c_i^{real} used in the final stage through correcting the carries $c_i^t$of A+B+T , which can be computed by any existing prefix structure with proper pre-processing.

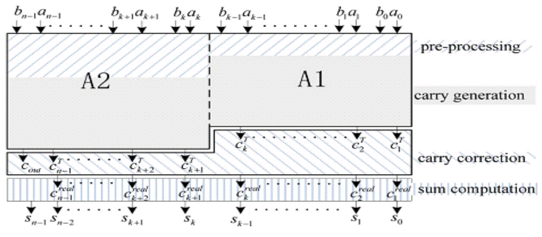


Fig. 2. The proposed modulo $2^n - 2^k - 1$ adder structure.

At last, we can get the final modular addition result from c_i^{real} and partial sum information. The proposed architecture shown in Fig. 2 can avoid the calculation of carries information for A+B+T and A+B separately. Thus, the area and delay in VLSI implementation can be reduced. Meanwhile, the proposed scheme offers flexible trade of area and delay with different parallel prefix structures.

2.1.1. Pre-Processing Unit

The pre-processing unit is used to generate the carry generation and carry propagation bits(g_i,p_i) of A+B+T .From(3),

When m=2^n-2^k-1

$$T = 2^{\lceil \log_2 2^n - 2^k - 1 \rceil} - m = 2^k + 1. \quad (8)$$

Obviously, the binary representation of Tis“ ”.

In Fig. 2, the computation of A+B+T can be performed by A1and A2 where and are used for lower-K bits and higher-n-k bits addition, respectively. Let T_{A1}= ,T_{A2}=and the binary representations of A and B bea$_{n-1}$…$a_k a_{k-1}$..$a_1 a_0$and b_{n-1}…$b_k b_{k-1}$..$b_1 b_0$ respectively. The operation of adder and can be regarded as

$$\begin{cases} S_{A1} = a_{k-1} \ldots a_0 + b_{k-1} \ldots b_0 + T_{A1} \\ S_{A2} = a_{n-1} \ldots a_k + b_{n-1} \ldots b_k + T_{A2} + c_{A1} \end{cases} \quad (9)$$

Where c_{A1} is the carry-out bit of adder A1 . For T_{A1} , one of the inputs of A1, every bit is “0” except the least significant bit. Thus A1, can be treated as a k-bit adder with he lowest carry-in bit, which is exactly as same as the general binary adder. And the way pre-processing of T_{A1} is also similar with the general binary adder. The difference is that the lowest carry-in bit should be considered. Therefore, carry generation and carry propagation bits are

$$\begin{cases} (g_0, p_0) = (a_0 + b_0, \overline{a_0 \odot b_0}) & i = 0 \\ (g_i, p_i) = (a_i b_i, a_i \oplus b_i) & i = 1, 2, \ldots, k-1. \end{cases} \quad (10)$$

For adder A2, it does not only add the constant T_{A2}, but also the carry-out bit C_{A1} from adder A1. It can be regarded as a three- inputs adder with the lowest carry-in bit. The three inputs are a_{n-1}…a_k, b_{n-1}…b_k , and T_{A2} in binary. In this paper,

We reduce the number of inputs from three to two for adder A2 by using Simple Carry Save Adder (SCSA). When i=k,k-1,…n-1

, we can get (g_i^1,p_i^1) for a_i and b_i , firstly

$$(g_i', p_i') = (a_i b_i, a_i \oplus b_i). \quad (11)$$

And then (g_i^1,p_i^1) is treated as the inputs of the second stage in SCSA. The second stage of SCSA generates the carry generation and carry propagation bits (g_i^1,p_i^1) from $T_{A2}$and .Actually, it is the carry saved addition of these two binary numbers $p_{n-1},..p_{k+1}p_k$and$g_{n-2}…g1_{k+1}g^1_k$ .Thus, the final outputs of pre-processing unit for adder A2 are

$$\begin{cases} (g_k, p_k) = \left(p'_k, \overline{p'_k}\right) & i = k \\ (g_i, p_i) = \left(p'_i g'_{i-1}, p'_i \oplus g'_{i-1}\right) & i = k-1, \dots, n-1. \end{cases} \tag{12}$$

From (10) and (12), all of the information required in the prefix computation is obtained. Furthermore, the carry-out bitof SCSA, c_{scsa} , is required to compute the carry-out bit of A+B+T, c_{out}. It is calculated as

$$c_{SCSA} = a_{n-1}b_{n-1} = g'_{n-1}. \tag{13}$$

Carry Generation Unit : In carry generation unit, the carries c_i^ti=(1,2,…n) of A+B+T can be obtained with the carry generation and carry propagation bits from the pre-processing unit. Any existing prefix structure can be used to get the carries c_i^t in this paper. It is worth pointing out that the carry-out bit of SCSA in the pre-processing unit, as shown in (13), is not involved in the prefix computation. Instead c_{SCSA}, combined with the carry-out bit of the prefix tree is required to determine the carry-out bit of A+B+T (denoted as c_{out}).

$$\begin{aligned} c_{out} &= c_{SCSA} + c_n^T = c_{SCSA} + G_{n-1:0} \\ &= c_{SCSA} + G_{n-1:l} + P_{n-1:l}G_{l-1:0} \\ &= c_{SCSA} + G_{n-1:l} + P_{n-1:l}c_l^T \end{aligned} \tag{14}$$

where0<Ln-1.

Carry Correction Unit : The carry correction unit is used to get the real carries c_i^{real} for each bit needed in the final sum computation stage. In order to reduce the area, we get the carries of A+B by correcting the carries of A+B+T in the carry correction unit. We first derive the relation of c_i^0 and c_i^1(i= 0,1,2,…n)in binary addition in Theorem 1, where c_i^0 andc_i^1 are the carry outputs of prefix tree when the lowest carry in is “0” and “1”,respectively.

Theorem 1:Let c_i(i=0,1,2,….n)be the carry bits of an n-bit adder, and they will be propagated to the higher adjacent positions, c_{in} be the lowest carry in (that is c_n=c_o), and c_{out} be the final carry-out bit (that is,c_{out}=c_{in}). Assuming the carries for each bit be $c_i^0$when$c_{in=0}$andthe carries for each bit be $c_i^1$when, we can get the relationship

$$c^0_{i+1} = \overline{P_{i:0}}\, c^1_{i-1} \quad (i = 0, 1, 2, \dots, n-1).$$

Proof: Let$a_{n-1}a_{n-2…}.a_1a_0$and $b_{n-1}b_{n-2}….b_1b_0$ be the binary representations of A and B, respectively. Then, we have p_i=aib$_i$,g_i=a_ib_i and $p_{i:0}$=$p_i….p_1p_0$ .According to the parallel prefix algorithm, we have

$$c_{i+1} = G_{i:0} + P_{i:0}c_{in}$$

This can be rewritten as,

$$\begin{cases} c^0_{i+1} = G_{i:0} & c_{in} = 0 \\ c^1_{i+1} = G_{i:0} + P_{i:0} & c_{in} = 1 \end{cases}$$

If $p_{i:0\,=1}$,thena$_{ib}$i , which yields g_i=a_ib_i=0And $G_{i:0}$=0 . Thus, we have c_{i+1}=c_{in}. That is c^0_{i+1}=0, c^1_{i+1}=1.If $p_{i:0\,=0}$, it means that c_{in} can’t be propagated to c_{i+1}.
Hence c_{i+1}=$G_{i:0}$, , which is irrelevant with c_{in} . That means
C^0_{i+1}=c^1_{i+1}.Thus, c^0_{i+1}=$p_{i:0}c^1_{i+1}$ .Q. E. D.

Theorem 1 means that c_i^0 can be determined from c_i^1 by simple logic operation. That is the foundation of the carry correction for the proposed modular adder. We present the procedure of the carry correction in our scheme based on Theorem1 as following. For the proposed modulo 2^n-2^k-1adder,T=2^k+1and can be represented as in binary.

The computation of A+B+T can be divided into two steps,

s_A=A+B+and .s_B=s_{A+}

Thetwo"1"bitsin T's binary representation can be regarded as the carry-in bits for adderA1 and A2 adder shown in Fig. 2,respectively. Correspondingly, the carry bits of A+B can be obtained with twice carry corrections of A+B+T based on Theorem 1. The first correction result is the carries of A+B+. The second correction result is the carries of A+B. Whether carry correction is performed or not depends on the carry-out bit of A+B+T, that is, c_{out} in (14).

Carry Correction for Adder

Since the binary representation of T isc_i^t, can be regarded as the carry bits of (A+B+T-1)+ci and c_{in}=1.Thereforec_i^T, can be modified with Theorem 1 to determine the carry bits ofc_i^{T-1}(i=0,1,2,…….n-2)of A+B+T-1,thatis

$$c_{i+1}^{T-1} - \overline{P_{i:0}} c_{i+1}^{T}. \qquad (15)$$

One point must be paid attention to perform (15). The lowest propagation bit in $p_{i:0}$,p_0 , is not equal to a_0b_0 that in(10). Actually, it is equal to .According to Theorem 1, the carries of A+B+T is corrected under the condition of c_{out}=0 . We can use a 2-to-1 Multiplexer (MUX) to perform the operation. For his MUX, c_{out} is the control signal, while c_i^T and c_i^{T-1} are input signals. And the output is the result of the first correction, denoted as c^{ci}_{i+1}(i=0,1,2,…..n-2)

$$\begin{aligned} c_{i+1}^{c1} &= \overline{c_{out}} c_{i+1}^{T-1} + c_{out} c_{i+1}^{T} = \overline{c_{out}}\,\overline{P_{i:0}} c_{i+1}^{T} + c_{out} c_{i+1}^{T} \\ &= c_{i+1}^{T}\left(c_{out} + \overline{P_{i:0}}\right). \end{aligned} \qquad (16)$$

Carry Correction for A2

From (16), c_i^{c1}(i=0,1,2,…..n-2) is the carry information of A+B+TorA+B+T-1 after the correction for adder A1 . Then we can perform the second correction based on c_{out} and let the carry bits of the second correction be c_i^{real} . Similar to the first correction, c_i^{real} is the carry of A+B+T-1-2^k (that is, A+B) when c_{out}=0.Otherwise, is the carry of .That is, is the $c_i^{rea\,l}$ final carry information needed in sum computation unit.

When i=1,2,…k ,T=2^k+1 . The bit "1" in T_{A2} will not affect c^{ci}_i . Hence,

$$c_i^{real} = c_i^{c1}. \qquad (17)$$

When i=k+1,….n-1 , the inputs of adder in Fig. 2

Are $p_{n-1}….p_{k+1}p_k$ and $g_{n-2}…g_{k+1}g_k$. And the carry-in bit is the carry-out bit of adder A1,that is c^{ci}_k, . Considering the least significant bit of $g_{n-2}…g_{k+1}g_k$ is "1", we can treat the operation of adder A2 as the addition of two inputs,$p_{n-1}….p_{k+1}p_k$and $g_{n-2}…g_{k+1}g_k$, with the lowest carry-in bit "1". That is, the results and carry information of (a),(b), in (18) are identical

$$\begin{cases} p'_{n-1}\cdots p'_{k+1}p'_k + g'_{n-2}\cdots g'_{k+1}g'_k 1 + \underbrace{00\ldots c_k^{c1}}_{(n-k)bit} & (a) \\ p'_{n-1}\cdots p'_{k+1}p'_k + g'_{n-2}\cdots g'_{k+1}g'_k c_k^{c1} + \underbrace{00\ldots 1}_{(n-k)bit} & (b). \end{cases} \qquad (18)$$

Thus, we can get the carries of A+B by modifying the carries of c^{c1}_i(i=k+1…,n-1)adder A2 with Theorem 1.Combined with the final carry-out bit of A+B+T, c_{out} ,the carries c^{real}_i required by the proposed modular adder are determined. Since the second carry correction is performed under the condition that the lowest carry-in bit of adder A2 is a constant "1",the propagation bits used in the carry correction unit should be computed by $p_{n-1}….p_{k+1}p_k$and $g_{n-2…..}g_{k+1}g_kc_k$ .From1 the above analysis, it is shown that the difference between these pre-processing unit (that is,p_i^1=p_i). Besides$p_i^{0p}1_i$, just

When i=0 and k. Consequently

$$\begin{cases} p_0^0 = \overline{p_0}, p_0^1 = p_0 & i = 0 \\ p_k^0 = \overline{p_k}, p_k^1 = p_k & i = k \\ p_i^0 = p_i^1 = p_i & i = 1, \dots, k-1, k+1, \dots, n-1. \end{cases} \tag{30}$$

Hence

$$s_0 = \overline{c_{\text{out}}}p_0^0 + c_{\text{out}}p_0^1 = \overline{c_{\text{out}}p_0} - c_{\text{out}}p_0 = c_{\text{out}} \odot \overline{p_0} \tag{31}$$

$$\begin{aligned} s_k - c_k^{\text{real}} \odot \left(\overline{c_{\text{out}}}p_k^0 + c_{\text{out}}p_k^1\right) - c_k^{\text{real}} \oplus \left(\overline{c_{\text{out}}p_k} + c_{\text{out}}p_k\right) \\ = c_k^{\text{real}} \odot c_{\text{out}} \odot \overline{p_k}. \end{aligned} \tag{32}$$

When $i \quad 1, \dots, k-1, k+1, \dots, n-1$

$$s_i = c_i^{\text{real}} \oplus p_i. \tag{33}$$

At last, the sum bits are

$$s_i = \begin{cases} c_{\text{out}} \oplus \overline{p_0} & i = 0 \\ c_k^{\text{real}} \odot c_{\text{out}} \odot \overline{p_k} & i = k \\ c_i^{\text{real}} \oplus p_i & i = 1, \dots, k-1, k+1, \dots, n-1. \end{cases} \tag{34}$$

In (34), $c_{out}p_k$ and c^{real}_k can be obtained at the same time. Therefore, there is no extra delay compared with other sum computation units.

Design Example : The VLSI implementation structure of modulo 2^8-2^4-1 adder based on the proposed scheme is shown in Fig. 3(a).Fig. 3(b) illustrates the function of each module.

Pre-processing Unit : The pattern “” in Fig. 3 is the pre-processing unit and used to generate carry generation and carry propagation bits for the following prefix computation. Since there are fixed “1” inputs at the 1st and the 4th places, the patterns“ ”and are used for this special situations. The pattern “ ” does not cost any resource in unit-gate model. The computations of these patterns correspond to (10), (11)and (12).

Prefix Computation : The pattern “” is the prefix computation unit. In this example, the Sklansky prefix reeisusedandthereare11prefix computation units, which corresponds to (4). The delay of “” is determined by its’ carry generation path which is one OR gate and one AND gate. However, the pattern “”in the final stage of prefix tree is not needed to compute propagation bits.

The Computation of cout : The is computed by pattern inFig.3. According to(14), $c_{out}=c_{SCSA}+G_{n-1:l}+p_{n-1:l}c^T_l$,$c_{SCSA}+G_{n-1:l}$and $p_{n-1:l}$ c_l^T can be computed concurrently. Then, we can get after an OR gate. Thus, the delay of c_{out} computation will not exceeding the Delay of pattern “” and there is no extra delay. In order to minimize the delay of “”, the value of can be selected so that the delay difference of $G_{n-1:l}$ and c_l^T is at

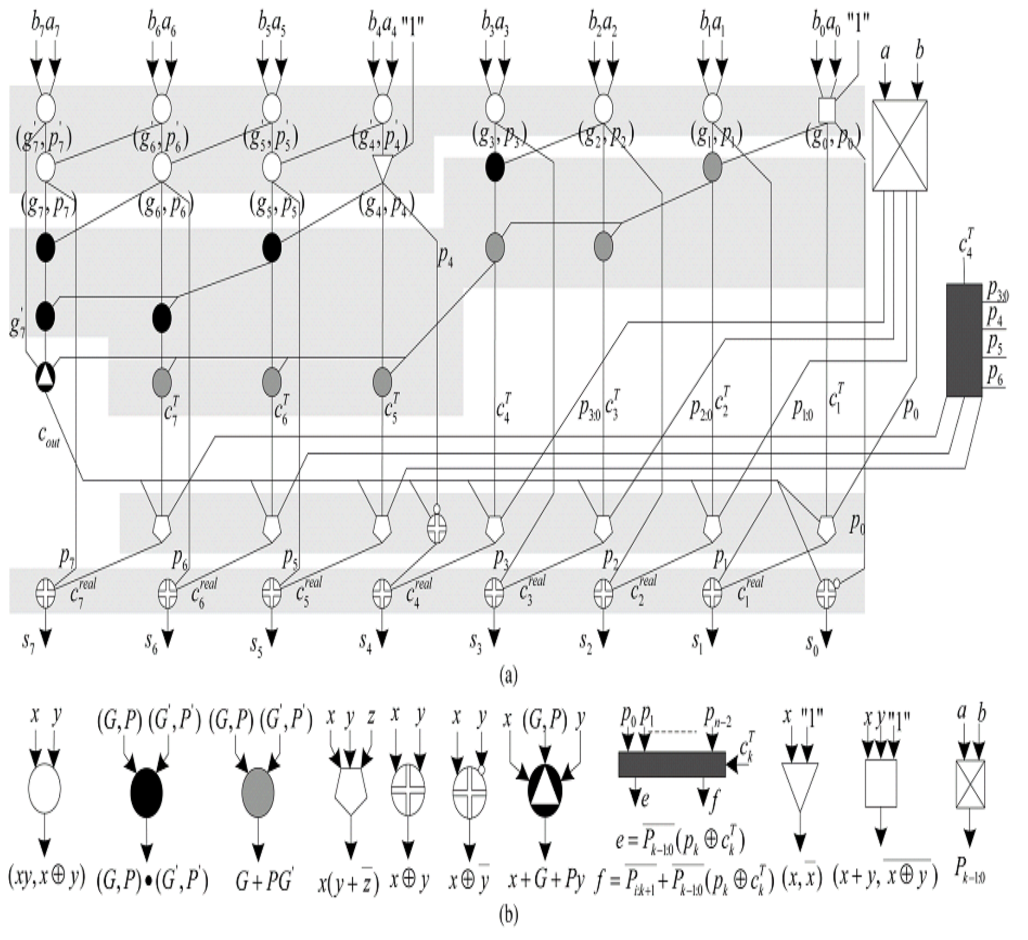


Fig. 3. Modulo $2^n - 2^k - 1$ adder based on the proposed structure.

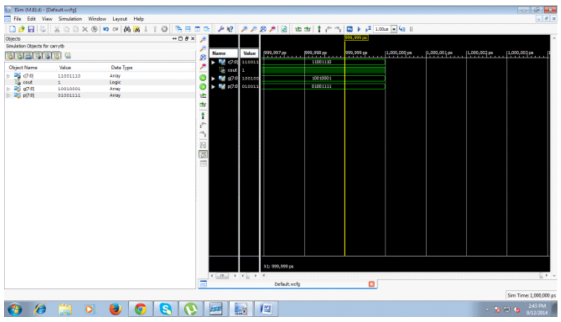
Least one OR gate delay. In (14) C_{SCSA}, is computed in pre-processing firstly. Meanwhile, the delay of $P_{n-1:l}$ is always smaller than that of $G_{n-1:l}$ in prefix tree. Thus, we can compute $C_{SCSA}+P_{n-1:l}c_l^T$ firstly if is obtained before . Otherwise, we can compute firstly. When the last one c_l^T, or $G_{n-1:l}$, arrived, only one OR gate is needed to compute the final value of c_{out}. That is, the delay is T_g if the value of list selected properly. In this example, l=k.

Carry Correction Unit : The pattern in Fig. 3 performs the computation correspond to (27). In this example, 7 correction operators are used. From (27), there are three different situations, that is i=0,1,2….k-1,i=k and i=K+1….n-2 .The $p_{i:0}$ z1, and z2 can be computed by independent modules. The pattern and“ ”in Fig.3isusedtocompute $p_{i:0}$ z1 , and z2, in (27). In this example c^T_k, is computed out c^T_i (i=k+1….,n-1) before with two prefix computation stages. Hence, we can get z1 and z2 without extra delay by using (26). In the worst case, the group propagation bits required in (26) are needed to be computed one by one from p_i (i=0,1,2,….n-2) . However, the extra Components for computing these group propagation bits can be removed when the group propagation bits exist in prefix structure.

Sum Computation Unit : The pattern “⊕” in Fig. 3 is used for performing the sum computation according to (34). As a matter of fact, this operator is the logic XOR operation. The pattern “⊕” in Fig. 3 is a modified XOR operator; one of its inputs is inverted. Because the computation of c_{out} “⊕” in (34) can be performed with carry correction simultaneously, only one XOR operations are required to perform the sum computation and no extra delay is introduced.

II. IMPLEMENTATION AND RESULTS

The proposed designed MODULO 2^n-2^k-1 ADDER using Verilog hardware description language and structural form of coding. The proposed system simulation results are as follows



Object Name	Value	Data Type
c[7:0]	11001110	Array
cout	1	Logic
g[7:0]	10010001	Array
p[7:0]	01001111	Array

III. CONCLUSION

In this paper, a new class of modulo 2^n-2^k-1adder is proposed. The proposed structure is consisted of four units, the pre-processing, the carry computation, the carry correction and the sum computation unit. The performance analysis and comparison show that the proposed algorithm can construct a new class of general modular adder with better performance in delay or "area*delay". It has some main features as following: The way using twice carry corrections improves the performance of area and timing in VLSI implementation and reduces the redundant units for parallel computation of A+B+T and A+B in the traditional modular adders. Any existing prefix tree can be used in this structure. That means fine tradeoff property between area and delay for the proposed scheme. The synthesis results also show that our scheme can be optimized to work at faster operation frequency. Furthermore, the modulus with the form 2^n-2^k-1(1kn-2)of facilitate the construction of a new class of RNS with larger dynamic and more balanced complexity among each residue channel. The work of this paper provides an alternative scheme of modular adder design for this type of RNS. .

REFERENCES

[1] S.Ma,J.H.Hu,L.Zhang,andL.Xiang,"Anefficient RNS paritychecker for moduli set and its applications,"Sci. inChina, Ser. F: Inform. Sci., vol. 51, no. 10, pp. 1563–1571, Oct.2008.

[2] Y. Liu and E. M.-K. Lai, "Design and implementation of an RNS-based2-D DWT processor,"IEEE Trans. Consum, Electron.,vol.50,no.1,pp. 376–385, Feb. 2004.This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.MAet al.: A NOVEL MODULO ADDER FOR RNS 11

[3] P.Patronik,K.Berezowski,S.J.Piestrak,J.Biernat,andA.Shrivastava, "Fast and energy-efficient constant-coefficient FIRfilters usingresidue number system," in Proc. Int. Symp. Low Power Electronicsand Design (ISLPED), 2011, pp. 385–390.

[4] J. C. Bajard, L. S. Didier, and T. Hilaire, " -direct form transposedand residue number systems forfilter implementations," inProc. IEEE54th Int. Midwest Symp. Circuits and Systems (MWSCAS), 2011, pp.1–4.

[5] M. Bayoumi, G. Jullien, and W. Miller, "A VLSI implementation ofresidue adders," IEEE Trans. Circuits Syst., vol. CAS-34, no. 3, pp.284–288, Mar. 1987.

[6] S. J. Piestrak, "Design of residue generators and multioperand modularadders using carry-save adders,"IEEE Trans. Comput,, vol. 43, no. 1,pp. 68–77, Jan. 1994.

[7] H. Vergos, "On the design of efficient modular adders,"J. Circuits,Syst., and Comput., vol. 14, no. 5, pp. 965–972, Oct. 2005.

[8] G. Jaberipur, B. Parhami, and S. Nejati, "On building general modular adders from standard binary arithmetic components," inProc. 45thAsilomar Conf. Signals, Systems, and Computers, 2011, pp. 6–9.

[9] M. Dugdale, "VLSI implementation of residue adders based on binaryadders,"IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process.,vol. 39, no. 5, pp. 325–329, May 1992.

[10] A. A. Hiasat, "High-speed and reduced-area modular adder structuresfor RNS,"IEEE Trans. Comput,, vol. 51, no. 1, pp. 84–89, Jan. 2002.

[11] R. A. Patel, M. Benaissa, N. Powell, and S. Boussakta, "ELMMA: Anew low power high-speed adder for RNS," inProc. IEEE Workshopon Signal Processing Systems, Oct. 2004, pp. 95–100.

[12] R. A. Patel, M. Benaissa, N. Powell, and S. Boussakta, "Novelpower-delay-area-efficient approach to generic modular addition,"IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 54, no. 6, pp.1279–1292, Jun. 2007.

[13] E. Vassalos, D. Bakalis, and H. T. Vergos, "Modulo arithmeticunits with embedded diminished-to-normal conversion," inProc. 14thEuromicro Conf. Digital System Design (DSD), 2011, pp. 468–475.

[14] G. Jaberipur and S. Nejati, "Balanced minimal latency RNS additionfor moduliset ,," inProc. 18th Int. Conf. Systems,Signals and Image Processing (IWSSIP), 2011, pp. 1–7.

[15] H. T. Vergos and C. Efstathiou, "A unifying approach for weighted anddiminished-1 modulo addition,"IEEE Trans. Circuits Syst. II,Exp. Briefs, vol. 55, no. 10, pp. 1041–1045, Oct. 2008.

[16] S. H. Lin and M. H. Sheu, "VLSI design of diminished-one moduloadder using circular carry selection,"IEEE Trans. Circuits SystII, Exp. Briefs, vol. 55, no. 9, pp. 897–901, Sep. 2008.

[17] C. Efstathiou, H. T. Vergos, and D. Nikolos, "Fast parallel-prefixmodulo adders,"IEEE Trans. Comput., vol. 53, no. 9, pp.1211–1216, Sep. 2004.

[18] R. A. Patel and S. Boussakta, "Fast parallel-prefix architectures formodulo addition with a single representation of zero,"IEEETrans. Comput., vol. 56, no. 11, pp. 1484–1492, Nov. 2007.

[19] P. M. Matutino, R. Chaves, and L. Sousa, “Arithmetic units for RNSmoduli and operations,” inProc. 13th Euromicro Conf.Digital System Design: Architecture, Methods and Tools (DSD), 2010,pp. 243–246.

[20] R. A. Patel, M. Benaissa, and S. Boussakta, “Fast moduloaddition: A new class of adder for RNS,”IEEE Trans. Comput.,vol. 56, no. 4, pp. 572–576, Apr. 2007.

[21] L. Li, J. Hu, and Y. Chen, “An universal architecture for designingmodulo multipliers,”IEICE Electron. Expr.,vol.9,no.3, pp. 193–199, Feb. 2012.

[22] R. Zimmermann, “Binary Adder Architectures for Cell-Based VLSIand their Synthesis,” Ph.D. dissertation, Integrated Syst. Lab., SwissFederal Inst. of Technol., Zurich, 1997.

AUTHORS:

P. Rajender received the B.Tech degree in Electronics and Communication Engineering in the year 2012 in Vaageswari College Of Engineering and pursuing M.Tech degree in VLSI Design from Jyothismathi Institute Of Technology & Science , Karimnagar, Telngana. Her area of interests includes VLSI Design, VLSI, AISC and Signal Processing.

Second Author:

R.Srinivas received his **B-tech** in **Electronics and Communication Engineering** from Jyothismathi Institute Of Technology & Science , Karimnagar and the **M.Tech** degree in Balaji Institute Of Technology & Science Narsempet, Warangal, Telangana State, India, . He is now working as Assistant Professor in the department of **Electronics and communication Engineering** Jyothismathi Institute Of Technology & Science , Karimnagar, Telangana, India .His research interests include **VLSI and Embedded Systems design, Networks on Chip and Design for Testability.**