IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834, p- ISSN: 2278-8735.
PP 97-107
www.iosrjournals.org

Performance Comparison of Rns Modular Adder Based On Existing Parallel Prefix Trees And Random Number Generator Design

[1]Anjitha Purushothaman, [2]Divya S.
[1,2]Electronics and Communication Engineering Department,
Sree Narayana Gurukulam College of Engineering, Kadayiruppu, Kerala, India

Abstract: *Residue number system is widely used in application like cryptography, numeral analysis and signal processing. Random number generators are extensively used in cryptography and its design is based on traditional linear feedback shift registers. Residue number system are very suitable for the implementation of fast VLSI system. Modular adder is the basic component in RNS system. Suitable parallel prefix trees are used for the modular adder design. In this paper a performance comparison of modular adder of moduli set* $2^n - 2^k - 1$ *is made and a random number generator based on this adder is proposed to generate random numbers with good randomness properties desirable for cryptographic applications. Moduli set with the form of* $2^n - 2^k - 1$ *(*$1 \leq k \leq n-2$*) is best suitable for multichannel RNS processing. The proposed model offers excellent randomness property which is the basic requirement for network security and also offers better area and delay performance.*

Keywords: *Residue number system, parallel prefix adder,modular adder,carry correction, FPGA*

I. Introduction

The demand for new techniques in network security is increasing with the growth of network services in our world. Cryptography plays an important role in network security and it is a vital tool that provides security against various external and internal threats in the network. Data confidentiality is mainly achieved by means of cryptography. The aim of cryptographic techniques is to secure the information so that only the intended parties can read. For transmitting audio and video signals for cable TV, commercial and sensitive data and video conferencing the speed of the cryptographic module is required to be high. However the traditional software implementation of cryptographic algorithms are not efficient in real time applications.

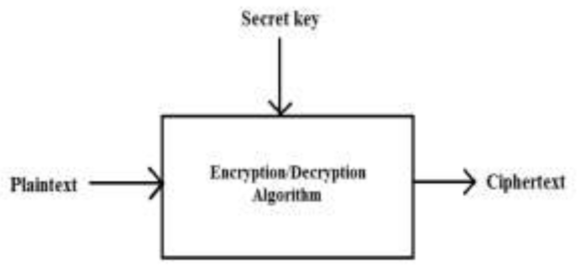

Fig 1 Cryptographic System

The random number generator is a vital cryptographic module widely used for key generation and authentication protocols. The security of such systems completely relies on the excellent randomness property provided by the generators. Thereby future sequence pattern in the random number sequence cannot be predicted by the observed sequence. The random number generators broadly categorized into true random number generator and pseudo random number generators. The design of cryptographically secure random number generator is extremely difficult. Linear feedback shift register is the traditional method for designing and generating random numbers which uses shift registers. This method has good statistical properties and leads to very efficient hardware implementation. Modular adders can be used to design LFSR for generating random numbers that can offer good randomness property.

Modular adders are the most prime component of residue number system. RNS is an ancient numerical representation system. It is a non weighted numerical representation system and have carry free property in addition and multiplication operations. Modular adder is the key module for RNS based DSP systems. Moduli set with the form of $2^n - 2^k - 1$ can offer excellent balance among the RNS channels for multichannel RNS processing. This modular adder can be used to design LFSR based random number generator with good randomness property.

This paper is organized as follows: Section II is a review of the related work used.RNS and modular adder arithmetics are discussed in Section III. An introduction of Parallel prefix trees is presented in Section IV. Section V presents the proposed design and Section VI gives the performance comparison. Conclusion and future work is given in Section VII.

II. Related Work

First a brief survey on modular adders were made and discussed the designing techniques of random number generators based on modular adders. Modular adders can be classified into two types: the general modular adder and the special modular adder and it is based on the form of modulus. In the former adder design the two values A+B and A+B+T should be computed first and one of them is selected as the final output. Bayoumi and Miller [2] proposed a general modular adder for arbitrary modulus by using 2 cascaded binary adders and its delay is the sum of two binary adders. Several modular adders with two binary adders to calculate A+B and A+B+T were proposed subsequently. However this approach offers better delay performance the area consumption is relatively larger and it is twice the binary adder. Reused binary adder configuration [3] is the another type of general modular adder design and was proposed by Dugdale. The drawback of this type of adder is that it will use two operation cycles to perform one modular addition. Subsequently many studies on modular adders were done that have better area and delay performance. A high speed and reduced area modular adder structure for RNS were proposed by Hiasat[6] where any regular carry lookahead based binary adder can be used in the final stage. This structure needs an extra CLA unit to get the carry out bit of A+B+T before the final CLA addition and as a result delay is not reduced significantly. ELMMA [9]algorithm is another popular modular adder design proposed by R.A. Patel. In this adder two carry computation modules for A+B and A+B+T were used and some carry computation units were shared. But in the worst cases almost two independent carry generation modules were used. Dimitris Bakalis [10]proposed fast parallel prefix adder for modulo $2^n + 1$ adders. This architecture is based on parallel prefix carry computation units.

The complexity of special modular adder is much less than that of general modular adder since optimization is possible in special modular adder. The optimization is done according to the modulus. Several architecture for modulo $2^n + 1$ and $2^n - 1$ adder were proposed based on parallel prefix and carry correction.[10][19][20]. Piestrak [4] made a comprehensive study of residue generators and multioperand modular adders. He proposed a design using carry save adder with end around carry and are well suited for VLSI implementation, R.A. Patel [13] first proposed a literature on modulo $2^n - (2^{n-2} + 1)$addition based on carry offset where the carry information of A+B+T is only required to calculate. The carry information for A+B is obtained by modifying the carries of A+B+T. Even if all the redundant modules of carry computation are eliminated, the structure of carry computation is fixed and can only perform the special modular addition of modulo $2^n - (2^{n-2} + 1)$. In most of the RNS based application ,addition and multiplication intensive systems are used and the main issue is the selection of moduli set accordingly. For such systems residue channels are always expected as many as possible where dynamic range is fixed ie. the wordlength of the residue channels can be reduced inorder to achieve better speed performance. Width of the each channel is also expected as close as possible to get similar critical path delay and thereby fine balance is achieved between each residue channels. The modular adders discussed yet are high performance adders but are not always suitable to construct multichannel RNS that offers fine channel balance ie. It is haed to construct a multichannel that have fine balance with moduli set $2^n + 1$ and $2^n - 1$. Recently [1] Shang Ma and Jian Ho proposed a modular adder particularly applicable for RNS systems with modulus of the form $2^n - 2^k - 1$ $(1 \leq k \leq n - 2)$. These adder have outstanding performance in constructing multichannel moduli set with fine balance. L.Li, J Hu and Y Chan recently proposed a general architecture for $2^n - 2^k - 1$ multiplier. However there were only little discussion and recently a detailed discussion was made [1].

Random number generators are the most vital component used in network security systems like cryptography and encryption techniques. Cryptography is mainly concerned with confidentiality where a message is converted from comprehensible form to incomprehensible form rendering it unreadable by interceptors and eavesdroppers. RNG[7] are basically classified as true and pseudo generators. A common method of producing a pseudo random number generators is to use the output of a linear feedback shift register.

Almost all PRNG patterns are reputable and predictable for small cycles. In this paper a new design for random number generator based on modular adder is proposed. This design uses a modular adder $2^n - 2^k - 1$ which has better area and delay performance. This adder is best suitable for RNS multichannel since a class of modulo is designed instead of a single moduli based on different values of k. Moreover when LFSR design based on $2^n - 2^k - 1$ adder and conventional modular adder are compared the proposed gives better area and delay performance. Since randomness is the most important requirement in cryptography it is very necessary to design such generator that have good randomness property. This proposed random number generator based on LFSR offers excellent randomness property.

In the rest of the paper a brief introduction of RNS and modular addition are presented in Section II. And Section III introduces the hardware architecture of modulo $2^n - 2^k - 1$ adder. Section IV describes the proposed random number generator design. Performance of different modular adders and LFSR are evaluated and compared in Section V.

III. Rns And Modular Arithmetics.

RNS arithmetic seems especially suitable for DSP hardware as rapid computation using simple operation of addition subtraction and multiplication can be performed which the basic arithmetic operations of DSP algorithms. RNS arithmetic also has the desirable properties for VLSI implementation of concurrency , modularity and fault tolerant capability.

Residue number system consists of N pairwise relatively prime moduli. A number X is represented as $(|X|_{m1}, |X|_{m2} \ldots\ldots |X|_{mN})$ where $|X|_m \in [0, m-1]$, N>1, GCD $(m_i, m_j) = 1$, i, j = 1,2,….N and GCD is the greatest common divisor of $m_i$and m_j. Let A and B be two integers represented by N-tuple word $\{a_{RNS}^0, a_{RNS}^1 \ldots\ldots.. a_{RNS}^{N-1}\}$ and $\{b_{RNS}^0, b_{RNS}^1, \ldots\ldots. b_{RNS}^{N-1}\}$ respectively in residue number systems. Let ◊ denote the binary operation of addition ,subtraction and multiplication. Then C=A◊B is isomorphic to $C = \{c_{RNS}^0, c_{RNS}^1, \ldots\ldots.. c_{RNS}^{N-1}\}$ where $c_{RNS}^i = |a_{RNS}^i \lozenge b_{RNS}^i|$ and i$\in [0, N-1]$.c_{RNS}^i is solely dependent on a_{RNS}^i and b_{RNS}^i, this results in fast, parallel, independent processing within each of the N residue channels.The modulo m addition for integers A and B in the range of [0,m] is defined as

$$C=\langle A+B\rangle_m = \begin{cases} A+B & A+B<m \\ A+B-m & A+B \geq m \end{cases} \quad (1)$$

If C=$\langle A+B\rangle_m$ and the bit width of the modular adder is n bit where n = $[log_2 m]$ ie n is the smallest integer no less than $log_2 m$. Then eqn (1) can be represented as

$$C = \begin{cases} A+B & A+B+T < 2^n \\ \langle A+B+T\rangle_{2^n} & A+B+T \geq 2^n \end{cases} \quad (2)$$

Where the correction factor $T = 2^n - m$.that is if the carry out bit of A+B+T is '1' then the result of the modular addition is the least significant bits of A+B+T otherwise the result is A+B.

A. Parallel Prefix Addition

The key element in fast addition of two n-bit operands X and Y is in the reduction of the latency in the carry network. Carry computation can be considered as a prefix problem. This method is widely adopted in binary adder design where each sum bit s_i and carry bit $c_i$can be calculated with the previous carries and inputs. Prefix based binary adder can be divided into3 units, the preprocessing unit, prefix computation and sum computation unit.In the preprocessing unit ,prefix computation is calculated as

$$(g_i, p_i) = (a_i b_i, a_i \oplus b_i) \ (3)$$

where $g_i$and p_i represents the i^{th} carry generation and carry propagation bits respectively. The prefix computation unit is used to compute the carry information used in the sum computation unit. For carry computation group generate and group propagate bits are obtained from $g_i$and p_i respectively.

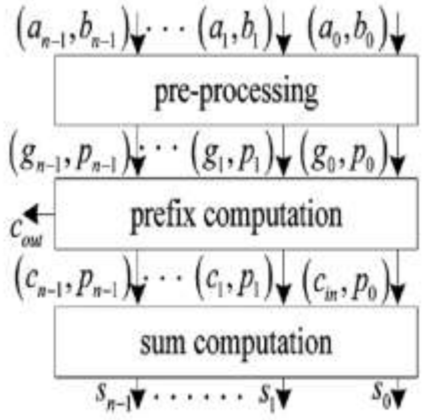

Fig 2. Prefix adder.

$$\begin{cases} (G^0_{i:i}, P^0_{i:i}) = (g_i, p_i) \\ (G^l_{i:k}, P^l_{i:k}) = (G^{l-1}_{i:j+1}, P^{l-1}_{i:j+1}) \bullet (G^{l-1}_{j:k}, P^{l-1}_{j:k}) \\ \qquad = (G^{l-1}_{i:j+1} + P^{l-1}_{i:j+1} G^{l-1}_{j:k}, P^{l-1}_{i:j+1} P^{l-1}_{j:k}) \end{cases} \quad (4)$$

Where i=0,1,….n-1, 0$\leq k \leq j \leq ll = 1,2, \dots . m$ and l represents the l^{th} stage. There are several well known binary prefix addition structures such as Sklansky, Brent Kung, Kogge Stone, Han Carlson. There structures are usually called prefix trees. After prefix computation carries are obtained $c_{i,}$ i= 0,1,2…..n for i^{th} bit and computed as

$$\begin{cases} c_0 = c_{in} \\ c_i = G^l_{i-1:0} + P^l_{i-1:0} c_{in} \quad (5) \\ c_{out} = c_n \end{cases}$$

In the sum computation unit the carries $c_{i,}$ from prefix computation unit and partial sum p_i from the preprocessing unit are used together to compute the final sum s_i.

$s_i = p_i \oplus c_i i = 0,1, \dots . n - 1$ (6)

IV. Parallel Prefix Trees

Parallel prefix structures are widely used in high performance adders as the delay is logarithmically proportional to the adder width. Parallel prefix adders basically consists of three stages and are pre processing, prefix computation and final sum computation. The parallel prefix trees are used in the prefix computation stage where group generate/propagate signals are computed at each bit. Any existing prefix structures can be used to get the carries. In this work different parallel prefix trees areused in the adder of modulo $2^n - 2^k - 1$ and their performance are analysed. Sklansky, Kogge Stone, Brent Kung, Han Carlson are the common prefix trees that are used.

A. Sklansky

Sklansky prefix tree is the simplest among the prefix trees. Fig 3 shows the Sklansky prefix tree for 16 bit operand. It has the least logic levels to compute carries. It uses less cells than Knowles and Kogge Stone prefix structure. But this tree have higher fanout. The fanout of such adders grows exponentially from input to output. This leads to a large delay as tHe adder operands' s width increases. It is the compacted version of Brent Kung where logic level is reduced and fanout is increased.

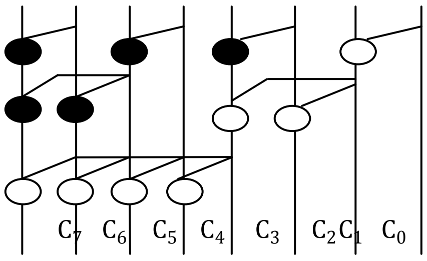

Fig 3 Sklansky Prefix Tree

B Kogge Stone

This tree has a regular layout. It is widely considered in the fastest adder design and is used for high performance adders. Fig 4 shows the Kogge Stone prefix tree for 16 bit operand It takes more area to implement than the Brent Kung , but has a lower fanout at each stage and this increases the performance. Wiring congestion is often a problem for Kogge Stone adder as well

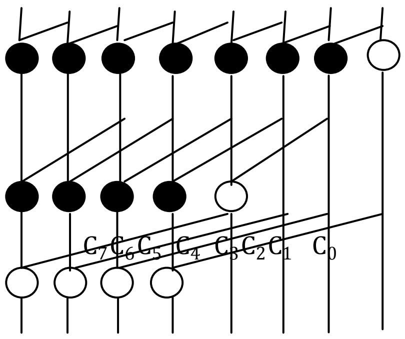

Fig 4 Kogge Stone Prefix Tree

4.3 Brent Kung

Brent Kung trees have large number of levels and so it reduces its operational speed. Fig 5 shows the Brent Kung prefix tree for16 bit operand It have the lowest area and delay with large number of inputs and hence it is power efficient. The number of cells required is less than the Kogge Stone therefore it takes less area to implement . However this adder is not the best for very fast addition. Brent Kung prefix tree is a bit complex to build because it has the most logice levels.

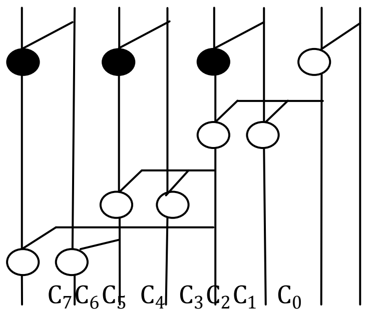

Fig 5 Brent Kung Prefix Tree

4.4 Han Carlson

The idea of Han Carlson prefix tree is very similar to Kogge Stone as it has a maximum fanout of 2. And it uses less cells and wire tracks than the Kogge Stone. Fig 6 shows the Han Carlson prefix tree for 16 bit operand . It requires extra logic level. It is viewed as a sparse version of Kogge Stone prefix tree.

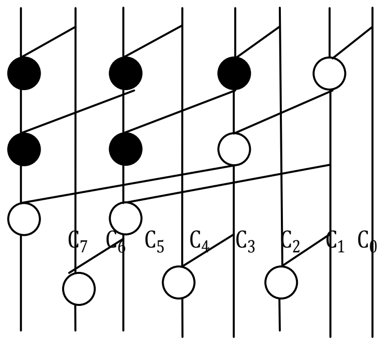

Fig 6 Han Carlson Prefix Tree

V. Novel $2^n - 2^k - 1$ Adder

The adder structure used for the design of random number generator is shown in fig…. it is of modulus $2^n - 2^k - 1$ and composed of four units , preprocessing unit, carry generation, carry correction and sum computation unit. Generally this adder structure can be divided into two general binary adders A1 and A2 as shown in fig 7 with carry correction and sum computation unit. This is based on the characteristics of correction T for modulus $2^n - 2^k - 1$. any existing prefix structures can be used to compute the carries of A+B+T, C_i^T . and by correcting the carries C_i^T we can obtain the final carries C_i^{real} used in the final stage. Thus final modular addition result is obtained from C_i^{real} and partial sum information from the preprocessing units. The main interesting feature of thisarchitecture is that it avoids the calculation of carry information for A+B+Tand A+Bseparately. Thereby area and delay can be reduced significantly and offers flexible tradeoff between area and delay.

A.Pre processing unit

This unit computes carry generation and carry computation bits for every bit I, $i \in [0.n-1]$ .When modulus m=$2^n - 2^k - 1$,then the correction factor is given as $T = 2^{[log_2 2^n - 2^k - 1]} - m. = 2^k + 1$. The binary representation of T is 00….001 00….001. the computation of A+B+T can be performed by A1 and A2 where A1 and A2 are used for lower-k bits and higher n-k bits addition respectively. Let T_{A1}= 00…001,
T_{A2}= 00…001 and the binary representation of A and B are $a_{n-1} \dots . a_{k-1} a_k \dots . a_1 a_0$ and $b_{n-1} \dots . b_{k-1} b_k \dots . b_1 b_0$ respectively.

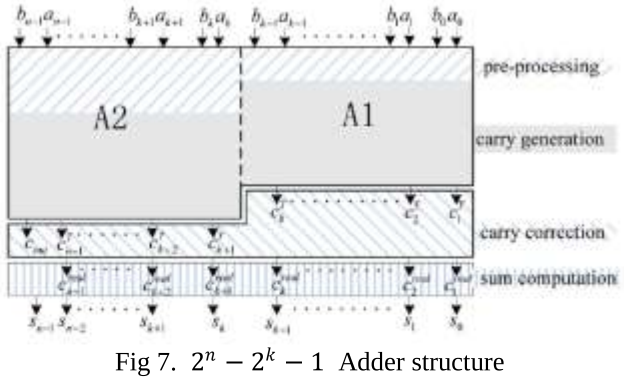

Fig 7. $2^n - 2^k - 1$ Adder structure

The operation of adder A1 and A2 can be given as

$$\begin{cases} S_{A1} = a_{k-1} \dots . a_0 + b_{k-1} \dots . b_0 + T_{A1} \\ S_{A2} = a_{n-1} \dots . a_k + b_{n-1} \dots . b_k + T_{A2} + c_{A1} \end{cases} \quad (7)$$

Where C_{A1} is the carry out bit of adder A1. This LSB bits of T_{A1} is '1' and all others are '0'.This A1 can be treated as a k-bit adder with lowest carry in bit since T_{A1} is one of the input ofA1. Since the LSB bit of T_{A1} is '1' it is considered for the carry generation and carry propagation bits and are computed as

$$\begin{cases} (g_0, p_0) = \left(a_0 + b_0, \overline{a_0 \oplus b_0}\right) & i = 0 \\ (g_i, p_i) = (a_i b_i, a_i \oplus b_i) & i = 1,2, \dots . k-1 \end{cases} \quad (8)$$

The second adder A2 adds the constant T_{A2} and the carry out bit C_{A1} from adder A1. So it can be regarded as a 3-input adder with lowest carry in bit. The 3-inputs are $a_{n-1} \ldots .a_k$, $b_{n-1} \ldots .b_k$ and T_{A2}. In this design the number of inputs is reduced from three to two for adder A2 by using Simple Carry Save adder (SCSA). The first stage of SCSA computes (g_i', p_i') for $i = k, k+1, \ldots n-1$

$(g_i', p_i') = (a_i b_i, a_i \oplus b_i)$ (9)

This $g_i' p_i'$ are treated as the inputs of the second stage in SCSA. The carry generation and carry propagation bits for $i = k, k+1, \ldots n-1$ are obtained from this second stage from (g_i', p_i') and T_{A2}. Thus the final output for preprocessing unit are

$$\begin{cases} (g_k, p_k) = (p_k', g_k') i = k \\ (g_i, p_i) = (p_i' g_{i-1}', p_i' \oplus g_{i-1}') i = k+1, \ldots, n-1 \end{cases} \quad (10)$$

The carry out bit of SCSA, C_{SCSA} is required to compute the carry out bit of A+B+T, C_{out}. It is calculated as

$C_{SCSA} = a_{n-1} b_{n-1} = g_{n-1}'$ (11)

B.Carry Generation Unit

This unit uses any existing prefix structure to compute the carries C_i^T of A+B+T from carry generation and carry propagation bits of pre processing unit. The carry out bit of SCSA is not involved in the prefix computation . C_{SCSA} is combined with the carry out bit of prefix tree and determines the carry out bit of A+B+T, C_{out}.

$$c_{out} = c_{SCSA} + c_n^T = c_{SCSA} + G_{n-1:0}$$
$$= c_{SCSA} + G_{n-1:l} + P_{n-1:l} G_{n-1:l}$$
$$= c_{SCSA} + G_{n-1:l} + P_{n-1:l} c_l^T (12)$$

C. Carry correction unit

The final carries C_i^{real} used in the final sum computation stage for each bit is obtained from the unit. In this design the carries for A+B is C_i^{real} is obtained by correcting the carries C_i^T of A+B+T. Hence area is reduced.The relation between C_i^0 and $C_i^1 (i = 0,1, \ldots n-1)$ is derived where C_i^0 and C_i^1 are the carry outputs of prefix trees when the lowest carry in is '0' and '1' respectively.The relationship is given as

$C_{i+1}^0 = \overline{P_{i:0}} C_{i+1}^1 (i = 0,1, \ldots n-1)$ (13)

Where $P_{i:0} = p_i p_{i-1} \ldots \ldots . p_0$ and $p_i = a_i \oplus b_i$. This means that C_i^0 can be determined from C_i^1 by simple logic operation. This is the foundation of carry correction for this modular adder. The carry bit of A+B can be obtained with twice carry correction of A+B+T. Whether carry correction is performed or not depends on the carry our bit of A+B+T, C_{out}.

Therefore the carry bits required by the modular adder are given as

$$C_i^{real} = \begin{cases} c_{i+1}^T (c_{out} + \overline{P_{i:0}}) & i = 0,1, \ldots, k-1 \\ c_{i+1}^T (c_{out} + \overline{P_{k-1:0}} (p_k \oplus c_k^T)) & i = k \\ c_{i+1}^T (c_{out} + \overline{P_{i:k+1}} + \overline{P_{k-1:0}} (p_k \oplus c_k^T)) & i = k+1, \ldots, n-2 \end{cases} \quad (14)$$

D . Sum computation unit

This unit computes the final result for modular adder. It is same as that in prefix based binary adder. C_i^{real} is used for final computation of sum with respect to C_{out}. If $C_{out} = 0, C_i^{real}$ is the carry bit of A+B, otherwise it is the carry bit of A+B+T. the partial sum bits of A+B and A+B+T are both required in the final sum computation.Let p_i^0 and p_i^1 be the partial sum of A+B and A+B+T respectively

$$\begin{cases} p_0^0 = \overline{p_0}, p_0^1 = p_0 i = 0 \\ p_k^0 = \overline{p_k}, p_k^1 = p_k i = k \\ p_i^0 = p_i^1 = p_i i = 1, \ldots, k-1, k+1, \ldots, n-1 \end{cases} \quad (15)$$

Hence

$$s_0 = \overline{c_{out}} p_0^0 + c_{out} p_0^1 = \overline{c_{out}}\, \overline{p_0} + c_{out} p_0 = c_{out} \oplus \overline{p_0}$$
$$s_k = c_k^{real} \oplus (\overline{c_{out}} p_k^0 + c_{out} p_k^1) = c_k^{real} \oplus (\overline{c_{out}}\, \overline{p_k} + c_{out} p_k)$$
$$= c_k^{real} \oplus c_{out} \oplus \overline{p_k} (16)$$

When $i = 1, \ldots, k-1, k+1, \ldots, n-1$

$$s_i = c_i^{real} \oplus p_i \quad (17)$$

Therefore the sum bits for all i are,

$$s_i = \begin{cases} c_{out} \oplus \overline{p_0} i = 0 \\ c_k^{real} \oplus c_{out} \oplus \overline{p_k} i = k \\ c_i^{real} \oplus p_i i = 1, \ldots, k-1, k+1, \ldots, n-1 \end{cases} \quad (18)$$

This $C_{out} \oplus \overline{p_k}$ and C_i^{real} can be obtained at the same time. Therefore there is no extra delay.

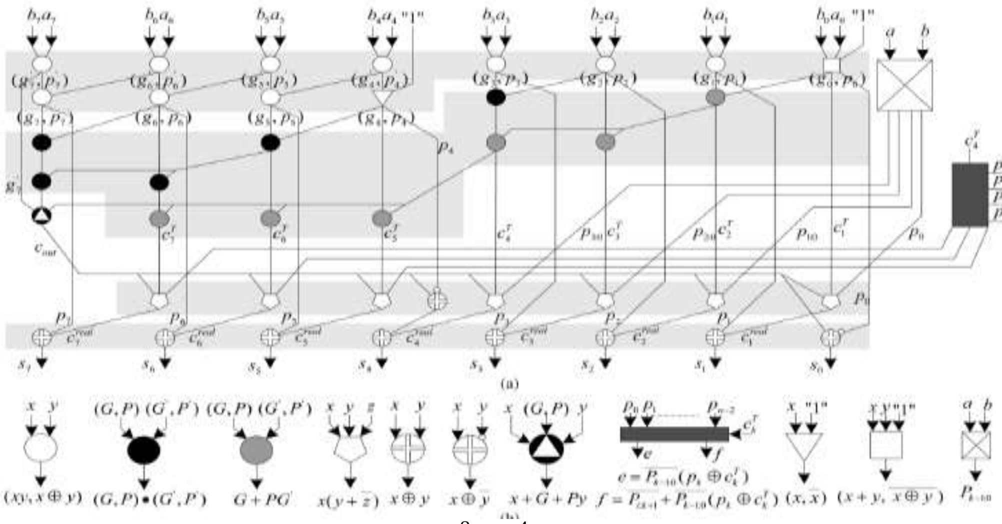

Fig 8 modulo $2^8 - 2^4 - 1$ adder

VI. Proposed System

The data confidentiality in secured communication system is achieved by means of cryptography. This security is maintained by keeping the secret key used for data encryption and decryption confidential. The secret keys should be extremely strong enough so that attackers and eavesdroppers could not predict out and break the cipher text and misuse it.. Therefore we require strong keys. The keys are usually generated by simple random number generators. And the random numbers generated must have excellent randomness properties. Fig 4 shows a conventional random number generator based on linear feedback shift register.

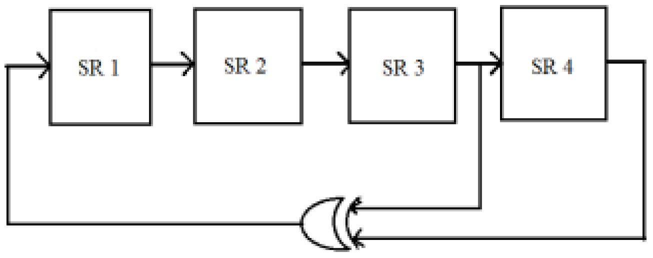

Fig 9. Conventional LFSR

The generator is uses XOR based feedback. The input of shift register is the linear function of previous states. Fig shows the proposed design for random number generator that have excellent randomness properties. In this design the XOR based feedback is replaced by modular adder.

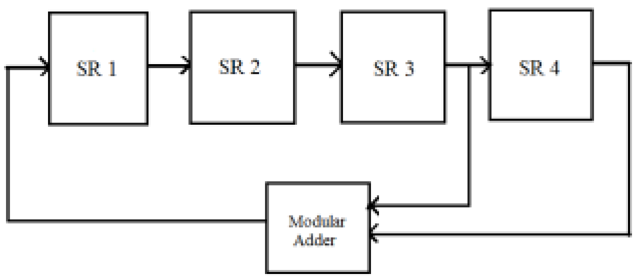

Fig 10 Proposed random number generator.

Whenever the sum exceeds the modulus, the adder produces an exactly different result as sum. By keeping the moduli sets used in the design of modular adders confidential we can produce extremely strong cipher texts rendering it unreadable by interceptors and eavesdroppers. The moduli set $2^n - 2^k - 1$ is very suitable in constructing balanced multichannel with fixed dynamic range and similar critical path delay. So by employing $2^n - 2^k - 1$ modular adder we could get generator with excellent randomness property, large dynamic range and better performance which is very suitable for cryptography application.

6. Fpga Implementation And Performance Comparison

A. FPGA Implementation

To understand the effectiveness of the proposed design, it is implemented on FPGA of device family SPARTAN 3E. this id synthesized in Xilinx 13.3 version. The simulation result for $2^n - 2^k - 1$ modular adder and random number generator are shown.Table 1 shows the device utilization summary and timing report for various adder and the proposed adder. Fig 6 and Fig 7 shows the simulation waveform of modular adder and random number generator.

Table. 1. Logic utilization and Timing report

Modular adders	No of Slices (Available 4656)	No of 4 i/p LUTs (Available 9312)	Delay (ns)
Bayoumi [2]	17	29	14.687
Dugdale [3]	25	41	33.735.
2^(n)-2^(n-2)-1 [13]	12	21	11.831.
2^n-2^k-1	19	33	14.878

Area and delay of the $2^n - 2^k - 1$ adder influence the area and delay of proposed LFSR design. So it is required to reduce the factors of adder so that we can improve the performance of LFSR correspondingly. In [2] where two binary adders are used to get A+B and A+B+T simultaneously. Since two adders are used the area requirement is much greater than other adders. In [3] where single binary adder is used to compute the result but requires twice the clock cyles. Delay is much increased in this scheme. An another scheme of modulo $2^n - 2^{n-2} - 1$ which is a special case of our adder has relatively small delay and give better area delay performance. The adder $2^n - 2^k - 1$ adder offers a difference set of moduli for difference values of ‘k’. That is it is a general design architecture for different modulus. Therefore some optimizations and extra design are applied for such purpose making it suitable for multichannel RNS application. However even if these optimizations are done it still offers better area and delay performance compared to [2][3] [13]. Thus by employing this adder fastest and large dynamic range LFSR can be obtained with excellent randomness property.

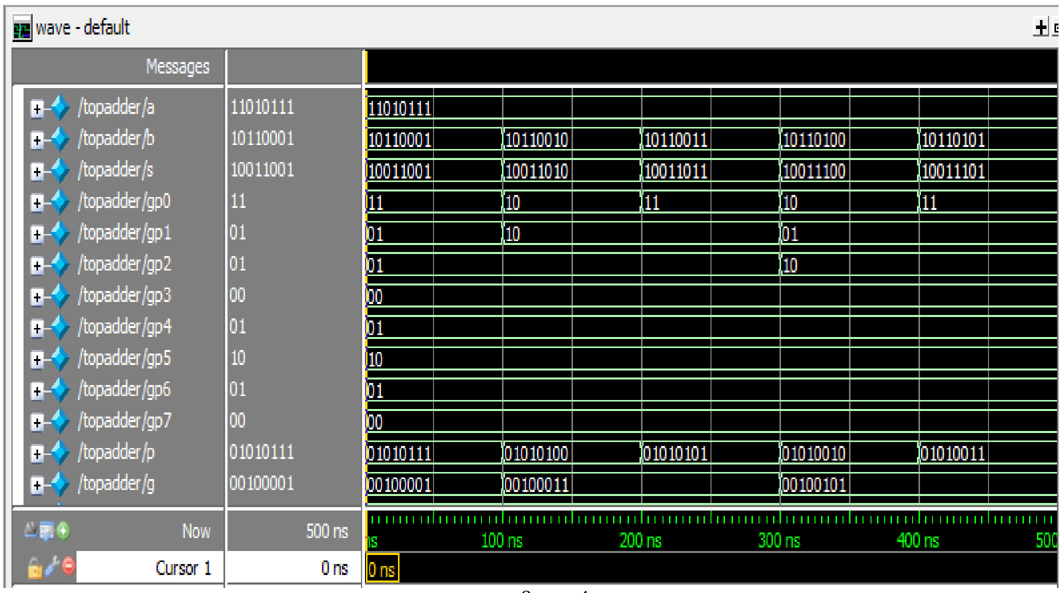

Fig 11 . Simulation of $2^8 - 2^4 - 1$modular adder.

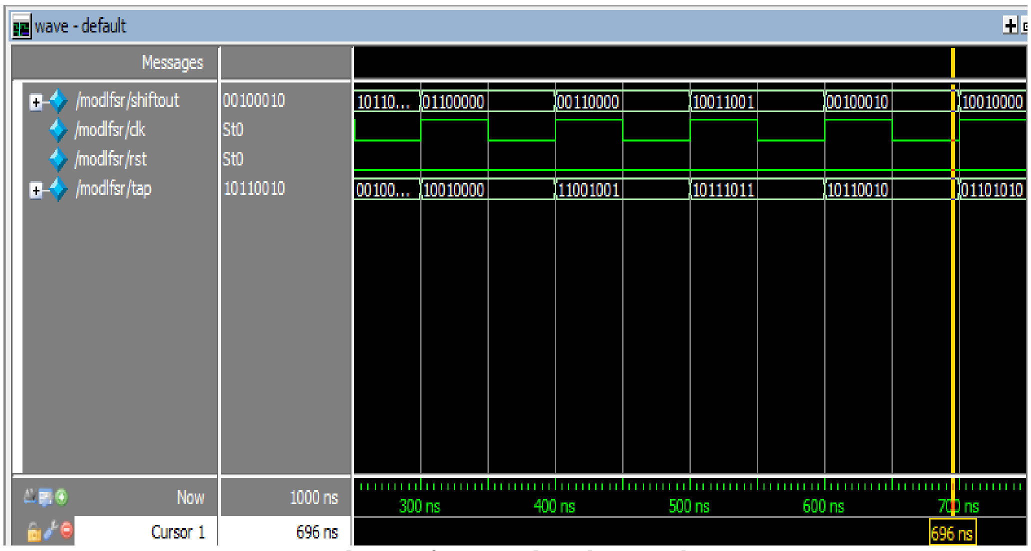

Fig 12 . Simulation of proposed random number generator.

Table II. Logic utilization and Timing report for $2^n - 2^k - 1$ modular adder with different parallel prefix trees.

Parallel Prefix Trees	Sklansky	Kogge Stone	Brent Kung	Han Carlson	Harris
No of Slices	19	19	18	22	22
No of 4 I/P LUTs	33	34	33	38	38
No of Bonded IOBs	24	24	24	24	24
Delay(ns)	14.878	14.637	14.667	14.684	14.684

VII. Conclusion

In this paper a new design approach for random number generator using modular adder is proposed. The proposed design consists of shift registers and modular adders. And modular adder is constructed of four units, preprocessing, carry computation, carry correction and sum computation unit. An analysis of performance of modular adders with different parallel prefix trees are also made. On comparison the Sklansky prefix tree offers the best area and delay performance with least number of logic levels . Since the modular adder use twice carry corrections instead of carry computation improved the area and timing in VLSI implementation and reduces the redundant units of parallel computation of A+B+T and A+B in the traditional adder. Hence comparison shows the LFSR designed using $2^n - 2^k - 1$ offer better area and delay performance when compared with traditional adders. The modulus with the form of $2^n - 2^k - 1$ facilitates the construction of RNS channels with large dynamic and more balanced complexity among each residue channels.

References

[1] Shang Ma, Jian Hao Hu and Chen Hao, "A novel modulo $2^n - 2^k - 1$ adder for residue number system" IEEE Transactions On Circuits And Systems—I: Regular Papers. vol. 60, no. 11, pp. 2962–2972, May. 2013

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

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

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

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

[6] Cerda J.C., Martinez C.C., Corner J.M. and Hoe, "An Efficient FPGA Random Number Generator Using LFSR and Cellular Automata", IEEE Trans. Circuits & Systems,pp.912-915, Aug 2012.

[7] Erkek E and Tuncer T., "The implementation of ASG and SG Random Number Generator", IEEE International Conference on Science and Engineering,pp 363-367,July 2013.

[8] Liang W. and Long Jing, "A Cryptographic Algorithm Based on Linear Feedback Shift Register", IEEE International Conf on Computer Applications and Systems Modeing, vol. 15, pp 526-529, Oct 2010.

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

[10] E. Vassalos, D. Bakalis, and H. T. Vergos, "Modulo $2^n + 1$ arithmetic units with embedded diminished-to-normal conversion," in Proc. 14th Euromicro Conf. Digital System Design (DSD), 2011, pp. 468–475

[11] R. A. Patel and S. Boussakta, "Fast parallel-prefix architectures for modulo $2^n - 1$ addition with a single representation of zero," IEEE Trans. Comput., vol. 56, no. 11, pp. 1484–1492, Nov. 2007.

[12] S. H. Lin and M. H. Sheu, "VLSI design of diminished-one modulo $2^n + 1$ adder using circular carry selection," IEEE Trans. Circuits Syst.II, Exp. Briefs, vol. 55, no. 9, pp. 897–901, Sep. 2008.

[13] R. A. Patel, M. Benaissa, and S. Boussakta, "Fast modulo $2^n - (2^{n-2} + 1)$ addition: A new class of adder for RNS," IEEE Trans. Comput., vol. 56, no. 4, pp. 572–576, Apr. 2007.

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

[15] P. M. Matutino, R. Chaves, and L. Sousa, "Arithmetic units for RNS moduli $2^n - 3$ and $2^n + 3$ operations," in Proc. 13th Euromicro Conf. Digital System Design: Architecture, Methods and Tools (DSD), 2010, pp. 243–246.

[16] E. Vassalos, D. Bakalis, and H. T. Vergos, "Modulo arithmetic units with embedded diminished-to-normal conversion," in Proc. 14th Euromicro Conf. Digital System Design (DSD), 2011, pp. 468–475.

[17] P. Patronik, K. Berezowski, S. J. Piestrak, J. Biernat, and A. Shrivastava, "Fast and energy-efficient constant-coefficient FIR filters using residue number system," in Proc. Int. Symp. Low Power Electronics
and Design (ISLPED), 2011, pp. 385–390.

[18] S. Ma, J. H. Hu, L. Zhang, and L. Xiang, "An efficient RNS parity checker for moduli set $\{2^n - 1, 2^n + 1, 2^{2n} + 1\}$ and its applications,"Sci. in China, Ser. F: Inform. Sci., vol. 51, no. 10, pp. 1563–1571, Oct. 2008.

[19] G. Jaberipur and S. Nejati, "Balanced minimal latency RNS addition for moduli set$\{2^n - 1, 2^n, 2^{2n} + 1\}$,," in Proc. 18th Int. Conf. Systems, Signals and Image Processing (IWSSIP), 2011, pp. 1–7.

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

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

[22] R. Zimmermann, "Binary Adder Architectures for Cell-Based VLSI and their Synthesis," Ph.D. dissertation, Integrated Syst. Lab., Swiss Federal Inst. of Technol., Zurich, 1997.