2017 30th International Conference on VLSI Design and 2017 16th International Conference on Embedded Systems

A 64b/66b Line Encoding for High Speed Serializers

Satyajit Mohapatra[1], Hari Shanker Gupta[2], Jatindeep Singh[1], Nihar Ranjan Mohapatra[1]
1, Department of Electrical Engg. Indian Institute of Technology, Gandhinagar-382355, India
2, Space Applications Centre, Jodhpur Tekra, Ahmedabad-380015, India
nihar@iitgn.ac.in , hari@sac.isro.gov.in

Abstract—**With advancement in technology, there is an increase in the size of current files, thus leading to a demand in increase in data transfer rate. Line encoding can potentially help in increasing speeds by dramatically reducing overheads without necessitating improvements in SERDES devices. Conventionally this technique is suited for optical fiber communication which involves very low BER, but can also be extended for higher BER by including proper error correction codes. This paper discusses a modification in the physical layer hence increasing the transfer rate without affecting the current communication devices. We have implemented line encoding technique for the design of high speed SERDES devices. The proposed 64b/66b line encoding technique reduces additional overhead by 16% with respect to conventional 8b/10b. Hence this technique is advantageous for high speed communication. However, it may lead to high BER. We have optimally selected polynomial for scrambling and descrambling process. The proposed polynomial results in improved efficiency of the scrambler. Algorithms have been developed to find primitive polynomial of three non-zero elements. Various polynomials have been generated and tested in MATLAB. The generated polynomials have been implemented on VHDL and hardware verification has been successfully completed and demonstrated on Actel ProASIC 3C.**

Keywords— ***Serializer-Deserializer (SERDES), Line Encoding Technique, Bit Error Rate (BER), Camera Electronics, Inter Symbol Interference (ISI), Scrambling, Descrambling, Cyclic Redundancy Check (CRC), Parallel Data Transmission, High Speed Serializers, Linear Feedback Shift Registers (LFSR).***

I.INTRODUCTION

Data transmission between sub-systems is a major challenge for miniaturization of electronics used for space applications. In modern satellite systems and applications, large amount of high speed data is required to be transmitted from one system to another. Conventional parallel data transmission requires a large number of cables/interface-packages and therefore results in large weight and volume. Parallel interface in camera systems requires >8000 cables between camera electronics and data handling system. In addition, with increase in transmission rate, problems associated with EMI/EMC, clock skew and crosstalk become more critical. A possible solution identified is SERDES interface which performs parallel-to-serial conversion at the transmitting end, transmit the serial stream preferably over differential medium and finally converts the serial data back to parallel at the receiving end. This considerably reduces the number of interconnecting signals and overcomes the issues of crosstalk. A typical SERDES comprises of an encoder/decoder circuitry, PLL, timing-control circuit and multiplexer/de-multiplexer. SERDES architectures suitable for space applications [1,2] include parallel clock/strobe SERDES and SERDES with encoder/decoder. Encoding of serial data enables high speed serial data transmission by incorporating clock embedding, pre-emphasis, DC balancing, sync insertion, error detection and inter-symbol interference. Currently available SERDES interface devices suffer from the limitations of speed, poor reduction factor, no clock embedding or non-availability of space qualified part. Off the shelf SERDES devices [3] available from few vendors suffer from limitations like poor reduction factor, no clock embedding and operating frequency. Most of them operate in excess of >1Gbps with their patented design. Information of internal functional blocks is not available, which is required from reliability point of view. In addition to costs being on the higher side, the non-availability of space qualified parts might hamper the mission timelines.

Line encoding can potentially help in improving speed by reducing overheads without necessitating improvements in SERDES devices [4]. Asynchronous mode data transmission is most suited due to non-requirement of high speed clock transmission along with data. However it adds a requirement of clock and data recovery circuit at receiver which further adds to the requirements of an encoding technique [5-8]. Serial data transmission essentially requires an encoding techniques at transmitter which provides more efficient and optimized serial transmission. An 8b/10b encoding based SERDES interface typically achieves speed of 250Mbps, which corresponds to a transmission of 8-bit at 25MSPS and reduces interfaces by a factor of 8. Higher factors can potentially be achieved with encoding techniques like 12b/14b and 64b/66b. 64b/66b line encoding technique has advantages [9,10] over 8b/10b encoding [11-16] and provides a much smaller overhead. With large overhead, it is difficult to achieve high speed using SERDES technology. But with the help of 64b/66b line encoding technique we can achieve speed up to 10 GbPS. The paper is organized into five sections. Various design requirements, architectures and analysis of the 64b/66b technique are discussed in section II. The implementation details of proposed encoding technique are provided in section III. Hardware verification of the design and achieved results are summarized in section IV followed by concluding remarks.

II. SYSTEM ANALYSIS AND DESIGN

A typical high resolution electro-optical camera for remote sensing applications requires processing of multiple video ports. Each port is processed to yield typically ≥7 bit digitized video data, which needs to be transmitted to other subsystems for further processing/transmission. This is conventionally implemented by parallel data transmission i.e. using multiple cables/interfaces-packages. Increase in video ports results in increased weight, volume and system power requirements.

2380-6923/16 $31.00 © 2016 IEEE
DOI 10.1109/VLSID.2017.47

A comparative overview of detector data parallel interface (the digitized CCD video data between camera electronics and the data handling system) requirements for earlier and future space missions is provided in Table-I shown below.

TABLE-I: OVERVIEW OF CAMERA INTERFACE REQUIREMENTS

Missions/Parameters	Previous	Future
Sampling Rate (MSPS)	4.2	10
Video Ports	16	384
Interfaces	320	8448
Interface Power (W)	1.5	90.3
Harness Weight (excl. connector) (Kg)	3.6	94.1

With large overhead, it is difficult to achieve high speed using SERDES technology. Line encoding can potentially help in improving speed by reducing overheads without necessitating improvements in SERDES devices. As the overhead requirement of 8b/10b technique is 25%, it is challenging to achieve the speed of 10GbPS with currently available SERDES devices. Available SERDES that could run at just over 10GbPS, could not be pushed to the 12.5GbPS limit required to support 8b/10b overhead. Hence the need for a new encoding technique with less overhead was looked into.

A. 64b/66b System Architecture

The implemented system architecture with 64b/66b encoding scheme is shown in Fig.1. It consists of the encoder/decoder, scrambler/descrambler modules, CRC encoding module and the transmitter/receiver sections. The various modules are coded in Matlab as well as VHDL and verified on the FPGA.

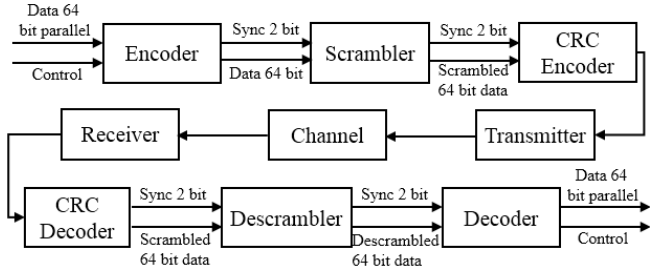

Figure 1. Implemented system architecture with 64b/66b encoding scheme

The proposed implementation of 64b/66b encoding is very different from 8b/10b encoding due to its less coding overhead. Rather than using a 8b/10b type lookup table, 64b/66b uses a scrambling method combined with a non-scrambled sync pattern and control type. The scrambler is the backbone of 64b/66b encoding but it is not able to provide word alignment. Word alignment is provided by addition of sync bit at the starting of the word. Scrambling eliminates long strings of 1's and 0's and works to eliminate other patterns that may have a negative impact on the receiver's signal decoding ability. Selection of polynomial involved in scrambling/descrambling process plays a critical role in determining efficiency of the scrambler. Extended Galois polynomial is therefore the most suitable for implementing the 64b/66b encoding technique. Strategic extension of Galois field for scrambling polynomial implemented in 64b/66b encoder is further described in following sections B, C and D.

B. The Galois Field Operations

All operators in this field are defined uniquely. Addition and subtraction is defined as XORing of elements and multiplication as ANDing of elements [17]. Table-II describes Galois field operations such as add / sub and product / division.

TABLE-II: OVERVIEW OF THE BASIC GALOIS FIELD OPERATIONS

Elements		Add/Sub	Product	Division
0	0	0	0	Not Defined
0	1	1	0	0
1	0	1	0	Not Defined
1	1	0	1	1

C. Linear Feedback Shift Registers (LFSR)

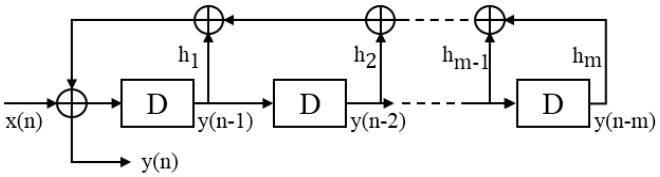

Figure 2. General representation of linear feedback shift registers (LFSR)

From Fig. 2 above, the output can be expressed as following

$$y(n) = x(n) + \sum_{k=1}^{m} h_k y(n-k) \quad (1)$$

For x(n)=0, it results in PN Sequence Generator[18,19], which otherwise results in Multiplicative Scrambler. Eqn.(1) can further be simplified using Huffman Transform given as,

$$X(D) = \sum_{n=0}^{\infty} x(n)D^n \quad (2)$$

An important property of the Huffman Transform is given by

$$H\big(y(n-k)\big) = H(y(n))D^n \quad (3)$$

By applying Equation(2) and (3) to Equation(1) and solving,

$$X(D) = Y(D)\big[1 - \sum_{k=1}^{m} h_k D^k\big] \quad (4)$$

Hence the connecting polynomial is expressed as following,

$$h(D) = 1 - \sum_{k=1}^{m} h_k D^k \quad (5)$$

D. Polynomial in Galois Field

General representation of polynomial remains the same as in the Euclid Space but the difference lies in the definition of addition and multiplication operations.

$$f(x) = h_m x^m + h_{m-1}x^{m-1} \cdots\cdots h_1 x^1 + h_0 \quad (6)$$

The polynomials may be of the reducible or irreducible form. Reducible polynomial are those that can be reduced into multiple polynomials of smaller order. An example of reducible polynomial $h(D) = D^3 + I = (D + I)(D^2 + D + I)$, Since h(1) = 0,As D+1 should be root of the given polynomial

$$h(D) = D(D^2 + D + I) + (D^2 + D + I) \quad (7)$$

$$= D^3 + D^2 + D + D^2 + D + I = D^3 + I \quad (8)$$

Irreducible polynomial are those polynomials that cannot be reduced into multiples of polynomials of smaller order. An example of irreducible polynomial is $h(D) = D^3 + D + I$, Where, h(0)=1;h(1)=1. As connecting polynomial define the structure of LFSR, hence by use of a reducible polynomial, a larger LFSR could be replaced by a smaller order LFSR. Therefore in this work, an irreducible polynomial is used for implementing the LFSR. An irreducible polynomial of order m is said to be primitive polynomial if it divides $x^L + 1$ perfectly where L is the smallest integer equal to 2^m-1[20,21]

E. Extension of Galois Field into LFSR

Let us consider an irreducible polynomial,$f(x) = 1 + x + x^3$. As the roots of the polynomial are not present in GF(2), they will have a root in GF(2^m). If A be the root, $I + A + A^3 = 0$. Properties of the primitive polynomial are given in Table-III.

TABLE-III: PROPERTIES OF PRIMITIVE POLYNOMIAL EXAMPLE

	y(n)	y(n-1)	y(n-2)
A=A	0	1	0
A^2=A^2	0	0	1
A^3=A+1	1	0	0
A^4=A^2+A	1	1	0
A^5=A+A^2+1=A^3+A^2	1	1	1
A^6=A^2+A^3+A=A^2+1	0	1	1
A^7=A^3+A=1	1	0	1
A^8=A	0	1	0

F. Scrambler Module Implementation

The scrambler is the backbone of 64b/66b encoding but it is not able to provide word alignment. Word alignment is provided by addition of sync bit at the starting of the word. Scrambling eliminates long strings of 1's and 0's and works to eliminate other patterns that may have a negative impact on the receiver's ability to decode the signal. In order to test the performance of the new 64b/66b encoding technique and scrambling capacity using a primitive polynomial, the design specifications for the encoding block was kept to be simple. At the input of encoder, there is a single bit specifying whether the data is a pure data or a control [22]. Further the use of sync bit (Fig.3)reduces the misreading of 64-bit data as pure data or control as if in case of error '1' can change to '0'.

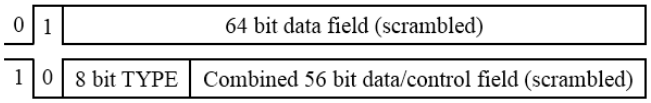

Figure 3. Addition of sync bit for the word alignment in 64b/66b encoding.

Scrambling randomizes long strings of '1' or '0' as only limited run length can be tolerated by the system for error free transmission [23]. It also encrypts the data as the initial value placed in the scrambler is required for descrambling of data as well. Scramblers can either be additive (synchronous) or multiplicative (self-synchronizing). Additive scrambler [24] uses Linear Feedback Shift Registers (LFSR) to generate pseudo random sequence and continuously XOR with the received data while multiplicative scramblers use the feedback of whatever data is being scrambled thus making the scrambled data more random. Here also we use LFRS with feedback as shown aside. Fig.4 (a) and (b) shows the implementation of additive and multiplicative scrambler respectively. The pseudo random pattern generated by additive scrambler is repetitive hence for the same inputs only one type of scrambled output can be generated, but multiplicative scrambler can generate different output for the same input depending on the initial seed provided [25]. Even if in the case that the frame sync is lost, multiplicative scrambler need not be reset, but additive scramblers must be reset by the frame sync. If this fails then a massive error propagation will result as a complete frame cannot be descrambled. Hence to address this issue, multiplicative scrambler has been used in implementing 64b/66b encoding.

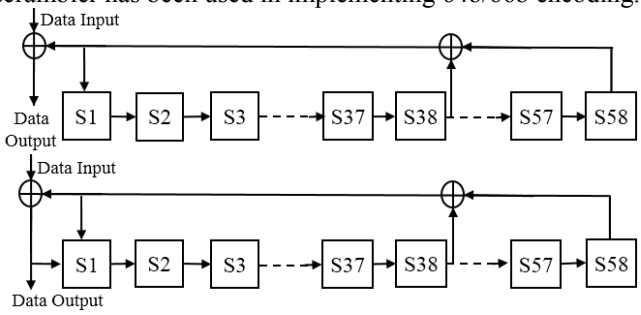

Figure 4. Serial Additive Scrambler (up), Multiplicative Scrambler (below)

The descrambler works on a similar principle as the scrambler and hence serves as a complementary circuit. There is a need to initialize the descrambler and then all scrambled data can be converted back to the same form. There are two blocks namely, the XOR tree and the LFSR block. First the 64-bit data entering is stored onto the LFSR and serves as the initial value of registers in LFSR for the next 64-bit of data. In the XOR tree we apply protocol logic needed for unscrambling the data. We also use the values given by LFSR block, as these are the initial values of registers used for descrambling. Fig.5 shows the implementation of the multiplicative descrambler.

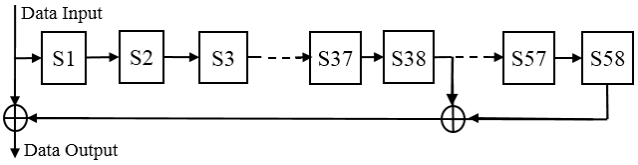

Figure 5. Implementation of the serial multiplicative descrambler design.

For high speed applications serial form of scrambler cannot be implemented as for scrambling of every single bit, one clock is required, thus to scramble a large number of bits, large time delay in necessary. Hence we have implemented parallel scrambling while using the protocol logic. The new 64-bit word in every clock are applied to a 64-bit parallel scrambler. The figure-6 below shows the algorithmic implementation of the scrambler-descrambler module. The process for developing a 32-bit parallel scrambler is described below. Let $x^1(0), x^2(0)$ represent the logic in register S-1 and S-2 respectively as shown in Fig. 5 and a zero represents the case that no any bit has been passed. The initialization sequence of the registers is done as per following algorithms.

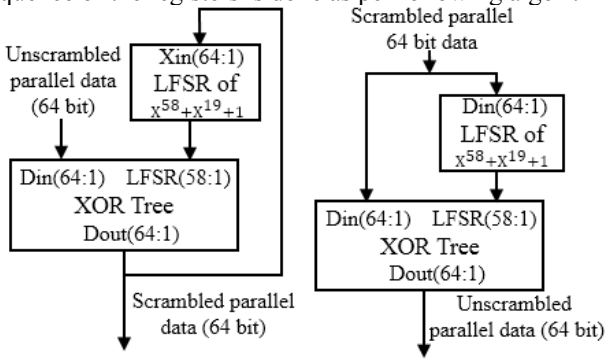

Figure 6. 64-bit parallel scrambler (left) and de-scrambler (right) algorithm

$x^{58}(0) = x^{58}$

$x^{57}(0) = x^{57}$

$\vdots$

$x^{2}(0) = x^{2}$

$x^{1}(0) = x^{1}$

After passing of one bit,

$x^{58}(1) = x^{57}$

$x^{57}(1) = x^{56}$

$\vdots$

$x^{2}(1) = x^{1}$

$x^{1}(1) = x^{58} + x^{38} + 1$

Scrambled 32-bit of data is given by logic,

$d1scram = x^{58} + x^{38} + 1$

$d2scram = x^{57} + x^{37} + 1$

$\vdots$

$d32scram = x^{27} + x^{7} + 1$

Resetting of the registers is given below as,

$x^{32}(32) = d1scram$

$x^{31}(32) = d2scram$

$\vdots$

$x^{2}(32) = d31scram$

$x^{1}(32) = d32scram$

Resetting of the left registers is given below,

$x^{58}(32) = x^{26}$

$x^{57}(32) = x^{25}$

$\vdots$

$x^{34}(32) = x^{2}$

$x^{33}(32) = x^{1}$

G. Cyclic Redundancy Check (CRC) Implementation

In general, as single bit error can propagate to triple bit error after passing through descrambler. Hence it is necessary to detect errors and possibly correct them [26]. Here, we have implemented CRC polynomial of 8th order, in order to keep the overhead low to 11% while sufficiently adding error handling capability. The CRC polynomial simulations have been carried out on MATLAB and finally implemented on FPGA. CRC decoder before descrambler is implemented to minimize errors. The principle of the CRC error correction is as follows. Let M(x) be the message polynomial and P(x) be the generator polynomial. For a given CRC scheme and P(x) is a fixed value and known both by sender and receiver. The transmitted polynomial F(x) is setup such that F(x) becomes divisible with P(x).The CRC decoder checks whether the received polynomial R(x) is divisible by P(x). i.e. $F(x)/P(x) = Q(x) + Z/P(x)$. If R(x) contains some error E(x), it would not be divisible with P(x) hence error is detected. For error correction we could look at the remainder or syndrome and identify the least number of bits that could have error and change them. A case also arises where E(x) might be divisible with P(x) and error might not be detected. Hence only those polynomials are chosen that could minimize the probability of such cases. Fig. 7 shows the implementation of serial CRC.

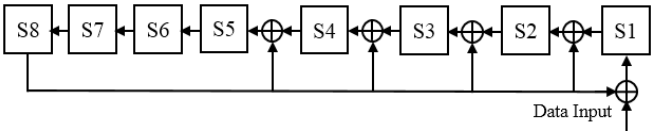

Figure 7. Block implementation of serial cyclic redundancy check (CRC)

As shown above the divider circuit takes serial input, so cannot be implemented for high speed applications. Hence we go for the parallel implementation along with protocol logic. The logic is based on the fact that the highest bit would be XORed and used to form smaller bits. We have implemented using the CRC-8 polynomial $x^8 + x^4 + x^3 + x^2 + 1$. In transmitting end, we multiply message code M(x) by x^n and divide $x^nM(x)$ by P(x), ignoring the quotient while keeping the reminder C(x) to form and send $F(x)= x^n M(x) +C(x)$. At receiving end, received polynomial R(x) is divided by P(x) and is accepted if remainder is '0' else otherwise rejected. The design of the parallel CRC encoder-decoder is shown in Fig.8.

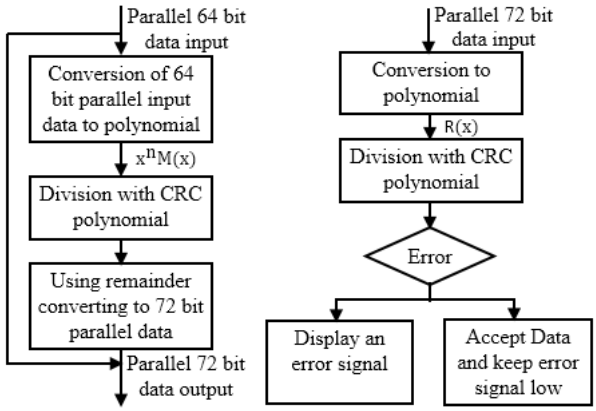

Figure 8. CRC Encoder (Left) and Decoder (Right) algorithm flow chart

III. SOFTWARE & HARDWARE IMPLEMENTATION

The implementation and testing of the circuit is done on Pro ASIC 3EA3PE1500PQG208. The test bench was setup for testing competing experimental cases. A top down approach is followed for the design implementation and verification. Equations are modelled into polynomials and implemented in MATLAB including both serial and parallel implementation of the scrambler and descrambler modules. Simulink is used for verifying the accuracy of the achieved results. The entire circuit was coded in VHDL at module and system levels, after thorough and complete system verification. The VHDL modules are finally burned onto the ACTEL Processor [28]. The following cases were implemented and results analyzed.

Case-1: The output of the circuit was tested for four different input conditions. The input conditions were switched via a reset button. If the input and output were matched, a pass led was activated, which otherwise was kept low. In all four input conditions the pass led glow confirmed that input data matched output data and received error free as illustrated in Table-IV.

Case-2: Here four different inputs conditions were provided as earlier, but a single condition was intentionally setup to give an error. The input conditions were switched via a reset button. If the input and output were matched, a pass led was activated, which otherwise was kept low. It was observed that after four resets the glowing led would switch off hence indicates the design is working perfectly (Table-IV). The hardware verification process of circuit is illustrated in Fig.9 and Fig.10.

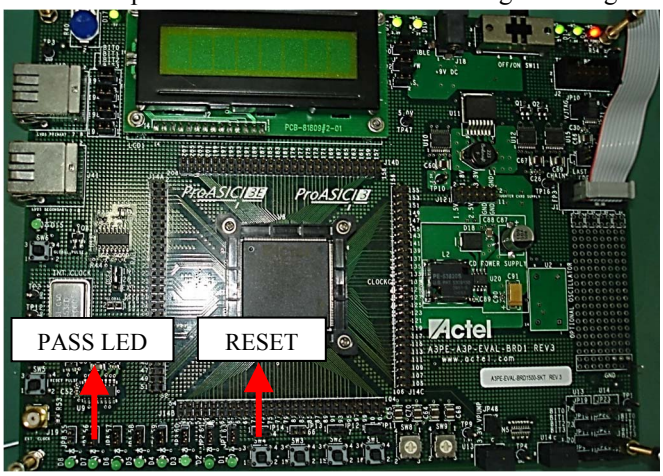

Figure 9. FPGA implement of 64b/66b encoding technique with error word

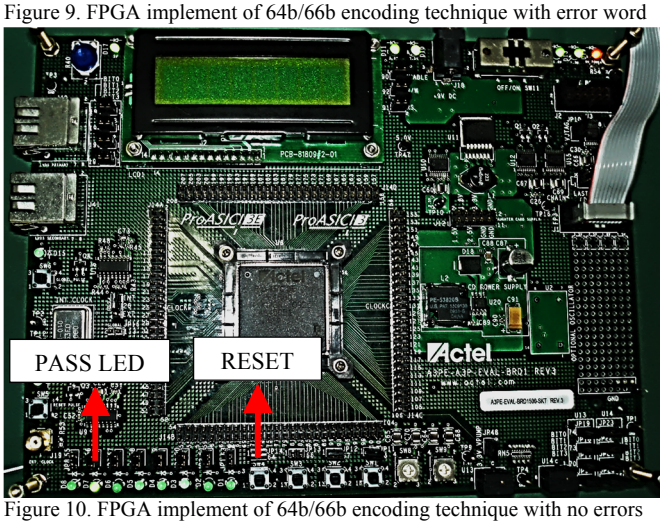

Figure 10. FPGA implement of 64b/66b encoding technique with no errors

TABLE-IV: Hardware Verification Process Involved in Testing Cases I & II

[Case-1]: Input Pattern	CRC	Output Pattern (HEX)	CRC:O/P	LED	RST
E100CAFEA5421642	86	E100CAFEA5421642	00(HEX)	HIGH	0
A1008AFEA5428643	E8	A1008AFEA5428643	00(HEX)	HIGH	1
D110AAFE55422643	E3	D110AAFE55422643	00(HEX)	HIGH	2
E139EC73A5420641	66	E139EC73A5420641	00(HEX)	HIGH	3
[Case-2]: Input Pattern	CRC	Output Pattern (HEX)	CRC:O/P	LED	RST
F10BCAFEAA421932	86	F10BCAFEAA421932	00(HEX)	HIGH	0
AB118A4EA522164A	E8	AB118A4EA522164A	00(HEX)	HIGH	1
D11022F85517664A	E3	D11022F85517664A	00(HEX)	HIGH	2
E139EC73A5420641	FF	FFFFFFFFFFFFFFFF	1E(HEX)	LOW	3

IV. RESULTS AND DISCUSSION

A. Behavioural Test & Hardware Implementation Result

The implementation and testing of circuit was done on Pro ASIC 3EA3PE1500PQG208. The behavioral simulation of the various test conditions along with the cyclic redundancy check algorithms in encoder and decoder is shown in Fig. 11.Further behavioural tests were carried in FPGA. Table-IV gives an overview of the verification process for cases-I & II.

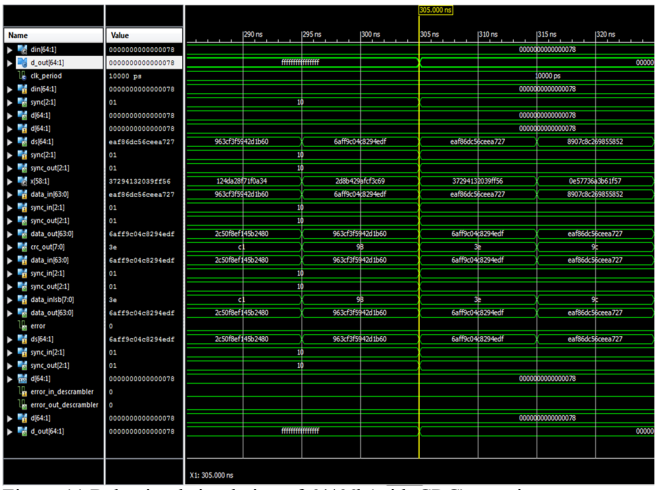

Figure 11.Behavioral simulation of 64/66b(with CRC)to various test vectors

B. Runlength and Effeciency Computations

To decrease the run length of output we propose the use of additional scrambler serially with existing one. The proposed polynomial being F(x)= $x^7 + x^6 + 1$.The output of both scramblers were tested against an input pattern of 10,000 bits. The result is showcased in Fig. 12. Fig. 12 (a, b) shows the input and output run length plots for 10,000 bit input patterns.

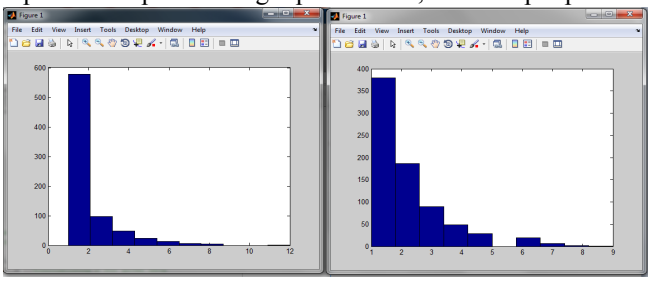

Figure 12(a). Input and output run length plots for the new scrambler design

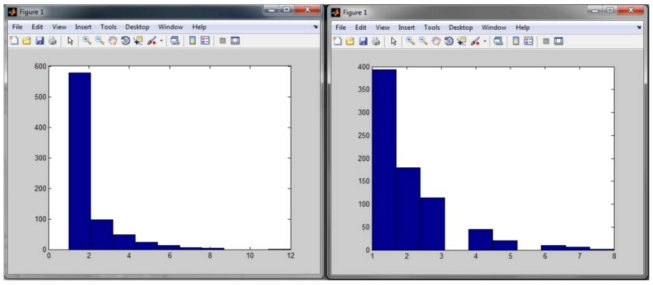

Figure 12(b).Input and output run length plots for existing scrambler design

From the above results, it is evident that new scrambler has less number of sequences with large run length and therefore potentially efficient for high speed communication systems.

The implemented 64b/66b encoding technique is highly efficient as it requires an overhead of only 3.18% and therefore potentially efficient for transferring large volume of data in Mega Bytes. A brief comparison of implemented 64b/66b technique with existing 8b/10b technique is shown in Table-V

TABLE-V: COMPARATIVE OVERVIEW 8B/10B&64B/66B ENCODING

Encoding Tech.	8B/10B	64B/66B
Run length	5	Relies on Scrambler
DC Balance	Running Disparity of 2	Not Guaranteed
Word Synchronization	"Comma" K-Characters	Synchronization Header
Control Characters	K-Characters	Control Codes
Overhead Fraction	25%	3.175% (11% @ CRC)
Implementation Technique	Look Up Table Approach	scrambling + non-scrambled sync pattern & control type

V. CONCLUSION

The proposed design occupies only a small portion of the available logic resources, indicating that it can be integrated with other functional blocks of the sub-system. The implemented 64b/66b encoding technique is highly efficient as it requires overhead of only 3.18% and therefore is potentially efficient for transferring large volumes of data in size of Mega Bytes. We have implemented 64b/66b line encoder with parallel input and CRC for SERDES application for the first time, however literature [9,22,27] discussed them as serial input and serial output of data. The proposed 64b/66b line encoding technique reduces additional overhead significantly with respect to conventional 8b/10b. Therefore this technique is advantageous for high speed communication. Selection of polynomial involved in scrambling/descrambling process plays a critical role in determining efficiency of the scrambler. Various polynomials have been generated and tested in MATLAB. The polynomials thus generated are implemented in VHDL. To decrease the run length of output sequences further we propose the use of an additional scrambler serially with the existing scrambler. Hardware verification has been successfully completed and demonstrated on Actel ProASIC 3C. The output of both scramblers are verified against input of pattern of 10,000 bits. The paper discussed theoretical, practical and implementation aspects of the 64b/66b line encoding technique in details.

ACKNOWLEDGEMENT

The authors express their deep gratitude to Prof. Joycee Mekie, Dept. of Electrical Engineering, IIT Gandhinagar, for her extended support and guidance during the design implementation and verification. The authors are thankful to Shri S.S Sarkar, Deputy Director, Sensor Development Area, Shri Arup Roy Choudhury, Group Director SEG and Shri Sanjeev Mehta, Space Applications Centre, Ahmedabad for motivation during design and implementation. The authors acknowledge with thanks to Shri Tapan Kumar Mishra, Director, SAC, ISRO, Ahmedabad, for insightful discussions.

REFERENCES

[1] Dave Lewis, "SERDES Architecture and Applications", an application note by National Semiconductor Corporation.

[2] High Speed Serdes Devices and Applications James Donald, Amanullah Mohammad et.al .2009, Springer Publications.

[3] Datasheet of the COTS SERDES (TLK2711 , DS90UR241)

[4] Richard Taborek, Alderrou, Donald. Steve Dreyer, Rara Gary(2003).U.S. Patent No. 20,030,217,215. Washington,2003

[5] Behzad Razavi, Monolithic phase-locked loops and clock recovery circuits,Theory and Design.Wiley IEEE Publication.

[6] Rick Walker, "Clock and Data Recovery for Serial DigitalCommunication", Hewlett-Packard Company Palo Alto, California, ISSCC in the Short Course, wal02, February 2002.

[7] Istvan Haller and Zoltan Francise Baruch,High-Speed Clock Recovery for Low-Cost FPGAs, Computer Science Dept. Technical University of Cluj-Napoca, Romania, 2012.

[8] J.Lee, K..Kundert, B. Razavi, Analysis and Modeling of Bang-Bang Clock and Data Recovery Circuits,IEEE Journal of Solid State Circuits, in Vol 39/4 , April 2004, pp.613-621.

[9] Raahemi, Bijan. "Error correction on 64/66 bit encoded links." IEEE Canadian Conference on Electrical and Computer Engineering, Saskatoon (2005), SK in page. 412-416, May.

[10] Tanaka, Keiji, Akira Agata, and Yukio Horiuchi. "IEEE 802.3 10G-EPON standardization its research development status." Journal Lightwave Technology 28.4 (2010):in page. 651-661.

[11] Actel, "Implementing an 8B/10B Encoder/Decoder for Gigabit Etherneting the Actel SX Family", Application Note.

[12] Al X. Widmer and Peter A. Franaszek, "A DC-Balanced, Partitioned -Block, 8B/10B Transmission Code", IBM Journal of Research and Development. 27/5, 440-451 (1983).

[13] Byte oriented DC balanced(0,4)8B/10B partitioned block transmission code, in U.S. Patent 4,486,739, December 1984.

[14] XILINX, XAPP336 (version 1.3), "Design of a 16b/20b Encoder/Decoder Using a Cool Runner XPLA3 CPLD"v1.3

[15] "Data Encoding Techniques",web link: www.rhyshaden.com

[16] Lattice semiconductor Corporation, article titled "8B/10B Encoder/Decoder", Reference Designs, in article no RD1012.

[17] Carlitz, L. (1932). The arithmetic of polynomials in a Galois field. American Journal of Mathematics, 54(1), 39-50.

[18] Chen, H. L., & Li, H. (2005). Generation &Analysis of PN Sequence Based on MATLAB[J].ComputerSim.,5, 028.

[19] Ahmad, M., & Farooq.Chaos based PN sequence generator for cryptographic applications. In Multimedia, Signal Processing and Communication Technologies (impact),IEEE 2011 p.83-86

[20] Kitsos, P., Theodoridis, G., & Koufopavlou, O. (2003). An efficient reconfigurable multiplier architecture for Galois field GF (2 m).Microelectronics Journal, 34(10), pp.975-980.

[21] Games, Richard A. "Cross correlation of M-sequences and GMW-sequences with the same primitive polynomial." Discrete Applied Mathematics 12.2 (1985): page no.139-146.

[22] Walker, et.al."Decoding method and decoder for 64b/66b coded packetized serial data." In U.S. Patent. 6,650,638.2003, Agilent.

[23] Kim, S. C., & Lee, B. G. (1994). Synchronization of shift register generators in distributed sample scramblers. Communications, IEEE Transactions on,42(234), 1400-1408.

[24] Kantute, Ms Payal, and Mrs Jaya Ingole.Study of MB-OFDM Transmitter Baseband System.Innoviative Sc(2014).

[25] Weldon Jr, Edward J. (1988). U.S. Patent No. 4,723,246. Washington,DC:U.S. Patent and Trademark Office.02-02-88.

[26] Cypress, Understanding Bit Error Rate Hotlink,AN1047.

[27] Dickson, Leonard Eugene. (2003). Linear groups: With an exposition of the Galois field theory. Courier Corporation.

[28] Radiation-Tolerant ProASIC3 Low Power Spaceflight Flash FPGAs with Flash Freeze Technology, Datasheet, Rev.5, 2012.