HCTL Open Int. J. of Technology Innovations and Research
HCTL Open IJTIR, Volume 2, March 2013
e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-111-6

Development of FSM based Running Disparity Controlled 8b/10b Encoder / Decoder with Fast Error Detection Mechanism

***Abdullah-Al-Kafi*[*], *Rajib Imran*[†] *and Monirul Islam*[‡]**
kafi@fastrack-design.com, rajib@fastrack-design.com and monirul@fastrack-design.com

Abstract

Transmitted bits have some certain characteristics which have effects on the error rate and the achievable bandwidth. This characteristic includes (1) ratio of 0's and 1's within a data byte and (2) maximum no. of clock periods between bit transitions (i.e. 1 -> 0 and vice versa). These characteristics are maintained and developed using an efficient encoder. Encoder increase the ability of a receiver performing clock recovery or deriving improved bit synchronization, helps distinguished control bit sequence and data

[*]ASIC Design Engineer, Fastrack Anontex Limited Dhaka, Bangladesh.
[†]ASIC Design Engineer, Fastrack Anontex Limited Dhaka, Bangladesh.
[‡]ASIC Design Engineer, Fastrack Anontex Limited Dhaka, Bangladesh.

HCTL Open Int. J. of Technology Innovations and Research
HCTL Open IJTIR, Volume 2, March 2013
e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-111-6

bit sequences, allows simple detection of byte and word boundaries. In order to achieve DC balance serial data for faster clock recovery at receiver site this research gives a simple and practical solution for 8b/10b encoder/decoder. The Running Disparity method used here is FSM based and the two ROM look up tables have been made to accomplished perfect inversion. Besides we propose new method in error detection of 8b/10b encoder/decoder where *k_error* detect the command code error in encoder, *code_error* detect the invalid code input and *RD_error* detect the running disparity error in decoder. The design while detecting error can hold the last correct running disparity until the correct running disparity input. On other whole, 10-bit for each 8-bit of data algorithm drops the data rate speed relative to line speed.

Keywords

8b/10b encoders, DC balance, Run length, Running Disparity, Finite State Machine.

Introduction

The encoder-decoder scheme is used mainly over a high speed serial interface to transfer data. In the transmission section, the data is encoded prior sending, while in the receiving section, the data is decoded providing the reliable transmission with low error rates. In the serial interface, it is necessary to encode clock information within the data stream. Because most of the serial interfaces do not provide separate clock while identifying the individual valid bit. In order to have the superior transmission characteristics in the encoded data stream, the sufficient clock information is attached in such a way that the receiver can synchronize with the embedded clock information to achieve the successful data at required error rate. Thus the line characteristic is improved enabling long transmission distances and more effective error detection. Moreover the algorithm adds 25% overhead to each character, where approximately 20% channel is reserved for 8b/10b encoding overhead, which is a significant factor in the high speed data transmission. Therefore 8b/10b encoding algorithm is widely used by many high performance protocols include the Enterprise Systems Connection architecture (ESCON), PCI express, Serial ATA, Fiber channel, Serial rapid IO, USB 3.0, Serial Storage Architecture (SSA), and Gigabit Ethernet optical links.

HCTL Open Int. J. of Technology Innovations and Research
HCTL Open IJTIR, Volume 2, March 2013
e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-111-6

Functional Description

The 8b/10b encoding scheme was first proposed by Albert X. Widmer and Peter A. Franaszek of IBM Corporation in 1983. This coding scheme is widely used for high speed serial data transmission. In transmitter side, encoder encodes 8- bit parallel data into 10-bit parallel data. The 10-bit encoded parallel data converted to serial data using high speed parallel to serial register. This serial data stream will be transmitted through transmission media to the receiver. In receiver side, the encoded serial data converted into 10-bit parallel data using high speed serial to parallel register. Then the decoder decodes 10-bit encoded parallel data into 8-bit data. The 8b/10b encoded serial data stream is DC balance and has a maximum run-length without transition 5 which helps to recover data and clock at receiver site [4].

The DC balance data means that it has the same number of 1's and 0's for a given length of data. By making a sustained DC balanced data, we can avoid any polarization on the cable in high speed data transmission. For example 10-bit data stream, 1001110011 (6 1's and 4 0's) is not a DC balance data but 1001110100 (5 1's and 0's) is a DC balance data.

The run length is defined as the maximum number of contiguous 0's or 1's in a serial data stream. A less valued run length is helpful for fast clock and data recovery. The maximum run length of 8b/10b encode decoder is 5. On other hand, NRZI (Non Return to Zero Inverted) with bit stuffing have maximum run length 6.

In order to make DC balance data stream disparity concept is applied. Disparity is calculated by number of 1's minus number of 0's. The zero value is called neutral disparity. In 8b/10b encoding/decoding, the 10-bit divided into 4-bit and 6-bit. For 4-bit and 6-bit the disparity is $+2$ or -2. To make 10-bit encode value neutral disparity, 4-bit positive disparity ($+2$) is concatenated with 6-bit negative disparity (-2) or 4-bit negative disparity is concatenated with 6-bit positive disparity.

Running Disparity (RD) Finite State Machine is used to make disparity neutral serial data stream. This state machine has to state RD+ and RD−. RD− is the initial state of the state machine. Another block count the number of 1's in 10-bit encoded data and determine the running disparity for next input data [4].

HCTL Open Int. J. of Technology Innovations and Research
HCTL Open IJTIR, Volume 2, March 2013
e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-111-6

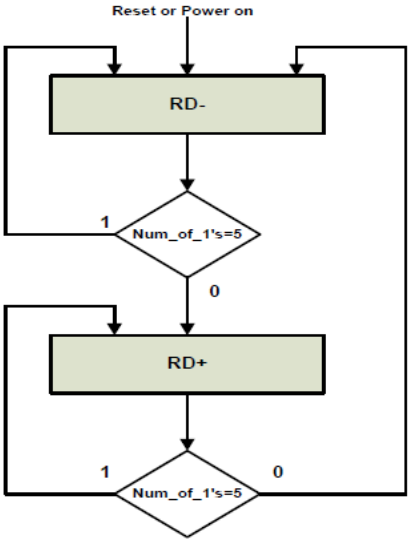

Figure 1: *Running Disparity Finite State Machine*

8b/10b Encoder Design

Our designed 8b/10b encoder divides into four blocks. They are (1) 5b/6b encoding block (2) 3b/4b encoding block (3) Internal signal generator block and (4) Running disparity FSM block. *Datain_8b*, clock, reset and *k_data* are inputs and *Dataout_10b*, *k_error* are outputs of encoder block.

Firstly, the 5b/6b encoding block encodes 5 bits input of *Datain_8b* [4:0] according to the current running disparity and *k_data* input. The 6X32 ROM table uses a sequential verilog case statement with the register version of the *Datain_8b* [4:0] inputs as selector. This block calculates the input data, current running disparity and *k_data* to determine the output of *Dataout_10b* [5:0]. Only fourteen 5 bits input data have 2 encode values and from them one encode value is determined by current running disparity [2].

The 3b/4b encoding block encodes 3 bits input of *Datain_8b* [7:5]. It implement by using 4X8 ROM table in sequential verilog case statement of *Datain_8b* [7:5] inputs as selector. This block calculate input data, *k_data*, flip six special case and current running disparity to determine the output of *Dataout_10b* [9:6].

e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-111-6

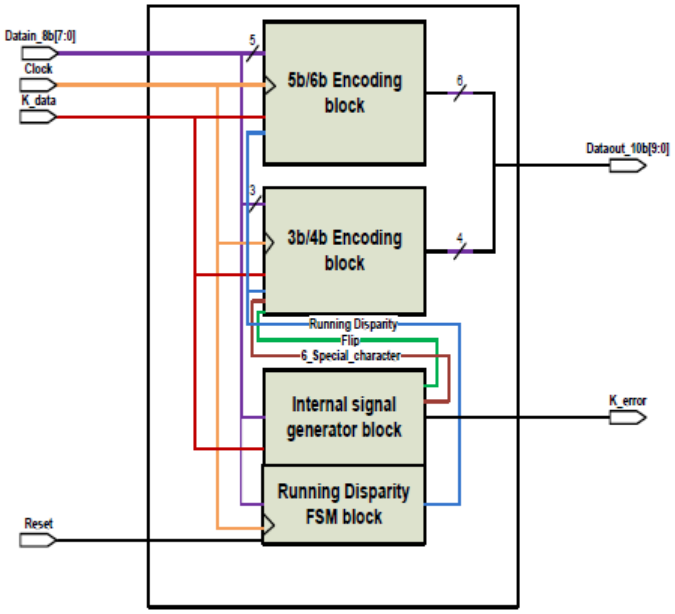

Figure 2: *8b/10b Encoder Block Diagram*

Internal signal generator block generates two internal signals six special case and flip to determine the encoding value. For some 5 bits inputs data values (3, 5, 6, 9, 10, 11, 12, 13, 14, 17, 18, 19, 20, 21, 22, 25, 26 and 28) the 6 bits encoding has only a single value and the 4 bits input data values (0, 3, 4, and 7) have a pair of encoding value. This two input data combination generates flip internal signal which inverts the current running disparity. There are six special cases, for first three (D11.7, D13.7 and D14.7) special case if the running disparity is positive then 4-bit encode value will be 1000. For second three (D17.7, D18.7 and D20.7) special case if the running disparity is negative then 4-bit encode value will be 0111 to maintain maximum run length 5 [2].

Finally, the running disparity FSM block generates current running disparity. The FSM contains two state one negative running disparity and positive running disparity. The default state of this FSM is negative running disparity. It calculates the number of one's to generate the current running disparity.

HCTL Open Int. J. of Technology Innovations and Research
HCTL Open IJTIR, Volume 2, March 2013
e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-111-6

8b/10b Decoder Design

Our designed 8b/10b decoder also divides into four blocks. They are (1) 6b/5b decoding block (2) 4b/3b decoding block (3) flip, *code_error* and *k_data* generator block (4) Running disparity FSM block. *Datain_10b*, clock and reset are inputs and *Dataout_8b*, *code_error*, *k_data* and *RD_error* are outputs of decoder block.

Firstly, the 6b/5b decoding block decode input encoded data *Datain_10b* [5:0]. Only 48 of the possible 64 values that can be received on bits 0 to 5 represent valid encode data. This can be modelled in verilog HDL code with case statement. There are 15 input data have two encode value for one decode data value out of 32 input data and other 17 input data have only one encode value for decode data. This 32 case also generate running disparity error and flip internal signal for 6 bit input data.

4b/3b decoding block decode input encoded data *Datain_10b* [9:6]. Only

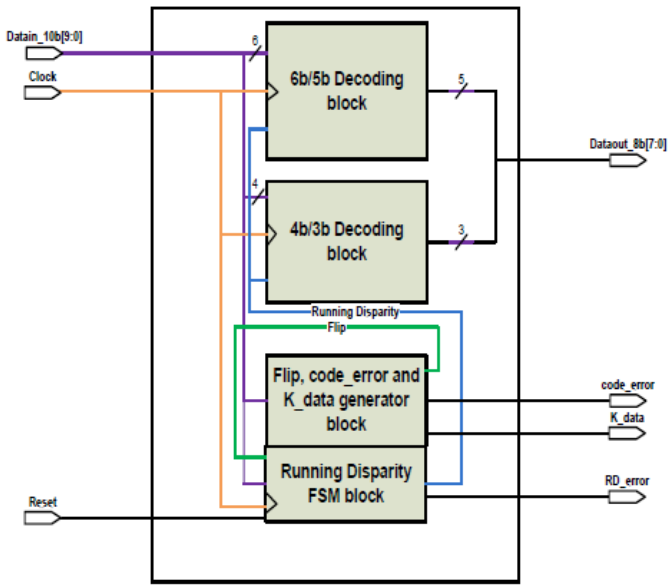

Figure 3: *8b/10b Decoder Block Diagram*

14 of the possible 16 values that can be received on bits 6 to 9 represent valid encode data. This can be modelled in verilog HDL code with case statement. There are 3 input data have two encode value for one decode data value out

e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-111-6

of 8 input data and other 4 input data have only one encode value for decode data. For decode data 7 there are four encode value (0001, 1110, 1000 and 0111). This 8 case also generate running disparity error and flip internal signal for 4 bit input data [2].

The Flip, code error and *k_data* generator block generate *code_error*, *k_data* output and flip internal signal. For 10 bit input, there are 1024 input can be generate and out of 1024 input only $(256+12)*2 = 536$ cases are valid and rest of 488 cases are invalid. For any of 488 cases *code_error* will generate. There are 24 valid k-code encoded inputs out of 1024 that generate valid 12 *k_data* output which indicate command character. 6b/5b decoding block generate $flip\_6b$ and 4b/3b decoding block generate $flip\_4b$. The AND of $filp\_6b$ and $flip\_4b$ generate flip signal which use to generate running disparity of input data.

Finally, the running disparity FSM block generates negative running disparity or positive running disparity for next input data using finite state machine. Another block generates running disparity from the input data. If the RD matches with the expected RD from running disparity FSM block then the RD is okay. If the RD not matches with expected RD from running disparity FSM then *RD_error* will generate and the running disparity FSM block hold the expected RD before getting correct RD input data. When the decoder block gets correct RD then it starts decoding again.

8b/10b Mapping and Calculation

The 8b/10b mapping converts 8 bit code groups into 10 bit codes. The mapping breaks the original data byte into two blocks: 1) the 3 most significant bits and 2) the 5 least significant bits. Lets name the 3 most significant bits H, G, F accordingly and the 5 bits as E, D, C, B and A [3].

The 3 bit then encoded to 4 bit j, h, g, f and the 5 bit is encoded into 6 bits named i , e, d, c ,b, a. Finally, the new 4 bit and the 6 bit then combined into a 10 bit encoded value.

Each character has a name format: Zxx.y
Z= [D=for a data character/K=for a control special character]
XX= Decimal value of the binary no composed of E, D, C, B, A of the unencoded 8 bit data.
Y= Decimal value of the binary no composed of H, G, F of the unencoded 8

e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-111-6

bit data.

Examples:
Data x60 = 8'b0110 0000 = D00.3
Data x8F = 8'b1000 1111 = D15.4
Ctrl xBC = 8'b1011 1100 = K28.5

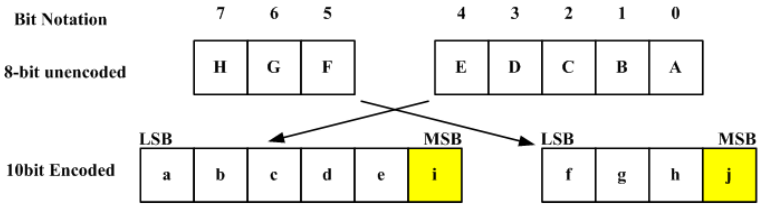

Figure 4: *Breaking structure of the data stream*

Encoding converts each 8 bit data byte into 2 possible 10 bit bytes. We need to ensure that the 1s and 0s are sustained to have a sustained positive DC on the transmit line. Each of the 256 Dxx.y data bytes has two corresponding transmission characters, one with positive disparity and another one is negative disparity. The RD− disparity will be either +2 or 0 (disparity neutral) and the RD+ disparity will be either −2 or 0. The encoder will choose the value based on the calculation of current Running disparity [6, 7, 8].

Table 1: *Random Input Data*

Input Data	Value (Hex)	RD+ abcdei fghj	RD− abcdei fghj
D00.3	60	100111 0011	011000 1100
D15.4	8F	010111 0010	101000 1101
K28.5	BC	001111 1010	110000 0101

The transmitter takes a first negative value (RD−) at start up. When an 8 bit data is encoding its will use values from RD− column to produce a disparity neutral 10 bit encoded data. As long as it is neutral, the RD will not be

e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-111-6

changed and the RD− column will still be used. If it is not disparity neutral, then the RD+ column will be used instead. Similarly, if the current disparity comes as positive and the 10 bit encoded data is disparity neutral, and then the RD will still be RD+. Otherwise it will be changed from RD+ back to RD− and the process will be running on and on. Most significantly, even though two characters separated by neutral disparity characters RD rules don't allow consecutive positive disparity or consecutive negative disparity characters for them [3].

Control data bits are somewhat different from the 256 standard data bytes. For example, k28.5 is a special control character widely used in data encoding decoding scheme. The sequence is made by 2 sequential bits of similar type, followed by 5 sequential bits of its opposite type which is used as a trigger for data recovery logic to identify 10 bit character boundaries [10, 11, 12].

Error Analysis Method

The 8b/10b encoder decoder detects only three errors which are command code error (*k_error*), decoder data error (*code_error*) and running disparity error (*RD_error*). *K_error* generates in encoder side and *code_error* and *RD_error* generate in decoder side. In this paper, we developed a fast error detection method [1].

Command code error generates when *k_data* input is high and input data is not command code input. For command code input the input data *datain_8b* [4:0] have only 5 values (11100, 10111, 11011, 11101, 11110). When input has other value and *k_data* input is high then *k_error* will generate. When the *k_error* is high then input data will not encoded and the running disparity FSM hold the last RD for next correct command code.

Verilog RTL code for command code error block:

```
always @(DataOut_i[4:0])
case(DataOut_i[4:0])
5'b11100:  k_valid <= 1'b1;
5'b10111:  k_valid <= 1'b1;
5'b11011:  k_valid <= 1'b1;
5'b11101:  k_valid <= 1'b1;
5'b11110:  k_valid <= 1'b1;
```

e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-111-6

```
default:  k_valid <= 1'b0;
endcase
assign k_error = k_data & (!k_valid);
```

In decoder side input is 10-bit so there are 1024 possible input values. Out of 1024 input value only 536 input values are valid and other 488 input are invalid values. When the decoder gets any one of invalid value then the decoder shows *code_error*. When the *code_error* generates it will not decode the encoded input data and running disparity FSM hold the last correct RD and wait for valid input data.

In the encoder every encoded data has a specific running disparity. When the data receive in the decoder if the running disparity does not match with expected running disparity then *RD_error* will generate. So 8b/10b decoder calculate two running disparity. One running disparity generate from input data another running disparity generate from the same running disparity FSM which is used in encoder block. The initial running disparity is negative then the next running disparity determined by previous input data which is expected running disparity. If the input data match with expected running disparity, decoder will decode the input data. If input data's running disparity does not match with expected running disparity then decoder will not decode the input data and RD FSM hold the last correct running disparity for another correct running disparity match [5].

Simulation and Result Analysis

Comparison with other encoding methods

Data encoding decoding can be performed in many ways such as Manchester encoding, 4b/5b encoding, NRZI encoding etc. All the techniques have been developed with their own pros and cons. However, for precise result in fast speed data transmission 8b/10b encoding/decoding scheme is widely used in modern TX and RX design. In this section we are representing a brief comparison among all the encoding method for better understanding of designers choice for 8b/10b encoding scheme [9, 10].

Manchester encoding consumes double bandwidth of the raw signal. For 10 Mbps transmission, the signal spectrum lies between the 5 and 20 MHz. For high speed data encoding and limited bandwidth transmission Manchester encoding is not efficient encoding method.

e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-111-6

4b/5b is a block encoding method encodes 4-bit data into 5-bit encoded data which break up long string of 1's and 0's without increasing frequency bandwidth. The five bit code are selected in a way so that, there can be no more than three consecutive zeros. Despite of occupying limited bandwidth and less consecutive zeros, the maximum run length is 8 which is a large drawback for clock recovery in receiving end.

Non Return to Zero Inverted (NRZI) is another simple encoding method. This method has no problem with large consecutive 1's but it create problem with large consecutive 0's. In large consecutive 0's it has no transition at encoded data. This problem solve using bit stuffing method with NRZI encoding. The maximum run length of NRZI encoding with bit stuffing is 6.

8b/10b encoding method solves all the drawbacks of previous encoding method. In 8b/10b encoding 1 Gbps data can transmit into 100 MHz bandwidth. The maximum run length of this method is 5. So this encoding method is very much efficient in limited bandwidth transmission, high speed encoding and most importantly faster clock recovery [13].

Simulation

We verify our 8b/10b encoder/decoder RTL code in **Quartus** software and simulate our code in **Modelsim** simulator. In 8b/10b Encoder simulation; we tested all kinds of inputs such as flip (00000011, 10000101), 6-special case (11101101, 11101101, 11110001, and 11110100), k-data (11011100, 11111110) and invalid k-data (11100000) shown in figure 4. For the reference in 4X8 ROM tables and 6X32 ROM tables we have used the predefined table presented by **Actel Corporation**. The encoder output is sequential and it outputted encoded data after one active edge clock of input data.

In 8b/10b decoder simulation; we also tested all kinds of inputs such as command code (0010111100, 0100111100, and 1010111100), wrong RD data (1000110101) and invalid encoded data (0000111111). For command code input, output successfully decode the 10-bit data and k-data output is high. For valid encoded wrong RD data input, decoder output *RD_error* and for invalid 10-bit data, decoder output *code_error* successfully.

HCTL Open Int. J. of Technology Innovations and Research
HCTL Open IJTIR, Volume 2, March 2013
e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-111-6

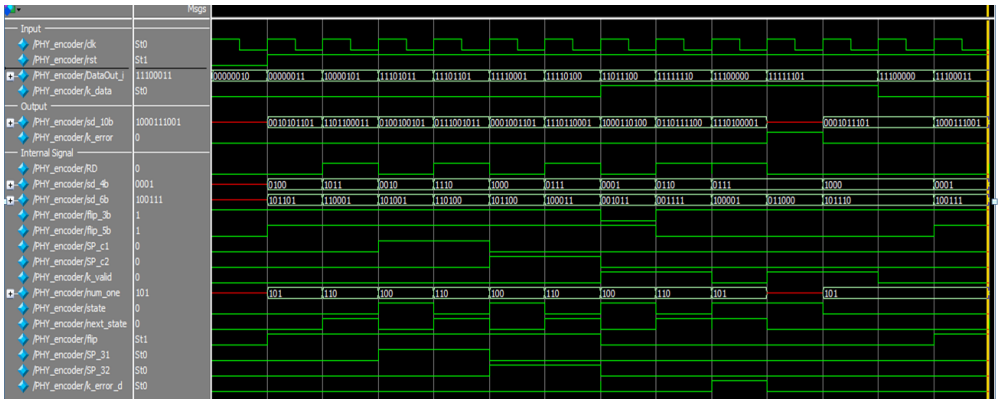

Figure 5: *8b/10b Encoder Simulation*

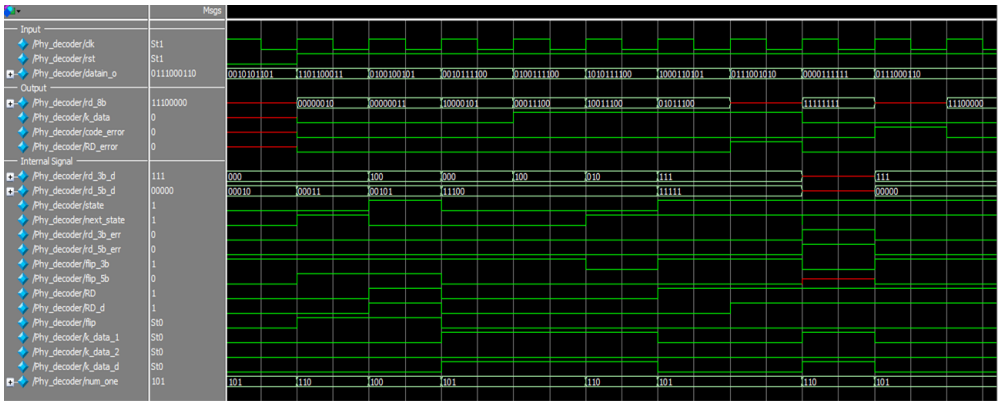

Figure 6: *8b/10b Decoder Simulation*

Conclusion

8b/10b encoder uses for encrypting clock information into transmitted data to recover the clock at receiver site. We have already developed the verilog HDL code of encoder and decoder. Our designed 8b/10b encoder/decoder meets all requirements specified in IEEE 802.3z and ANSI X3.230-1994. [4]. Three types of errors - *K_error*, *Code_error*, *RD_error* generated with this system are analyzed and developed in order to get precise and DC balanced data. For high performance receiver data recovery run length of 5 is avoided and delay is minimized. The maximum operating frequency of encoder/decoder is 105MHz. and this maximum operating frequency can be increase by including more logic in verilog RTL code. Our future goal is to extend for higher values in encoding

and decoding scheme. Theoretically we enlarged the design by breaking up words in 8-bit scheme and providing them a single clock individually to the smallest unit. Then at the receiving end, we can combine the whole to form our respected words. Otherwise for fastest data transmission, we can also enlarge the scheme to higher bit encoding which yet needs lots of researches.

References

[1] `IBM Research and Development Journal`, 1983, Volume 27(5).

[2] `Actel FPGAs to Implement the 100Mbps Ethernet Standard`, Actel, 1996.

[3] Widmer A X, Franaszek P A. `A DC -Balance, Partitioned Block 8B/10B Transmission Code` [J]. IBM Journal of Research and Development, 1983.

[4] `Lattice Semiconductor implemented 8b/10b Encoder/Decoder`, February 2012.

[5] `Altera designed 8b/10b Encoder/Decoder Megacore Function`, May 2011.

[6] Gallager RG., `Low density parity check codes` [J]. IRE Trans. Inform. Theory, 1962 (1): 21-28.

[7] Perez J M, Fernandez V. `Low-cost encoding of IEEE 802.11t` [J] Electronics Letters 2008, 44(4):1411-1412.

[8] `CYPRESS CY7B923 HOTLI NK Transmitter/ Receiver`, CY7B923 HOT Link datasheet [Z]. Cypress Semiconductor Corporation, 2003

[9] Franaszek P A, `Sequence State Coding for Digital Transmission`, Bell Syst. Tech J. 47. 143-157 (1968).

[10] `FC-1 8/10 Bit Encoding & Decoding`, http://www.atlantis-press.com/php/download_paper.php?id=2489

[11] Xu Qiaoyu, Liu Huijie, `8b/10b Encoder Design`, http://www.storagedocs.org/?page_id=73 The 2nd International Conference on Computer Application and System Modelling (2012).

e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-111-6 *References*

[12] Daniel Elftmann and Jing Hua Ma, `Implementing an 8B/10B Encoder/Decoder for Gigabit Ethernet`, http://www.eetasia.com/ARTICLES/1999JUN/1999JUN29_BD_NTEK_TAC1.PDF?SOURCES=DOWNLOAD

[13] Jon Tate, Geoff Cole, Ivo Gomilsek, Jaap van der Pijll, `Designing an IBM Storage Area Network`, http://www.scribd.com/doc/36479469/Desining-an-IBM-Storage-Area-Network

`Mr. Abdullah-Al-kafi` received the B.Sc. degree in Electrical and Electronic Engineering from American International University-Bangladesh (AIUB) in 2012. He is currently working as ASIC Design Engineer at Fastrack Anontex Limited Since 2012. His research interests include high speed data encoding decoding, Physical layer design, Digital Phase Lock Loop and clock multiplier.

`Mr. Rajib Imran` completed his Masters from San Jose State University, CA, USA. He has experiences in both the local and multinational engineering firms. At present, he is working as a design engineer in Fastrack Anontex Ltd, Dhaka, Bangladesh. He has hobbies in different research works, publications and technical writings. His research interest includes high speed data transfer, different error correction procedures, physical layer design, data link layer, different memories and universal serial bus etc.

`Mr. Monirul Islam` received the B.Sc. degree in Electrical and Electronic Engineering from Ahsanullah University of Science and Technology in 2010. Being an ASIC design engineer in Fastrack Anontex Limited, he researched on Digital Phase Locked Loop, PCI Express design, DDR# SDRAM memory controller. He has published articles in IEEE xplore digital library and is a program committee member of Student Research Symposium (SRS-2013).