International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 03 | Mar-2018 www.irjet.net p-ISSN: 2395-0072

FPGA implementation of orthogonal codes for efficient digital communication

Shubham Patne[1], Abhinav kalambe[2], Sushmit Ingle[3], Nitin Dongre[4], Anuradha Kondelwar[5]

[1,2,3,4] *Student & department of electronic and telecommunication, Pce Nagpur, Maharashtra, India*
[5]*Assistant professor & department of electronic and telecommunication, Pce Nagpur, Maharashtra, India*

Abstract - *There has been an increase in data transmission and noise & interference has also been increased in it. More efficient and reliable techniques have been developed for detecting and correcting errors in received data. There are various approaches and techniques used for the purpose of detecting and correcting errors. But still in data transmission there is a problem of data reliability. In this paper orthogonal codes have been used to detect and correct errors. Field programmable gate array (FPGA) is used for implementation. With the help of simulation this technique can detect upto 99.99% of errors and corrects it more efficiently.*

Key Words: **Orthogonal codes, FPGA, VHDL, error detection and correction.**

1. INTRODUCTION

There has been an increase in data transmission and noise & interference has also been increased in it. More efficient and reliable techniques have been developed for detecting and correcting errors in received data. There are various approaches and techniques used for the purpose of detecting and correcting errors. But still in data transmission there is a problem of data reliability. In this paper orthogonal codes have been used to detect and correct errors. Field programmable gate array (FPGA) is used for implementation. With the help of simulation this technique can detect upto 99.99% of errors and corrects it more efficiently. There are some techniques such as cyclic redundancy check (CRC) which only detects errors. Moreover there are again some techniques designed such as Solomon codes, hamming codes, which can detect as well as correct errors. But the problem with these techniques are that it cannot correct errors upto high efficiency.

A.ORTHOGONAL CODES

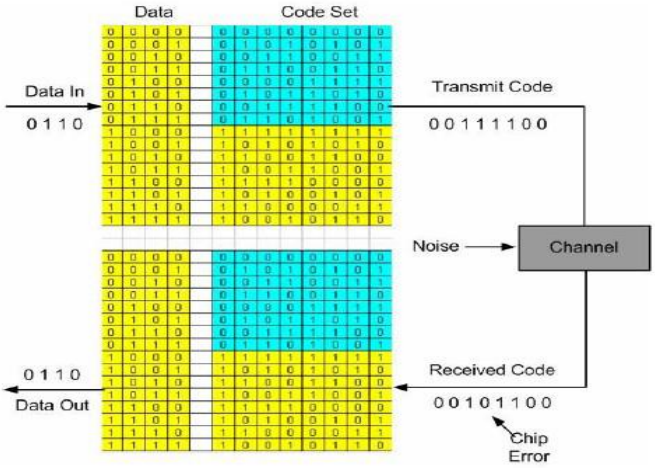
Orthogonal codes contains binary value. they consists same number of 1's and 0's. An n-bit orthogonal code has n/2 1's and n/2 0's, i.e, Therefore, all orthogonal codes will originate zero parity bits. The paper concept is thereby illuminated by 8-bit orthogonal codes as described in Fig. 1. It has 8 orthogonal codes and 8 antipodal codes for a total of 16 bi-orthogonal codes. Inverse of orthogonal codes are antipodal codes.

A notable distinction in this method is that the transmitter does not have to send the parity bit for the code, since it is known to be always zero. If there is a transmission error, By generating a parity bit at receiving end it can be detected by the receiver. Before transmission, a m-bit data set is designed into a unique n-bit orthogonal code. For example, a 4-bit data set is shown by a unique 8-bit orthogonal code, which is transmitted without the parity bit. The data is decoded based on code correlation upon receiving. It can be done by setting a threshold between two orthogonal codes. Following equation describes

$$d\,th = n/4$$

Where *n* is the code length and *dth* is the threshold, which is center between two orthogonal codes. Therefore, for the 8-bit orthogonal code (Fig. 2), we have *dth = 8/4 = 2*.

This structure provide a decision process in error correction, where the incoming imperfect orthogonal code is examined for correlation with the codes stored in a look-up table, for a possible match. The acceptance criterion for a valid code is that an n-bit comparison must yield a acceptable cross correlation value; otherwise, a false detection will occur. This is governed by the following correlation process, where a pair of n-bit codes *x1x2...xn* and *y1y2...yn* is compared to return.



Orthogonal Code												Antipodal Code								
0	0	0	0	0	0	0	0	0				1	1	1	1	1	1	1	1	0
0	1	0	1	0	1	0	1	0				1	0	1	0	1	0	1	0	0
0	0	1	1	0	0	1	1	0				1	1	0	0	1	1	0	0	0
0	1	1	0	0	1	1	0	0				1	0	0	1	1	0	0	1	0
0	0	0	0	1	1	1	1	0				1	1	1	1	0	0	0	0	0
0	1	0	1	1	0	1	0	0				1	0	1	0	0	1	0	1	0
0	0	1	1	1	1	0	0	0				1	1	0	0	0	0	1	1	0
0	1	1	0	1	0	0	1	0				1	0	0	1	0	1	1	0	0
0	0	0	0	0	0	0	0	P				0	0	0	0	0	0	0	0	P

The concept is illustrated by means of an 8- bit orthogonal code as shown in Fig.1. It has 8-orthogonal codes and 8-antipodal codes for a total of 16-bitorthogonal codes. Antipodal codes are just the inverse of orthogonal codes; they are also orthogonal among themselves. The same concept, of improving error detection using orthogonal codes, can be extended for 16-bit orthogonal codes. There are 16, 16-bit orthogonal codes and 16, 16-bit antipodal codes total of 32 bi-orthogonal codes. Antipodal codes are opposite to that of orthogonal codes. Since there is an equal number of 1's and 0's, each orthogonal code will generate a zero parity bit. Therefore, each antipodal code will also generate a zero parity bit. A notable distinction in this method is that the transmitter does not have to send the parity bit since the parity bit is known to be always zero [7]. Therefore, if there is a transmission error, the receiver will be able to detect it by generating a parity bit at the receiving end. Before transmission a k-bit data set is mapped into a unique n-bit. For example, a 4-bit data set is represented by a unique 8-bit orthogonal code which is transmitted without the parity bit. When received, the data are decoded based on code correlation. It can be done by setting a threshold midway between two orthogonal codes. This is given by the following equation D= n/4. Where n is the code length and it is the threshold, which is midway between two orthogonal codes.

Here XOR operation is performed between the received code and each code in the look-up table. A counter is used to count the number of 1's in the resulting signal. For example, for 8-bit orthogonal code, we get sixteen counter results. If any one bit of the results is zero, it means there is no error. The corrected code is correlated with the minimum count. yet it is not possible to correct the corrupted code if the minimum count is correlated with more than one combination of the orthogonal codes. Therefore, for the 8-bit orthogonal code we have dth = 8/4 = 2.

2. METHODOLOGY

A. Design methodology:-

Since there is an equal number of 1's and 0's, each orthogonal code will generate a zero parity byte. If the data has been corrupted during the transmission the receiver can detect errors by generating the parity bit for the received code and if it is not zero then the data is corrupted. However the parity bit doesn't change for an even number of errors, hence receiver can detect the 2n/2 error means 50 %.Our approach is not to use the parity generation method to detect the errors. But a simple technique based on the comparison between the received code and all the orthogonal code combinations stored in a look up table.

B. Transmitter:-

Transmitter comprises encoder and Parallel to Serial Conversion. Mapping unit encode the incoming 5-bit data to corresponding 16-bit orthogonal code. To store orthogonal codes we have used a look-up table. For Example input to the transmitter is "00001" this corresponds to 16-bit orthogonal code "0101010101010101". Parallel to Serial Conversion converts the 16-bit code set into a serial bit stream which is to be transmitted from the transmitter.

C. Receiver:-

Receiver functionality comprises several steps i.e. Serial to Parallel Conversion, Splitting of orthogonal code, Error Detection and Correction and Decoder. Incoming serial bit stream is first converted to 16 bit parallel code. This received code either can be impaired or correct one. Now received code is processed with each code in the lookup table. Now the split-Orthogonal come into picture. We have split received code and each orthogonal code in the look-up table in two equal parts. Then XORed each time and count the number of 1's in the output. We will get two counts while counting both XORed outputs. These two counts are checked for different conditions and will work inter-dependently and then decision is taken for correct orthogonal code. Beside detection and correction with these two different counts we can detect the position of the error in the received code, so that we can enhance the effectiveness of over all trans-reception system.

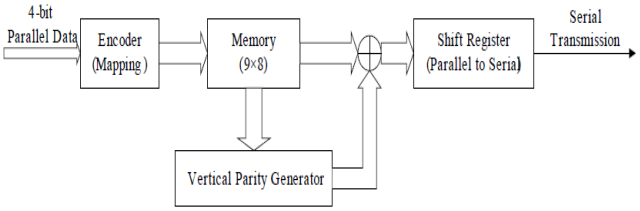


Fig.3. Block diagram of the transmitter.

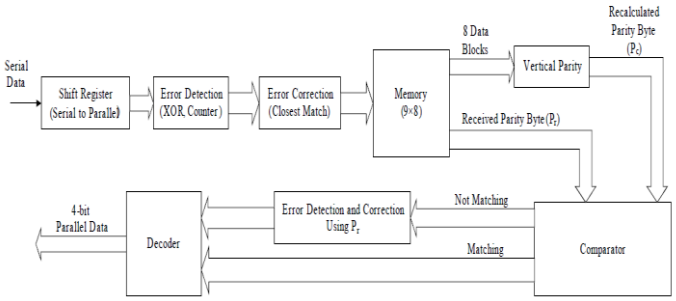


Fig.4. Block diagram of the receiver.

III.IMPLEMENTATION AND RESULTS

In this setup DE-2 cyclone-4 chip (FPGA) is used for implementation with the help of simulation software Xilinx ISE 13.2. Problem cannot be detected if an orthogonal code turns over to another orthogonal code in previous work. Although, with the help of OCCMP, if one block in the package changes to another orthogonal code, then parity byte is used to detect the error. Assuming that the number of blocks in the memory is k (including the parity byte). When more than one block in this memory turns over to other

orthogonal codes and if there is an even number of errors in one column, the errors cannot be detected.

$$n_i = \begin{cases} C_k^i \cdot \left(C_{2n-1}^1\right)^{\frac{i}{2}} & \text{for } i \text{ even} \\ C_k^i \cdot C_{2n-1}^1 \cdot \left(C_{2n-1}^2\right)^{\frac{i-1}{2}} & \text{otherwise} \end{cases} \quad (4)$$

Where *i=2, 3,..., k* is the number of corrupted blocks, and *ni* is the amount of combinations that cannot be detected.

The total number of combinations that cannot be detected for *k* blocks of n-bit code is *N* given by:

$$N = \sum_{i=2}^{k} n_i \quad (5)$$

For 9 blocks of 8-bit code,

$$\begin{aligned} N &= \sum_{i=2}^{9} n_i \\ &= C_9^2 \cdot C_{15}^1 + C_9^4 \cdot \left(C_{15}^1\right)^2 + C_9^6 \cdot \left(C_{15}^1\right)^3 + C_9^8 \cdot \left(C_{15}^1\right)^4 \\ &\quad + C_9^3 \cdot C_{15}^1 \cdot C_{14}^1 + C_9^5 \cdot C_{15}^1 \cdot \left(C_{14}^1\right)^2 + C_9^7 \cdot C_{15}^1 \cdot \left(C_{14}^1\right)^3 \\ &\quad + C_9^9 \cdot C_{15}^1 \cdot \left(C_{14}^1\right)^4 \\ &= 3214095 \end{aligned}$$

The detection rate for *k* block of n-bit code is

$$\frac{2^{k \cdot n} - N}{2^{k \cdot n}} \times 100\% \quad (6)$$

This gives for OCCMP-8 a detection rate of

$$\frac{2^{72} - 3214095}{2^{72}} \times 100\% = 99.99\%$$

The example of 8-bit orthogonal codes can be used for error correction the same can be used for explaining different schemes:

Closest Match technique can be used to detect and correct error when one error originate in one block code.

If there are two error in one block code, the corrupted codes can be corrected with the help of vertical parity byte. If two errors occur in different blocks, then Closest Match is used to correct them. Now if three errors occur, incidental to the locations, the Closest Match, vertical parity, or both can be used to detect and correct the corrupted block codes. If four errors turns up, incidental on the locations, some errors can be detected and corrected, and some errors can only be detected but not corrected. Convincingly, this technique can correct up to 3-bit errors. In general, for n-bit orthogonal code, it can correct (n/2-1)-bit errors.

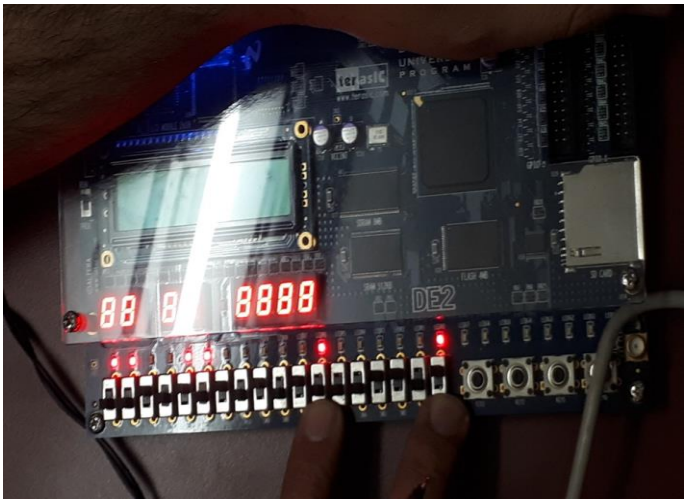


Fig -1: Altera DE2 board

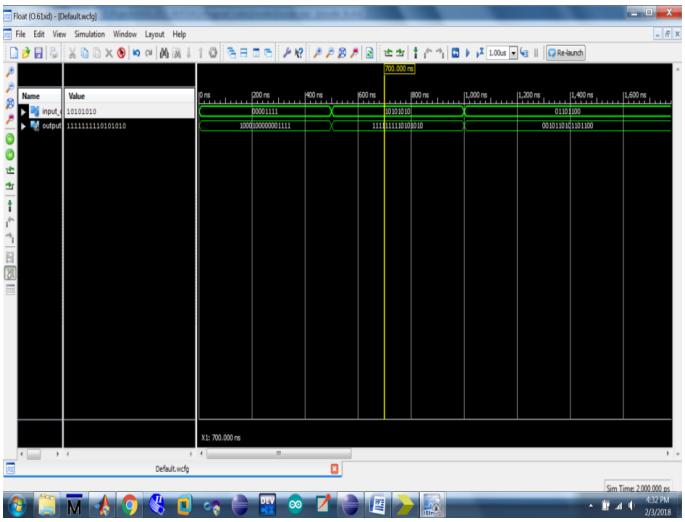


Fig -2: Simulated output

3. CONCLUSIONS

In this paper, we have designed and implemented a realtime high-efficiency technique to detect and correct errors in digital communication. The proposed technique can detect 99.99% of errors and correct up to (n/2-1)-bit of errors for n-bit orthogonal codes as the experimental results shows.

TABLE IV

COMPARISON OF DETECTION AND CORRECTION CAPABILITY OF DIFFERENT TECHNIQUES.

Techniques	Detection Rate	Correction Capability (n_c)
OCC-8	71.88%	1
OCCM-8	93.57%	1
CRC-32	99.99%	0
OCCMP-8	99.99%	3

TABLE V

COMPARISON OF CORRECTION CAPABILITIES BETWEEN OCC, OCCM AND OCCMP.

Codes length (*Bit*)	Techniques		
	OCC	OCCM	OCCMP
8	1	1	3
16	3	3	7
32	7	7	15
64	15	15	31
n	(n/4)-1	(n/4)-1	(n/2)-1

REFERENCES

[1] Douglas L . perry "VHDL programming by Example" Mc Graw Hill , 4th Edition.

[2] Nazein M. Botros "HDL programming Fundamentals: VHDl & verilog"

[3] Volnei A . Pedroni "Circuit Design & Simulation with VHDL, 2nd Edition

[4] Brown, Digital Logic with VHDL Design

[5] Stephen Brown;V Zvonko " Fundamentals of digital logic with VHDL",Mc Graw Hill International 2ndEdition,2006

[6] Charles H,Roth,Jr,"Digital System Design using VHDL,2nd Edition

[7] Mark Z wolinski, "Digital System Design with VHDL" Prentice Hall, 2000

[8] Jikku Jeemon "pipelined 8- bit RISC processor design using Verilog HDL on FPGA"IEEE international conference in electronics information communication Technology, pp. 2023 -2027

[9] Anlei Wang, Member, IEEE, and Naima Kaabouch, Member, IEEE Department of Electrical Engineering, University of North Dakota, Grand Forks, ND 58202-7165