FPGA Implementation of Orthogonal Code Convolution for Efficient Digital Communication

*Raghvendra Dubey & P. Lakshmi Sarojini
Department of Electronics & Communication Engineering
Swarnandhra College of Engineering and Technology, Seetharampuram, Narsapur (A.P.), INDIA

Abstract

In digital communication system, convolution coding is preferred for the channel coding as it facilitates a better error correction as comparison to block coding which does not require memory. Among other techniques such as Cyclic Redundancy and Solomon Codes; orthogonal coding is one of the codes which can detect errors and correct corrupted data in an efficient way. In this paper, FPGA implementation of orthogonal code convolution is presented by employing Xilinx and Modelsim softwares. It is found that the orthogonal code implementation improved the error detection upto 99.9%. With this method, the transmitter does not have to send the parity bit since the parity bit is known to be always zero. Therefore, if there is a transmission error, the receiver will be able to detect it by generating a parity bit at the receiving end.

Keywords: *Code Convolution, Orthogonal Codes, Antipodal Codes, FPGA.*

1. Introduction

Information and communication technology has brought enormous changes to our life and turned out to be one of the basic building blocks of modern society. Day by day, there is an increasing demand of network capacity due to the use of internet and real time transmission of voice and picture. To fulfil these requirements data transmission at high bit rates is essential for various aspects such as video, high-quality audio and mobile integrated service digital network (ISDN). However, the data transmitted at high bit rates over mobile radio channels, leads to inter symbol interference (ISI). The significant factors which cause the reliability of digital data communication are the transmission medium i.e. cable or air, sources of noise and some others like electromagnetic interface, crosstalk and distance. To overcome this problem, error correction coding is a solution for the best possible communication. The main advantage of using coding is the efficiency of the channels use becomes higher as comparison to the case when code is not used. Therefore, error detection and correction techniques are needed which can detect errors such as the Cyclic Redundancy Check and others which can detect as well as correct errors such as Solomon Codes [1-3]. The CRC check includes table driven CRC calculation and loop driven CRC calculation however, this application describes the implementation of the CRC-16 polynomial. Further, there are several formats for the implementation of CRC such as CRC-CCITT, CRC-32 or other polynomials. The CRC generation has many advantages over simple sum techniques or parity check. CRC error correction gives the detection of single, double and bundled bit errors and useful where large data packages are transmitted. Reed–Solomon (RS) codes are non-binary cyclic error-correcting codes which described a systematic way of building codes that could detect and correct multiple random symbol errors. This coding has found its applications from deep-space communication to consumer electronics (CDs, DVDs, Blu-ray Discs). Among these methods, orthogonal code is one of the codes which can detect errors and correct corrupted data in an efficiently with increased quantity of data transmitted [4]. This coding is binary valued and with equal number of 1's and 0's. All orthogonal codes can generate zero parity bits as n-bit orthogonal code has n/2 1's and n/2 0's. In simple there are n/2 positions where 1's and 0's differ and hence, each antipodal code can also generate a zero parity bit [5]. It is noted that with this method, the transmitter does not have to send the parity bit since the parity bit is known to be always zero. Therefore, if there is a transmission error, the receiver will be able to detect it by generating a parity bit at the receiving end.

In this paper, the FPGA implementation of orthogonal code convolution is presented by employing Xilinx and Modelsim softwares; in section second and third, the theory of orthogonal coding and design approach are presented. The simulated results and analysis are discussed in section fourth. Finally, section fifth concludes the paper.

2. Theory of Orthogonal Coding

Orthogonal codes are consists of equal number of 1's and 0's e.g. n-bit orthogonal code consist n/2 1's and n/2 0'. Meaning, there are n/2 positions where 1's and 0's differ. In this way, all orthogonal codes generate zero parity bits. An illustration of 16-bit orthogonal code is shown in figure 1.

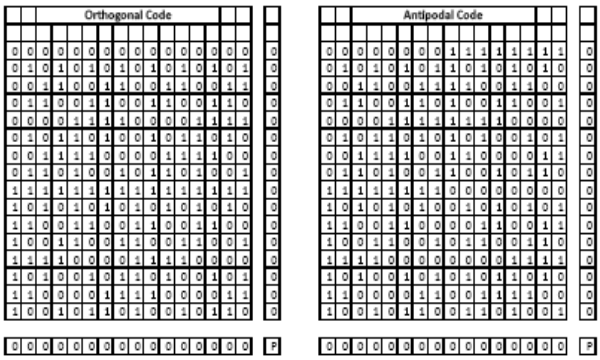

Figure 1. A 16-bit orthogonal code has 16 orthogonal codes and 16-antipodal codes for a total of 32 bi-orthogonal codes.

It is comprised of 16-orthogonal codes and 16-antipodal codes (just the inverse of orthogonal codes) for a total of 32 bi-orthogonal codes as depicted in figure 2. The advantage with this approach is that transmitter does not need to send the parity bit as parity bit is known to be always zero. In this way, if error exists during, the receiver can detect by generating a parity bit at the receiving end. In orthogonal coding, a k-bit data set is mapped into a unique n-bit before transmission. Here, we have considered a 5-bit data set which is can be represented by a unique 16-bit orthogonal code and transmitted without the parity bit. After receiving the data, it is decoded based on code correlation by setting a threshold midway between two orthogonal codes. The threshold midway is represented as

$$d_{th} = \frac{n}{4} \qquad (1)$$

Where n is the code length and dth is the threshold midway between two orthogonal codes. According to above equation, for 16-bit orthogonal coding, threshold midway is 4 between two orthogonal codes. This approach offers a decision process, where the incoming impaired orthogonal code is examined for correlation with the neighbouring codes for a possible match. It is noted that the acceptance criterion for a valid code is that an n-bit comparison must yield a good autocorrelation value; otherwise, a false detection will occur.

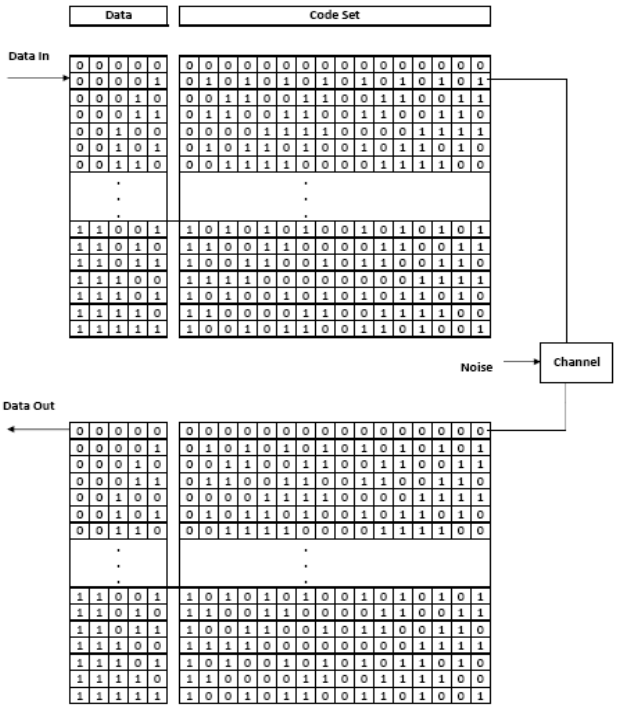

Figure 2. Encoding and Decoding Process.

This is governed by the correlation process, where a pair of n-bit codes $x_1, x_2, x_3, \ldots x_n$ and $y_1, y_2, y_3, \ldots y_n$ is compared to yield, autocorrelation value which is given as

$$R(x,y) = \sum_{i=1}^{n} x_i y_i \le \frac{n}{4} - 1 \qquad (2)$$

Where $R(x,y)$ = autocorrelation function, n= code length, and d_{th} =threshold midway. For reliable detection, an additional 1-bit offset is added to equation (2).

Further, we can estimate the average number of errors which can be corrected by combining equation (1) and equating (2), and can be represented as

$$t = n - R(x,y) = \frac{n}{4} - 1 \qquad (3)$$

Here, t is the number of errors which can be corrected by means of an n-bit orthogonal code.

3. Design Approach

Our design approach is based on the comparison between the received code and all the orthogonal code combinations stored in a look up table; which has two major components such as a transmitter and a receiver. The first component (transmitter) consists of two blocks such as encoder and shift register which is shown in figure 3.

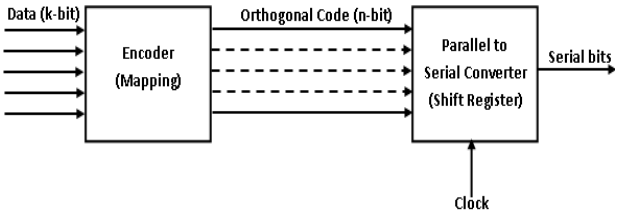

Figure 3. Block diagram of the 5-bit data transmitter.

Here, encoder encodes a k-bit data set to n=2k-1 bits of the orthogonal code and the shift register transforms this code to a serial data in order transmit these. In our case of 5-bit data, it can be encoded to 16-bit (24) orthogonal code according to the lookup table shown in figure 3. Further, the generated orthogonal code can be serially transmitted by using a shift register with the rising edge of the clock.

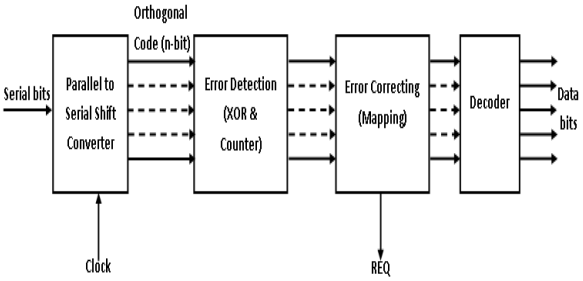

Figure 4. Block diagram of Receiver.

The second component (receiver) consists of two major blocks such as serial to parallel converter and decoder as shown in figure 4. Here, the incoming serial bits are converted into n-bit parallel codes and compared with all the codes in the lookup table for error detection. This is processed by counting the number of ones in the signal resulting from 'XOR' operation between the received code and each combination of the orthogonal codes in the lookup table. Further, counter is used to count the number of ones in the resulting n-bit signal and also searches for the minimum count. However, a value rather than zero shows an error in the received code. The orthogonal code in the lookup table which is associated with the minimum count is the closest match for the corrupted received code. The matched orthogonal code in the lookup table is the corrected code, which is then decoded to k-bit data. The receiver is able to correct up to (n/4)-1 bits in the received impaired code. Signal (REQ), goes high when the minimum count is associated with more than one combination of orthogonal code.

3.1 Encoder Simulation

We have done RTL simulation of encoder to ensure the proper working of stand alone module. The encoder reset, using the reset signal 'reset'. This resets the encoder to the default value "0000000000000000". The encoder is then enabled using the signal 'data_rdy'. This signal is HIGH when the data is ready.

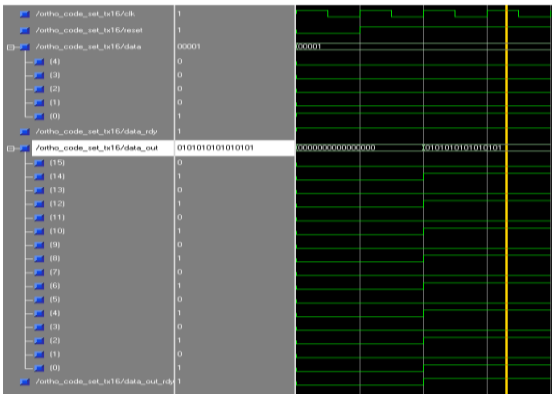

Figure 5. Simulated result of encoder.

The input data is encoded to 16-bit orthogonal code with the rising edge of the clock signal. The 16-bit orthogonal code is outputted through a signal 'data_out'. The signal 'data_out_rdy' is using to indicate the availability of output data. This signal is HIGH when the output data is ready. For example, the 5-bit data "00001" is encoded to "0101010101010101" 16-bit orthogonal code, this is shown in figure 5.

3.2 Parallel to Serial Shift Register Simulation

The shift register reset, using the reset signal 'reset'. This resets the shift register to the default value "0000000000000000". The shift register is then enabled using the signal 'data_rdy'.

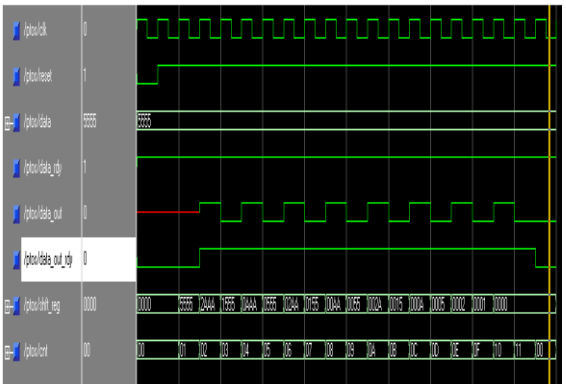

Figure 6. Simulated result of parallel to serial shift register.

This signal is HIGH when the data is ready. The parallel input data is loaded into a temporary register 'shft_reg'. The simulated result of parallel to serial shift register is shown in figure 6. The shift register transmits the bits serially using a signal 'data_out' with the rising edge of the clock signal. The signal 'data_out_rdy' is used to indicate the availability of output data. This signal is HIGH when the output data is ready

3.3 Transmitter Simulation

After simulating encoder and parallel to shift register, the transmitter is simulated and results are shown in figure 7. The transmitter reset, using the reset signal 'reset'. This resets the transmitter to the default value "00000". The encoder encodes a k-bit data set to n=2k-1 bits of the orthogonal code and the shift register transforms this code to a serial data in order to be transmitted. The transmitter is then enabled using the signal 'data_rdy'. This signal is HIGH when the data is ready. For example, the 5-bit data "00001" is encoded to "0101010101010101" 16-bit orthogonal code. The generated orthogonal code is then transmitted serially using a shift register with the rising edge of the clock. The bits are outputted through a signal 'data_out'. The signal 'data_out_rdy' is using to indicate the availability of output data. This signal is HIGH when the output data is ready.

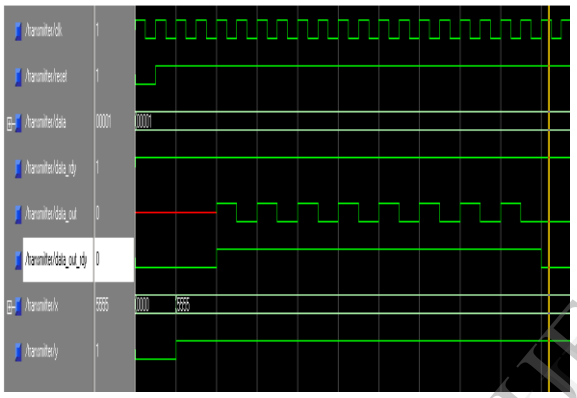

Figure 7. Simulated result of Transmitter

3.4 Serial to Parallel Shift Register Simulation

The sub module of receiver, serial to parallel shift register is simulated and shown in figure 8. The shift register reset, using the reset signal 'reset'. This resets the shift register to the default value "0000000000000000". The shift register is then enabled using the signal 'data_rdy'. This signal is HIGH when the data is ready. The input data bits are serially loaded into a temporary register 'shft_reg' with the rising edge of the clock signal. After the final input bit is clocked in the signal 'cnt' is greater than "10000".

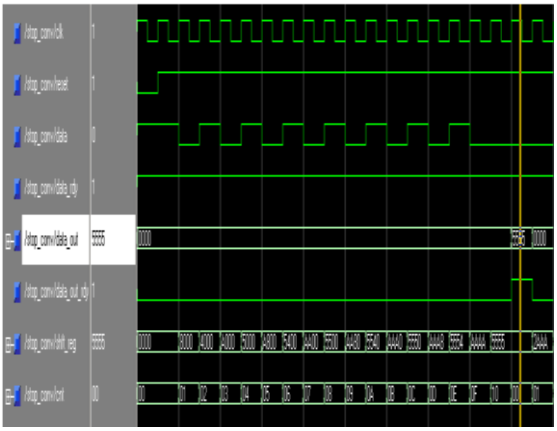

Figure 8. Simulated result of serial to parallel shift register.

When the 'cnt' value is greater than "10000" the data in temporary register is available at the output signal 'data_out'. The signal 'data_out_rdy' is using to indicate the availability of output data. This signal is HIGH when the output data is ready.

3.5 Counter Simulation

The simulated result of counter module is shown in figure 9. The counter reset, using the reset signal 'reset'. This resets the counter to the default value "00000". The counter is then enabled using the signal 'data_rdy'. This signal is HIGH when the data is ready. The counter counts the number of 1's present in the given 16-bit data and produces the count value at the signal 'cnt_out'. The signal 'cnt_rdy' is using to indicate the availability of output data. This signal is HIGH when the output data is ready.

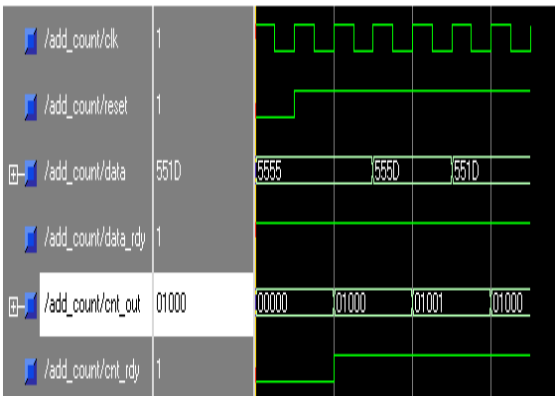

Figure 9. Simulated result of Counter.

3.6 Receiver Simulation

The receiver is simulated with its components such as serial to parallel converter and counter; and results are shown in figure 10. The receiver reset, using the reset signal 'reset'. This resets the receiver to the default value "00000". The receiver is then enabled using the signal 'data_rdy'. This signal is HIGH when the data is ready. The 16-bit code at the signal 'data' is compared with all the

codes in the lookup table for error detection. This is done by counting the number of ones in the signal resulting from 'XOR' operation between the received code and each combination of the orthogonal codes in the lookup table. A counter is used to count the number of ones in the resulting 16-bit signal and also searches for the minimum count. However a value rather than zero shows an error in the received code. The orthogonal code in the lookup table which is associated with the minimum count is the closest match for the corrupted received code. The matched orthogonal code in the lookup table is the corrected code, which is then decoded to 5-bit data which is available at signal 'data_out'. The signal 'data_out_rdy' is using to indicate the availability of output data. This signal is HIGH when the output data is ready.

For example, if the 16-bit data is "0101010101010101", is XORed with each combination of 16-bit orthogonal codes in the lookup table. The resulting 16-bit data is given as an input to counter. The counter counts the number of 1's in each 16-bit XORed output. The minimum count value is "00000" for the orthogonal code is "0101010101010101". Therefore the associated 5-bit data is "00001". Another example, if the 16-bit data is "0101010101011101", is XORed with each combination of 16-bit orthogonal codes in the lookup table. The resulting 16-bit data is given as an input to counter. The counter counts the number of 1's in each 16-bit XORed output. The minimum count value is "00001" for the orthogonal code is "0101010101010101". Therefore the associated 5-bit data is "00001".

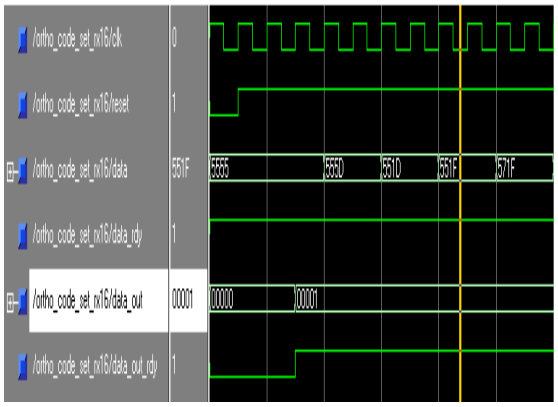

Figure 10. Simulated result of receiver.

4. Results and Discussion

In previous section, module by module simulation is done. Finally transmitter and receiver modules are combined and simulated. figure 11 shows the RTL schematic of transmitter and receiver.

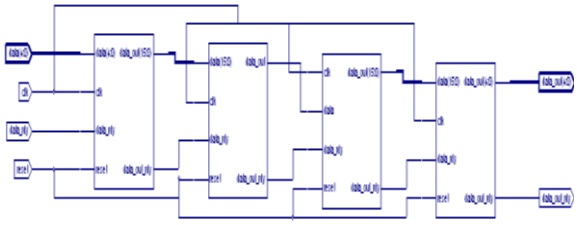

Figure 11. RTL Schematic of transmitter and receiver.

The Transmitter and Receiver reset, using the reset signal 'reset'. This resets the Transmitter and Receiver to the default value "00000". The receiver is then enabled using the signal 'data_rdy'. This signal is HIGH when the data is ready. For example, the input data value "00001" labeled as 'data' has been encoded to the associated orthogonal code "0101010101010101" labeled as "p_data". The signal 'p_data_rdy' is used to enable the transmission of the serial bits 'p_data' of the orthogonal code with every rising edge of the clock.

Upon reception, the incoming serial data is converted into 16-bit parallel code 'p_data1'. Counter is used to count the number of 1's after XOR operation between the received code and all combinations of orthogonal code in the lookup table. The signal 'cnt' gives the minimum count of one's among them. The orthogonal code 'p_data' associated with the minimum count is the closest match for the received code, which is then decoded to the final data given by signal 'data_out'.

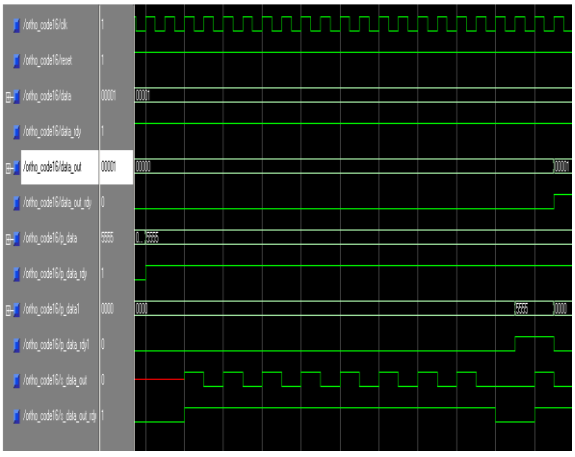

Figure 12. First Case: Simulated result of Transmitter and Receiver.

We have considered five cases to observe the output from above simulation. In first case, the received code has a match in the lookup table. As shown in figure 12, the received code is p_data1 = "0101010101010101", the value of minimum count is "00000" and hence the received code is not corrupted. The code is then decoded to the

corresponding final data “00001” which is given by signal ‘data_out’.

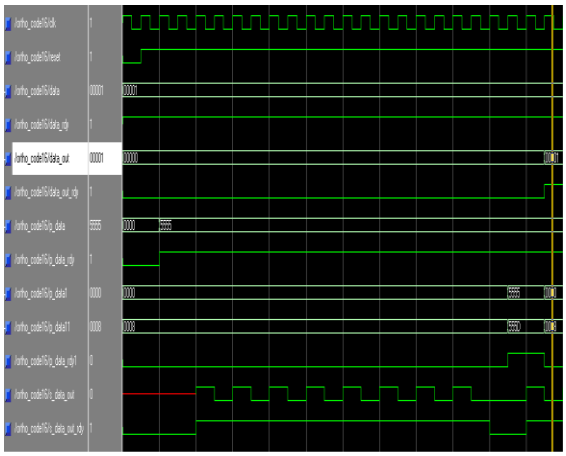

Figure 13. Second Case: Simulated result of Transmitter and Receiver.

In second case, the received code has no match in the lookup table. As shown in figure 13, the received code is p_data11 =“0101010101011101”, the value of minimum count is ‘00001’, which reveals an error. The corresponding orthogonal code is “0101010101010101” which is the closest match for the received code given by the minimum count, and the decoded final data is “00001” ” which is given by signal ‘data_out’. In this case the single bit error is detected and corrected by the receiver.

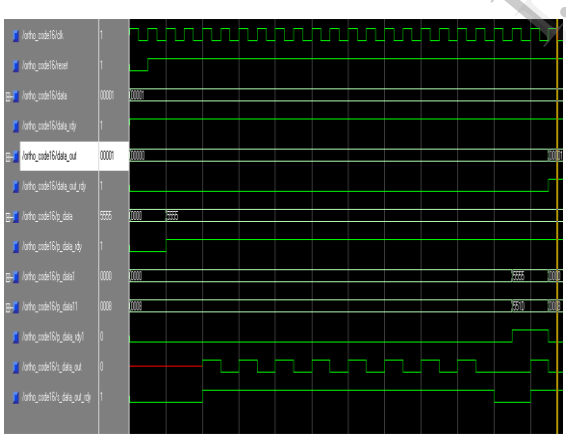

Figure 14. Third Case: Simulated result of Transmitter and Receiver.

In third case, the received code has no match in the lookup table. As shown in figure 14, the received code is p_data11 = “0101010100011101”, the value of minimum count is ‘00002’, which reveals an error. The corresponding orthogonal code is “0101010101010101” which is the closest match for the received code given by the minimum count, and the decoded final data is “00001” which is given by signal ‘data_out’. In this case the two bit errors are detected and corrected by the receiver.

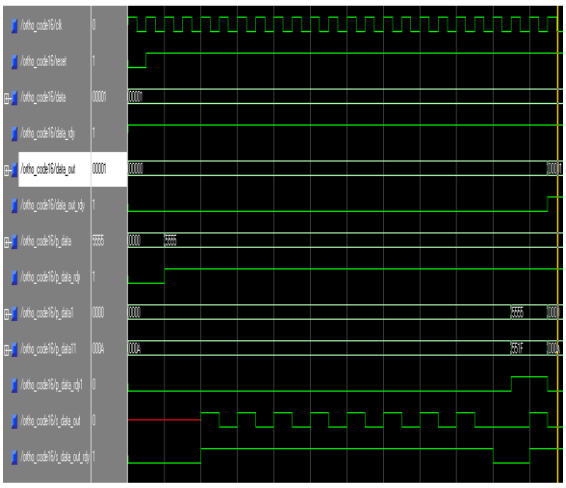

Figure 15. Fourth Case: Simulated result of Transmitter and Receiver.

In fourth case, the received code has no match in the lookup table. As shown in figure 15, the received code is p_data11 = “0101010100011111”, the value of minimum count is ‘00003’, which reveals an error. The corresponding orthogonal code is “0101010101010101” which is the closest match for the received code given by the minimum count, and the decoded final data is “00001” which is given by signal ‘data_out’. In this case the three bit errors are detected and corrected by the receiver.

In fifth case, there is more than one possibility of closest match in the lookup table. As shown in figure 16, the received code is p_data11 = “0101011100011111”. The value of minimum count is associated with more than one orthogonal code and thus it is not possible to determine the closest match in the lookup table for the received code.

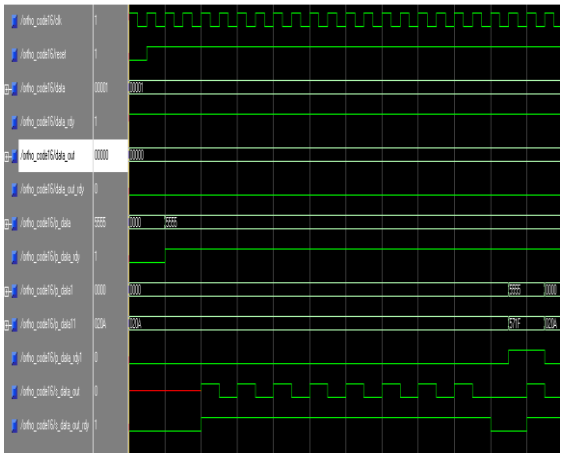

Figure 16. Fifth Case: Simulated result of Transmitter and Receiver.

In summary, 5-bit data is encoded into 16-bit orthogonal code which has $2^5 = 32$ combinations of orthogonal code. Therefore, out of 65,536 possible combinations of 16-bit received code the receiver will not able to detect error in those codes which are one of the combinations of orthogonal code. The detection percentage for 16-bit orthogonal code is estimated to be 99.95% which is able to correct three bit error.

5. Conclusion

In present paper, FPGA implementation of orthogonal code convolution is presented to ensure the efficient digital communication. This work involved the implementation of various modules of the transmitter and receiver using VHDL. A fully synthesizable HDL code was written to ensure that the design was feasible. This orthogonal code implementation has improved the error detection upto 99.9% for 16-bit coding. It is noted that with this method, the transmitter does not have to send the parity bit since the parity bit is known to be always zero. Therefore, if there is a transmission error, the receiver will be able to detect it by generating a parity bit at the receiving end. Finally, this work has the future scope of further improvement in orthogonal coding for large digital data processing.

References

[1] Baicheva, T., S. Dodunekov, and P. Kazakov, "Undetected error probability performance of cyclic redundancy- check codes of 16-bit redundancy," *IEEE Proc. Comms.*, Vol. 147, No. 5, Oct. 2000, pp. 253- 256.

[2] A. Hokanin, H. Delic, S. Sarin, "Two dimensional CRC for efficient transmission of ATM Cells over CDMA," *IEEE Communications Letters*, Vol. 4, No. 4, April 2000, pp.131-133.

[3] Stylianakis V., Toptchiyski S, "A Reed-Solomon coding/decoding structure for an ADS modem," Electronics, Circuits and Systems, 1999. *Proceedings of ICECS apos; 99.* The 6th IEEE International Conference, Volume 1, Issue , 1999, pp. 473 – 476.

[4] Saleh Faruque, "Error Control Coding Based on Orthogonal Codes," *Wireless Proceedings*, Vol. 2, pp. 608-615, 2004.

[5] Naima Kaabouch, Aparna Dhirde, and Saleh Faruque, "Improvement of the Orthogonal Code Convolution Capabilities Using FPGA Implementation," *IEEE Proceedings*, Nov. 2007, pp.337-341.