ISTANBUL TECHNICAL UNIVERSITY
ELECTRICAL - ELECTRONICS ENGINEERING FACULTY

Implementation Of Zigbee Protocol On Physical Layer On An Fpga

BSc Thesis by

Efe Emre Pazarli

040110222

Department: Electronics and Communication Engineering

Program: Electronics and Communication Engineering

Supervisor: Assoc. Prof. Dr. Sıddıka Berna ÖRS YALÇIN

JANUARY 2017

ISTANBUL TECHNICAL UNIVERSITY
ELECTRICAL - ELECTRONICS ENGINEERING FACULTY

IMPLEMENTATION OF ZIGBEE PROTOCOL ON PHYSICAL LAYER ON AN FPGA

BSc Thesis by

Efe Emre PAZARLI

040110222

Department: Electronics and Communication Engineering

Program: Electronics and Communication Engineering

Supervisor: Assoc. Prof. Dr. Sıddıka Berna ÖRS YALÇIN

JANUARY 2017

ACKNOWLEDGEMENT

I would like to thank to my supervisor Assoc. Prof. Dr. Sıddıka Berna Örs Yalçın to give me the opportunity of working with her and guiding me on this subject during my final thesis although I had chosen the communication branch. Finally, I would like to emphasize that I owe to my friends, my girlfriend Çağla Akay for her endless support and my family who has the biggest role on my successes for my entire life.

January, 2017 Efe Emre Pazarlı

INDEX

ABBREVIATIONS

CRC : Cyclic Redundancy Check
FCS : Frame Check Sequence
FPGA : Field Programmable Gate Array
IEEE : Institute of Electrical and Electronics Engineering
ITU-T : International Telecommunications Union-Telecommunications
LPF : Low-Pass Filter
LR-WPAN : Low-Rate Wireless Personal Area Network
LSB : Least Significant Bit
MAC : Medium Access Control
MFR : MAC Footer
MHR : MAC Header
MPDU : MAC Protocol Data Unit
MSB : Most Significant Bit
MSDU : MAC Service Data Unit
OEM : Original Equipment Manufacturer
O-QPSK : Offset-Quadrature Phase Shift Keying
OSI : Open Systems Interconnection
PAN : Personal Area Network
PHR : PHY Header
PHY : Physical Layer
PN : Pseudo-Random Noise
PPDU : PHY Protocol Data Unit
PSDU : PHY Service Data Unit
RF : Radio Frequency
SFD : Start Frame Delimiter
SHR : Synchronization Header
VHDL : Very High Speed Integrated Circuit Hardware Description Language
WLAN : Wireless Local Area Network
WPAN : Wireless Personal Area Network

LIST OF TABLES

LIST OF FIGURES

ZIGBEE PROTOKOLÜ FİZİKSEL KATMANININ BİR FPGA ÜZERİNDE GERÇEKLENMESİ

ÖZET

Bu tez, ZigBee protokolünün ve IEEE 802.15.4 standardının özelliklerinin anlatılmasını, fiziksel katmanın FPGA üzerinde gerçeklenmesini ve elde edilen sonuçları içermektedir. Bu konunun seçilme amacı FPGA üzerinde gerçekleştirilen projelerin kablosuz iletişime ihtiyaç duyması halinde, kablosuz iletişimi sağlayacak cihazın dış bir modül olarak kullanılması değil de yine FPGA blokları içerisinde gerçeklenip esneklik ve kolaylığın arttırılmasıdır. Piyasada hali hazırda ZigBee modülleri yer alırken, bu modüllerin satın alınıp FPGA kartına dış bir modül olarak takılması verimliliği ve esnekliği düşürmektedir. Farklı kullanımlarda farklı özellikteki modüllere ihtiyaç duyulacağından bu, proje için dışa bağımlılığı arttıracaktır. ZigBee modülünün FPGA üzerinde gerçeklenmesi sayesinde, farklı koşullarda değiştirilmesi gereken parametreler jenerik olarak değiştirilebilir ve başka projelerde de yazılan kodun tekrar tekrar kullanılması sağlanabilir. Bu yapılan işe hem esneklik hem de düşük maliyet getirecektir.

Bu tezde sadece fiziksel katman gerçeklenmiştir. Öncelikle izlenmesi gereken yol haritası çıkartılmış, kaynaklar belirlenmiş ve araştırılmıştır. Yapılan araştırmalardan sonra fiziksel katmanın da belli bloklara ayrılarak ayrı ayrı incelenmesi ve oluşturulması uygun bulunmuştur. Her bir alt blok kendi içinde incelenmiş, kodları ayrı ayrı yazılmış ve test edilmiştir. Bu yöntem sayesinde bütüne geçmeden önce parçaların çalışabilirliği doğrulanmış hata payı en aza indirgenmiştir. Son olarak bütün küçük parçalar bir araya getirilmiş, alıcı ve verici ortaya çıkartılmıştır. Gerekli tasarım ve test ortamları MATLAB ve Vivado araçları kullanılarak oluşturulmuştur.

IMPLEMENTATION OF ZIGBEE PROTOCOL ON PHYSICAL LAYER ON AN FPGA

SUMMARY

In this thesis, features of ZigBee protocol and IEEE 802.15.4 standard, implementation of PHY (Physical Layer) of IEEE 802.15.4 are explained in detail. The reason that I have chosen this topic is if any project needs wireless communication, to implement wireless device with FPGA's (Field Programmable Gate Array) own blocks would be more flexible and practical instead of using an external module. There are some ready-made ZigBee modules, but using ready modules will decrement efficiency and flexibility. Requirement for different ZigBee modules with another features for different projects will cause more dependency of the project to the outside factors. By implementing ZigBee on FPGA, created generic parameters can be modified according to different cases. Written code can be re-used again and again at different projects. This will lead low cost and more efficiency and flexibility.

Only PHY of IEEE 802.15.4 is implemented in this final thesis. First of all, the methods should be followed are discussed and sources are found and investigated deeply. It has been decided that the best way would be to divide the whole layer to the sub-modules. Each sub-module is investigated, the codes are written and tested separately. With this proper method, the possible errors are reduced to minimum and each module is verified one by one. Finally, all sub-modules are composed and receiver and transmitter are obtained. Required test and design environments are implemented in MATLAB and Vivado tools.

1 INTRODUCTION

WLAN (Wireless Local Area Network) and WPAN (Wireless Personal Area Network) technologies are rapidly growing with the development of new emerging standards [2]. ZigBee is one of those standards which takes place of Bluetooth with lower data rate applications and low power consumption.

ZigBee is designed as LR-WPAN (Low Rate-Wireless Personal Area Network) standard which is commonly used for wireless sensor networks, home control applications etc. The main purpose of ZigBee standard is maintaining a wireless network with low cost, low power and low data rate. For this reason, PHY (Physical) and MAC (Medium Access Control) layers are standardized with IEEE 802.15.4 which is Wireless MAC and PHY layer specifications for LR-WPANs [7]. The higher protocol layers are specified by Zigbee Alliance which is an alliance consists of group of companies that publish and maintain the standard. Application layers can be specified by ZigBee Alliance or other Original Equipment Manufacturers (OEM) [Figure 2.1].

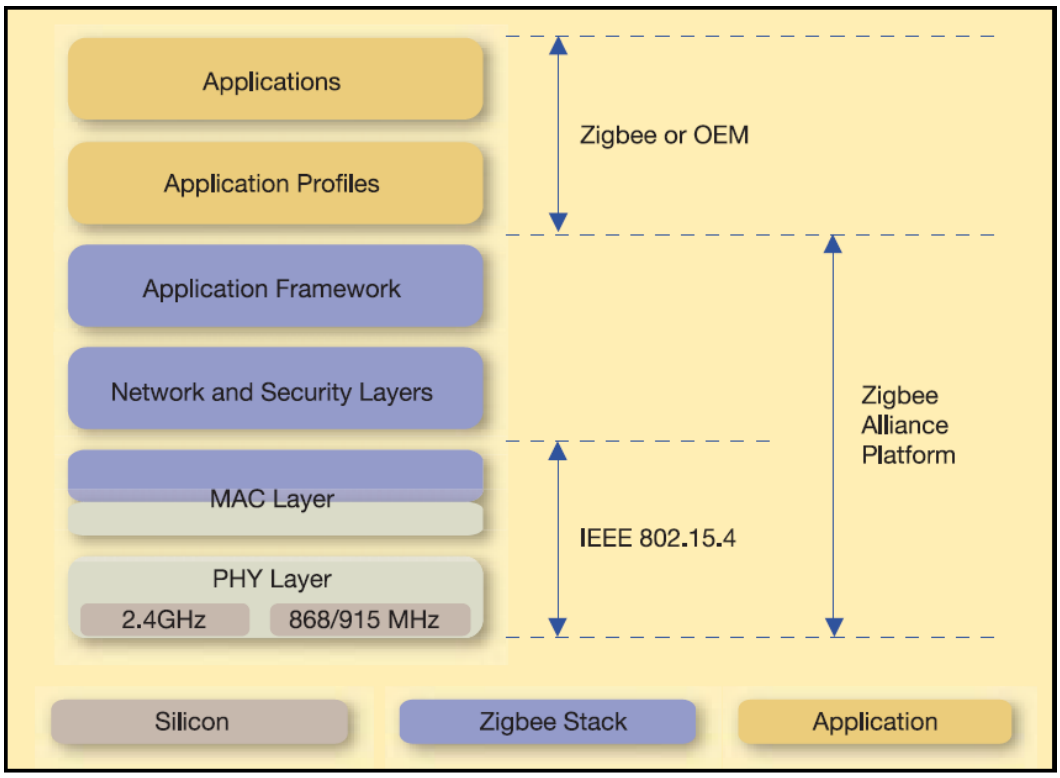


Figure 1.1: ZigBee Standard Layer [7]

IEEE 802.15.4 standard defines the operations of the LR-WPANs. WPANs are used to carry information over relatively short distances. There is no need of any infrastructure for WPAN connections. So, wide range of devices implement this small, power efficient, inexpensive solution. The PHY and MAC layers for LR-WPANs are specified by this protocol. It supports low data rate wireless connection for devices which can be fixed, portable or mobile devices. Battery consumption is so small that can give just such a personal operating space of 10 m. So, it can be said that there is a trade-off between longer range, low data rate and battery life. The requirements for IEEE 802.15.4 standard depends on the used application. The data rate for this standard will be maximum 250 kbps which is enough to satisfy simple needs. The minimum rate will be 20 kbps can be used in sensor and automation needs [1].

2 IEEE 802.14.5 PHYICAL LAYER AND MEDIUM ACCESS LAYER

The architecture of LR-WPAN is based on OSI (Open Systems Interconnection) seven layer model [1]. Each layer has some responsibility for a function of the standard and make services for the higher layers. Main layers for IEEE 802.15.4 are PHY and MAC layers. PHY layer contains RF (Radio Frequency) transceiver and its control mechanism. MAC layer accesses to all PHY channels to transmit the data.

2.1 PHY

PHY has following main tasks: Enabling the reception and transmission of PPDUs (PHY Protocol Data Unit) through the radio channel, selecting the channel frequency, transmitting and receiving the data. A proper LR-WPAN device can assign several frequency bands by modulation and spreading methods. The supported frequency bands and data rates are listed in Table 2.1.

Table 2.1: Supported Frequency Bands and Data Rates [1]

PHY (MHz)	Frequency Band (MHz)	Spreading Parameters		Data Parameters		
		Chip Rate (kchip/s)	Modulation	Bit Rate (kb/s)	Symbol Rate (ksymbol/s)	Symbols
868/915	868-868.6	300	BPSK	20	20	Binary
	902-928	600	BPSK	40	40	Binary
2450	2400-2483.5	200	O-QPSK	250	62.5	16-ary Orthogonal

2.1.1 Channel Assignments

Total of 27 channels are used in IEEE 802.15.4 standard numbered 0 to 26. Sixteen channels for 2450 MHz, ten for 915 MHz and one for 868 MHz band are assigned. The center frequencies can be calculated by following equations:

$$F_c = 868.3\ MHz\ for\ k = 0 \quad (2.1)$$

$$F_c = 906 + 2(k-1)\ MHz\ for\ k = 1, 2, \dots, 10 \quad (2.2)$$

$$F_c = 2405 + 5(k-11)\ MHz\ for\ k = 11, 12, \dots, 26 \quad (2.3)$$

k is the channel number.

2.1.2 PHY Protocol Data Unit Format

PPDU packet consists of following basic fields [1]:

- SHR field (Synchronization Header), allows receiving device to synchronize with coming data stream
- PHR field (PHY header), keeps frame length information
- Payload with variable length, the data comes from MAC layer

General packet structure is shown in Figure 2.2. The format is presented as the leftmost field should be transmitted or received first. Also each octet starts with least significant bit (LSB).

Octets: 4	**1**	**1**		**variable**
Preamble	SFD	Frame Length (7 bits)	Reserved (1 bit)	PSDU
SHR		PHR		PHY payload

Figure 2.1: General PPDU Packet Structure [1]

- Preamble field contains 32 bits of all zeros. This field is used by transceiver for synchronization.
- SFD field indicates that preamble finishes and data comes. The value of this field for IEEE 802.15.4 is 11100101 which starts with LSB.
- Frame Length field contains the length of PSDU (PHY Service Data Unit) coming from MAC layer.
- PSDU comes from MAC layer.

2.1.3 2450 MHz PHY Specifications

In this thesis 2450 MHz PHY properties of IEEE 802.15.4 are used. So, specifications will be introduced with details. The other frequency bands have similar characteristics. They aren't mentioned in the introduction section.

First of all, the data rate for 2450 MHz PHY is 250 kbps. It employs 16-ary quasi-orthogonal modulation. Four data bits are composed to create a symbol. Sixteen different nearly orthogonal pseudo-random noise (PN) sequences are represented by sixteen different combinations. Each symbol is represented with 32 chips. Chip rate is 2 Mchip/s such that the number of information bits is multiplied by eight as in the equation 2.1:

$$250kbps * 8 = 2Mchip/s \quad (2.4)$$

Finally PN sequences are modulated using offset quadrature phase-shift keying (O-QPSK). A block diagram for reference modulator can be seen in Figure 2.3.

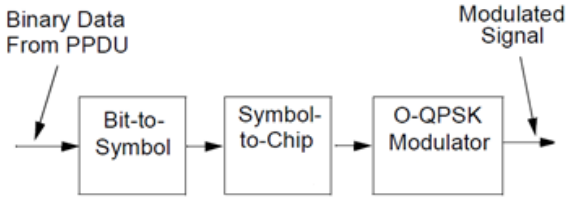


Figure 2.2: Reference Modulator Diagram [1]

2.1.3.1 Bit-to-Symbol Mapping

Four information bits create one symbol. This means there are sixteen different possible combinations between 0000 and 1111. The combinations are used for chip mapping. Each octet is symbolized as 4 LSBs (b_0, b_1, b_2, b_3) (Least Significant Bit) map into one symbol, the rest (b_4, b_5, b_6, b_7) maps into another symbol. The LSBs are processed first in every step of modulator shown in Figure 2.3.

2.1.3.2 Symbol-to-Chip Mapping

Relevant chip sequences are listed in Table 2.2. Every symbol will be mapped into one of those 32-chip PN sequences. 16 different chip sequences are related to each other through cyclic shifts and/or conjugation (i.e. inversion of odd indexed chips).

Table 2.2: Symbol-to-Chip Mapping [1]

Data Symbol (Decimal)	**Data Symbol (Binary) (b_0 b_1 b_2 b_3)**	**Chip Values (c_0 c_1 c_2 … c_{31})**
0	0000	11011001110000110101001000101110
1	1000	11101101100111000011010100100010
2	0100	00101110110110011100001101010010
3	1100	00100010111011011001110000110101
4	0010	01010010000101110110110011100011
5	1010	00110101001000101110110110011100
6	0110	11000011010100100010111011011001
7	1110	10011100001101010010001011101101
8	0001	10001100100101100000011101111011
9	1001	10111000110010010110000001110111
10	0101	01111011100011001001011000000111
11	1101	01110111101110001100100101100000
12	0011	00000111011110111000110010010110
13	1011	01100000011101111011100011001001
14	0111	10010110000001110111101110001100
15	1111	11001001011000000111011110111000

2.1.3.3 O-QPSK Modulation

Symbols are represented with chip sequences with a rate of 2 Mchip/s. For 2450 MHz PHY, all chips are modulated onto the carrier by use of O-QPSK (Offset-Quadrature Phase Shift Keying) and half-sine pulse shaping [5]. The chips are separated to I (in-phase) carrier for even-indexed bits and Q (quadrature-phase) carrier for odd-indexed bits. The only difference between QPSK and O-QPSK is setting an offset between I-phase and Q-phase carrier. To set offset, Q-phase chips are given a delay of T_c (inverse of the chip rate) with respect to I-phase (Figure 2.4).

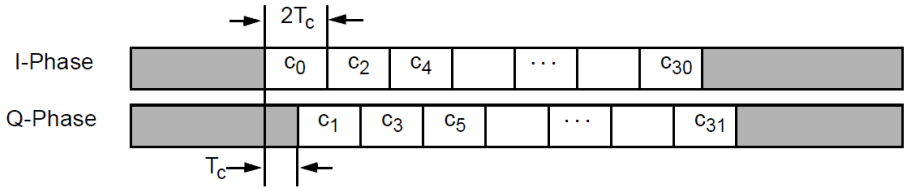


Figure 2.3: O-QPSK [1]

2.1.3.4 Pulse Shape

Each chip should be represented with a half-sine pulse shape as described in following equation:

$$p(t) = \begin{cases} \sin\left(\pi \frac{t}{2T_c}\right), 0 \le t \le 2T_c \\ 0, otherwise \end{cases} \quad (2.5)$$

Figure 2.5 shows a half-sine pulse shaped chip sequence.

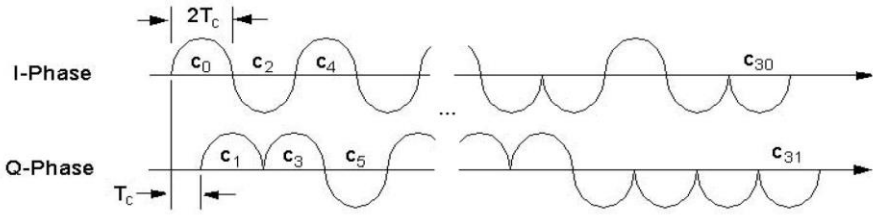


Figure 2.4: Half-Sine Pulse Shaped Chip Sequence [5]

2.2 MAC Layer

MAC layer manages to transmit and receive MPDUs (MAC Protocol Data Unit) through PHY layer and makes the communication with upper layers [1].

2.2.1 General MAC Frame Format

MPDUs basically consists of [1]:

- MHR (MAC Header), contains frame control, sequence number, address information.
- MAC payload, includes specific information of the frame type. The length can vary.
- MFR (MAC footer), contains FCS (Frame Check Sequence)

General MAC frame format can be illustrated as in Figure 2.6.

<table>
<tr><td>Octets: 2</td><td>1</td><td>0/2</td><td>0/2/8</td><td>0/2</td><td>0/2/8</td><td>variable</td><td>2</td></tr>
<tr><td rowspan="2">Frame control</td><td rowspan="2">Sequence number</td><td>Destination PAN identifier</td><td>Destination address</td><td>Source PAN identifier</td><td>Source address</td><td rowspan="2">Frame payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="6">MHR</td><td>MAC payload</td><td>MFR</td></tr>
</table>

Figure 2.5: General MAC Frame Format [1]

- Frame control field comprises of two bytes. It has some subfields as can be seen in Figure 2.7. Frame type field specify the type of the frame (Table 2.3).

Bits: 0-2	**3**	**4**	**5**	**6**	**7-9**	**10-11**	**12-13**	**14-15**
Frame type	Security enabled	Frame pending	Ack. request	Intra-PAN	Reserved	Destination addressing mode	Reserved	Source addressing mode

Figure 2.6: Frame Control Field [1]

Table 2.3: Frame Type Values [1]

Frame Type Value $b_2\ b_1\ b_0$	**Description**
000	Beacon
001	Data
010	Acknowledgement
011	MAC Command
100-111	Reserved

- Sequence number field indicates unique sequence identifier for the frame
- PAN identifier and address fields for both destination and source specifies the unique PAN identifier and addresses of the transmitter and the receiver.

- Frame payload field includes some specific information for each frame types with a variable length.
- FCS field will be investigated with more detail in the next header. FCS field method is also used in implementation of transmitter before PHY.

2.2.1.1 FCS Field

For 2450 MHz, FCS field has 16 bits in length calculated as ITU-T CRC (International Telecommunications Union-Telecommunications Cyclic Redundancy Check). The FCS field can be calculated with following steps [1]:

- There is a specific generator polynomial of degree 16 to calculate FCS:

$$G_{16}(x) = x^{16} + x^{12} + x^5 + 1 \quad (2.6)$$

- Before adding FCS field, remaining part can be represented with another polynomial M(x) where k is the length of frame:

$$M(x) = b_0 x^{k-1} + b_1 x^{k-2} + \cdots + b_{k-2} x + b_{k-1} \quad (2.7)$$

- M(x) will be multiplied by x^{16}.
- $M(x)*x^{16}$ will be divided to the generator polynomial G(x). The coefficients of the remainder polynomial will be FCS field:

$$R(x) = r_0 x^{15} + r_1 x^{14} + \cdots + r_{14} x + r_{15} \quad (2.8)$$

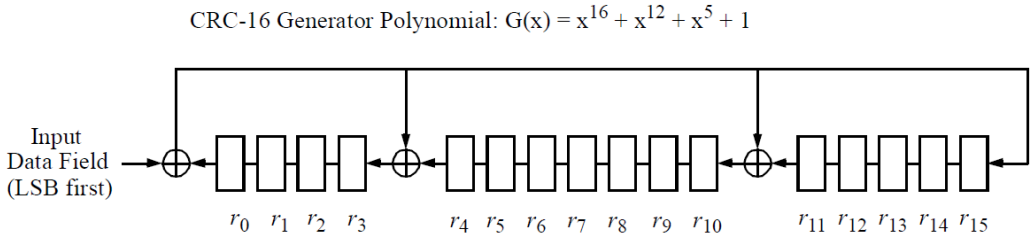
For instance, for an acknowledgement frame given 3 byte only MHR without payload:

0100 0000 0000 0000 0101 0110
b_0……b_{23}

The CRC will be in this case:

0010 0111 1001 1110
r_0……...r_{15}

The code for CRC block is implemented according to the following scheme in Figure 2.8.



1. Initialize the remainder register (r_0 through r_{15}) to zero.
2. Shift MHR and payload into the divider in the order of transmission (LSB first).
3. After the last bit of the data field is shifted into the divider, the remainder register contains the FCS.
4. The FCS is appended to the data field so that r_0 is transmitted first.

Figure 2.7: FCS Implementation [1]

At the receiver side the received information is processed through CRC block to check whether the information is received with or without error. If remainder in the register of CRC block is all zero, then the data would be taken correctly. It is shown in following equation:

$$\frac{M(x)x^{16}+R(x)}{G(x)} = \frac{M(x)x^{16}}{G(x)} + \frac{R(x)}{G(x)} = R(x) + R(x) = 0 \qquad (2.9)$$

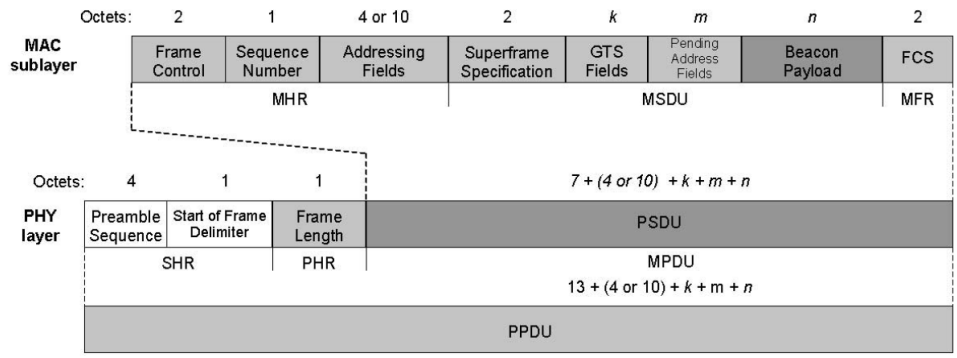
The last statement in the equation gives zero because of modulo 2 arithmetic.

2.3 Frame Structure

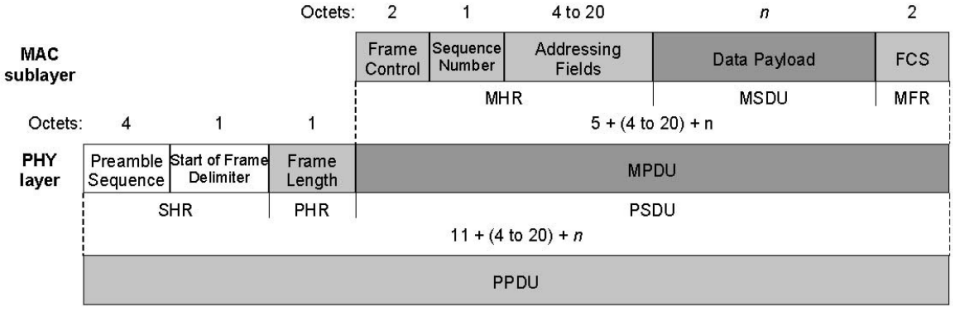
Minimum complexity and sufficient transmission on a noisy channel is achieved by creating a general frame structure. Each layer adds specific header and footer related to the layer to the structure. Four frame structures are defined in LR-WPAN:

- Beacon frame, to transmit beacons by coordinator
- Data frame, used for all types of data
- Acknowledgement frame, confirms the successful frame reception
- MAC command frame, handles all MAC peer entity control transfers

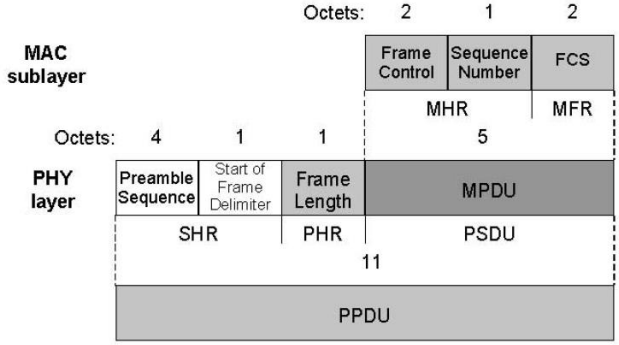
The structure for all frame types similar to each other. Each layer takes the SDU (Service Data Unit) from upper layer, adds header and footer specific to layer. By this way, PDU (Protocol Data Unit) is obtained. Finally, PHY transmits PPDU (PHY Protocol Data Unit) through the RF medium. All MAC and PHY data units can be seen in Figure 2.9.



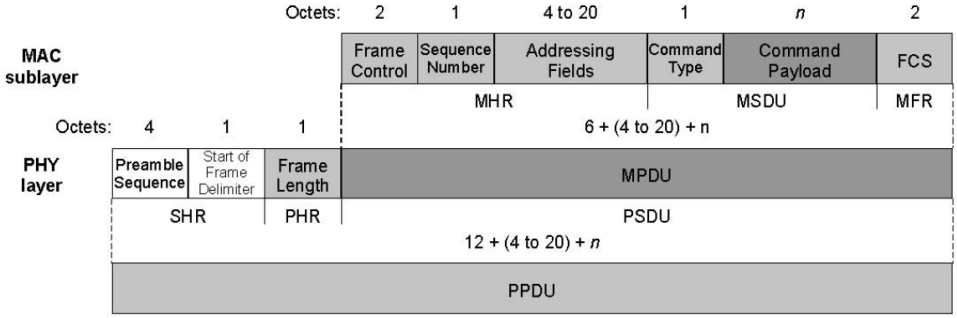
(a) Beacon Frame Structure



(b) Data Frame Structure



(c) Acknowledgement Frame Structure



(d) MAC Command Frame Structure

Figure 2.8: General Frame Structures [1]

3 DESIGN AND IMPLEMENTATION OF TRANSMITTER

The transmitter block has the following blocks: Cyclic Redundancy Check (CRC) block, Bit-to-Symbol block, Symbol-to-Chip block, Serial-to-Parallel block, Odd Bit Pulse-Shape block and Even Bit Pulse-Shape block [8]. The blocks are investigated separately and gathered after verification of each of them. Each block can work independently. The mission of transmitter starts with getting MSDU (MAC Service Data Unit). Firstly, it calculates Frame Check Sum (FCS) and then appends the value to the service data unit as footer. Then, four bits are grouped and formed the symbols. Symbols are represented with Pseudo-Random Noise (PN) sequences. Chips are modulated by using O-QPSK modulation technique with the Serial-to-Parallel block. Half sine pulse-shape is implemented in the last specific blocks.

3.1 CRC Block

Table 3.1: I/O Description of CRC Block

Pin List	Type	Size	Description
clk	in	1	250 kHz clock
rst	in	1	Synchronous reset
en	in	1	Active high trigger signal to operate CRC block
data_in	in	1	Input data
data_out	out	1	Output data

The transmitter module starts with CRC block. As we mentioned in previous sections, CRC block takes the data, do XOR operations at some points and shifts the data to the output. After all the data shifted through the register inside of the module, the 16 bits value in the register is appended to the end of the data [3]. In the initial state, register value is all 0. The register should be left as all 0 at the end of the operation.

In our case, we send a data with length of 24 bits in 250 kHz. As seen in first simulation, we keep active the CRC block during 96 μs (input duration) (Figure 3.2.a). The input comes first to the block on the 32 μs. At the same time enable signal is driven to binary 1 and CRC operation has started. For the idle case enable signal is kept as binary 0. The output data comes out first with a 1 clock cycle delay at the 36 μs. Final output of CRC block with 160 μs duration ends up on 196 μs. The I/O diagram is shown in Figure 3.1.

$$(24\ bits\ data + 16\ bits\ CRC) * 4\mu s = 160\ \mu s) \quad (3.1)$$

The second simulation shows us the check of the correction of the received packet at the receiver side (Figure 3.2.b). All bits are shifted through the CRC block at the receiver and then the decision is made according to the register state. If all registers contain 0, then packet is received correctly. Else, the receiver should send an acknowledgement frame to the transmitter about the error in the received packet.

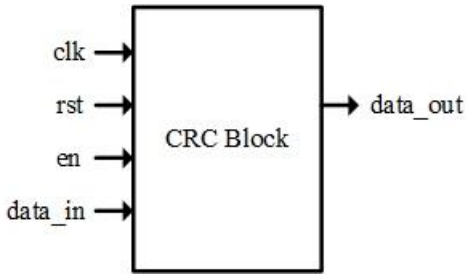
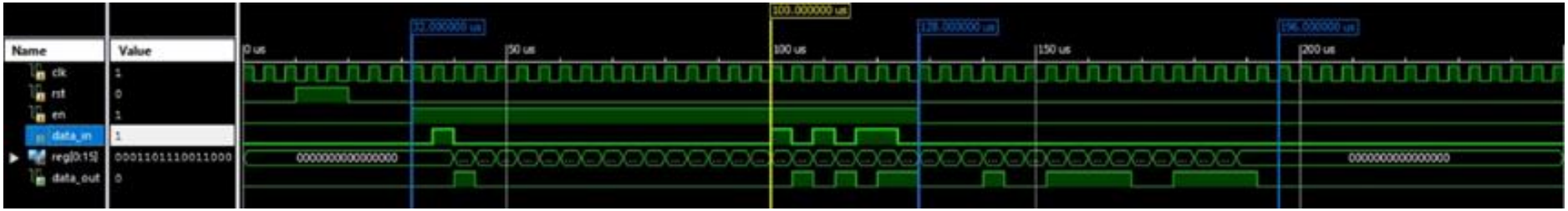
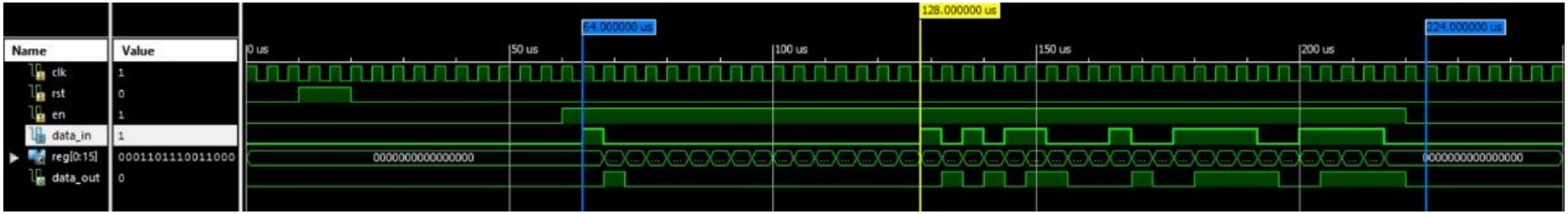


Figure 3.1: CRC Block I/O Diagram


Name
Value
clk
rst
en
data_in
reg[0:15]
data_out
0001101110011000
0 us
50 us
100 us
150 us
200 us


Name
Value
clk
rst
en
data_in
reg[0:15]
data_out
0001101110011000
0 us
50 us
64.000000 us
100 us
128.000000 us
150 us
200 us
224.000000 us

3.2 Bit-to-Symbol Block

Table 3.2: I/O Description of Bit-to-Symbol Block

Pin List	Type	Size	Description
clk	in	1	250 kHz clock
rst	in	1	Synchronous reset
data_in	in	1	Input data
data_out	out	4	Output data vector

The data comes to Bit-to-Symbol block after CRC operation. The enable input isn't needed in this block. There is a register with a 4 bits length at the output of the block. Block takes the data and shifts it in the register and gives the value in the register on every clock cycle [4]. The next block has to decide which inputs will be taken.

In our case, the output of CRC block has arrived on 36 μs, but the meaningful output of this block will be ready after shifting 4 bits on 52 μs (Figure 3.5). The I/O diagram is shown in Figure 3.3.

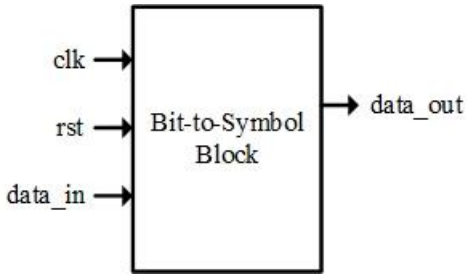


Figure 3.3: Bit-to-Symbol I/O Diagram

3.3 Symbol-to-Chip Block

Table 3.3: I/O Description of Symbol-to-Chip Block

Pin List	Type	Size	Description
clk	in	1	2 MHz clock
rst	in	1	Synchronous reset
en	in	1	Active high trigger signal to operate Symbol-to-Chip block
data_in	in	4	Input data vector
data_out	out	1	Output data

Symbols are coming to this block works with 2 MHz clock frequency from Bit-to-Symbol block. Symbol-to-Chip block has an array of PN sequences which are mentioned in the previous chapters. So, it gives the sequence to the register according to the value in the input. The block shifts the data in the register during 16 μs, then new data arrives and the block changes the register value and does the same shift operation for the new sequence [4].

In our case, the data comes to this block between 52 μs and 212 μs. The enable signal starts the operation with the incoming data. To be able to operate Symbol-to-Chip block needs enable signal at the right time With a 2 clock cycles delay, the sequences are given to output. So, next block will get the meaningful data on 53 μs and till 213 μs (Figure 3.6). The I/O diagram is shown in Figure 3.4.

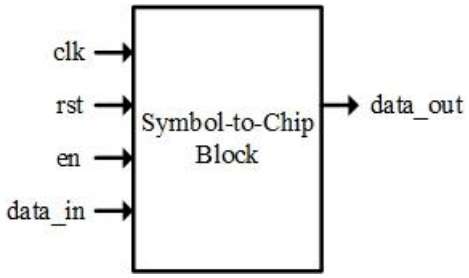
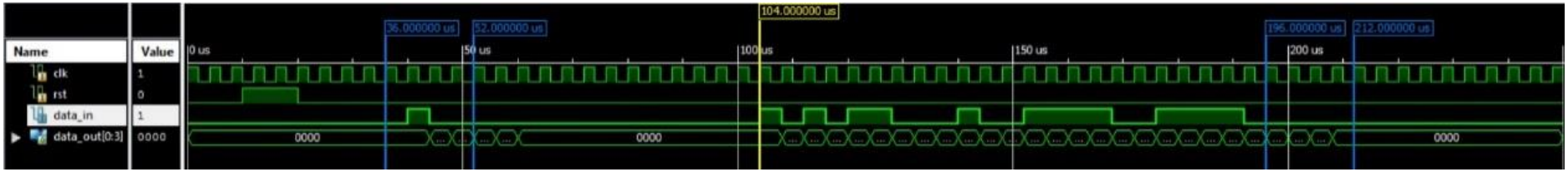
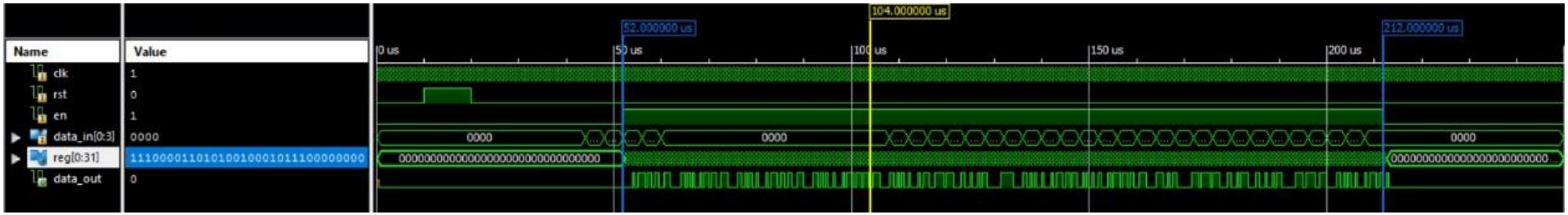


Figure 3.4: Symbol-to-Chip Block I/O Diagram


Name
Value
clk
rst
data_in
data_out[0:3]
1
0
1
0000
0 us
50 us
100 us
150 us
200 us
36.000000 us
52.000000 us
104.000000 us
196.000000 us
212.000000 us
0000
0000
0000


Name
Value
clk
rst
en
data_in[0:3]
reg[0:31]
data_out
1
0
1
0000
11100001101010010001011100000000
0
0 us
50 us
100 us
150 us
200 us
52.000000 us
104.000000 us
212.000000 us
0000
0000
0000
00000000000000000000000000000000
00000000000000000000000000

3.4 Serial-to-Parallel Block

Table 3.4: I/O Description of Serial-to-Parallel Block

Pin List	Type	Size	Description
clk	in	1	2 MHz clock
rst	in	1	Synchronous reset
en	in	1	Active high trigger signal to operate Serial-to-Parallel block
data_in	in	1	Input data
odd_bit	out	1	Odd bit output
even_bit	out	1	Even bit output

The chips with a 2Mchip/s rate are driven to the input of Serial-to-Parallel block. This module works like a demultiplexer. It separates the odd indexed bits and even indexed bits into the different channels which affects the transmission of data in the wireless medium positively. This process is named as I-Q channel separation. O-QPSK modulation is implemented in this block.

As mentioned in previous chapters, a delay with T_C (chip duration) duration, 0.5 μs, should be applied to bits with even index. Also, the duration of the each output bit is now two times T_C. Odd bits are given to output after 1 clock cycle delay. The delay for even bits is 2 clock cycles. Block starts to operate with the enable signal can be seen in Figure 3.10. The I/O diagram is shown in Figure 3.7.

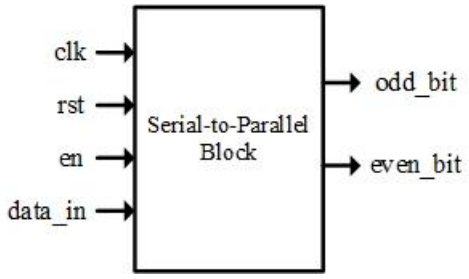


Figure 3.7: Serial-to-Parallel Block I/O Diagram

3.5 Odd Bit Pulse Shape Block

Table 3.5: I/O Description of Odd Bit Pulse Shape Block

Pin List	Type	Size	Description
clk	in	1	10 MHz clock
rst	in	1	Synchronous reset
en	in	1	Active high trigger signal to operate Odd Bit Pulse Shape block
data_in	in	1	Input data
odd_samples	out	8	Output data vector

Pulse shaping again increases the efficiency of the transmission and the modulation with the 2.4 GHz carrier like Serial-to-Parallel block. The working clock frequency depends on how many samples will be taken from the data. Our case, the duration of each bit is 1 μs and 10 samples have been gotten from each bit [5]. If more samples are needed, the higher clock frequency should be used.

Two different constant arrays, one for bit one and one for bit zero, are initialized before synthesis. They include the sampling values for a sine wave. If the block gets the data '1', it gives the values in the array for bit one to the output and do the same thing for the bit zero. Simulation waveform can be seen in Figure 3.11.

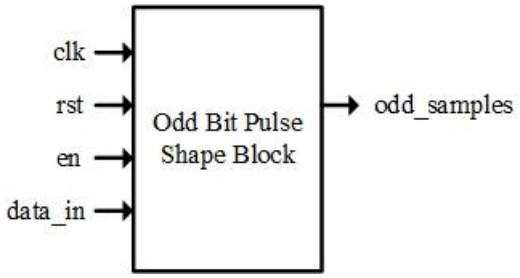


Figure 3.8: Odd Bit Pulse Shape Block I/O Diagram

3.6 Even Bit Pulse Shape Block

Table 3.6: I/O Description of Even Bit Pulse Shape Block

Pin List	Type	Size	Description
clk	in	1	10 MHz clock
rst	in	1	Synchronous reset
en	in	1	Active high trigger signal to operate Even Bit Pulse Shape block
data_in	in	1	Input data
even_samples	out	8	Output data vector

It does the same operation and has the same properties like Odd Bit Pulse Shape block. The only difference is that input comes to this block 0.5 μs later than the other block due to the delay of Q channel in O-QPSK modulation.

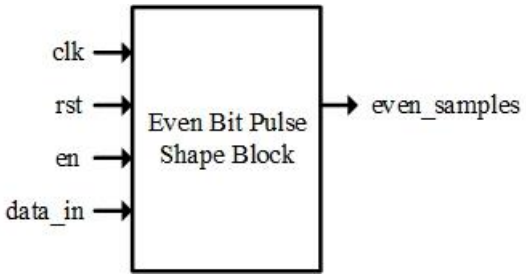
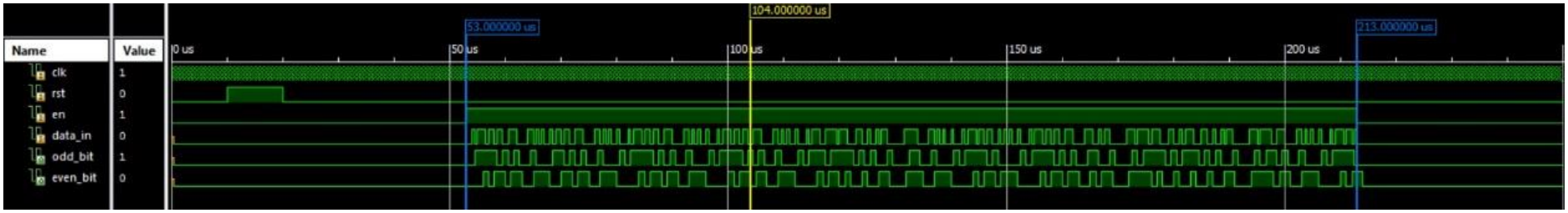
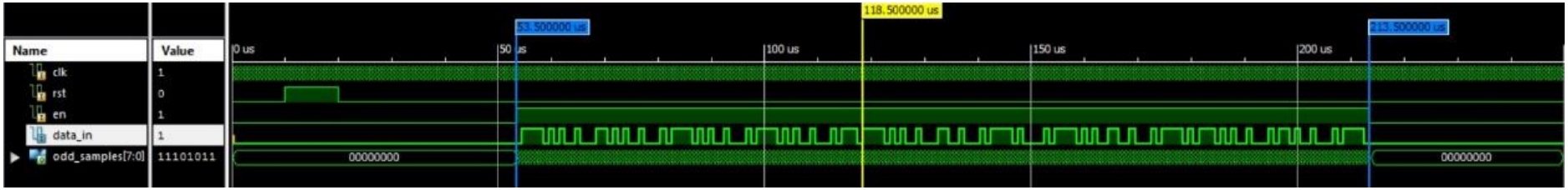
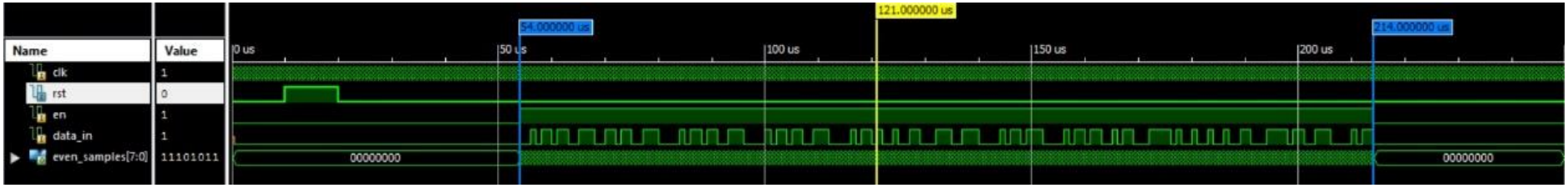


Figure 3.9: Even Bit Pulse Shape Bloc I/O Diagram


104.000000 us
53.000000 us
213.000000 us
Name
Value
0 us
50 us
100 us
150 us
200 us
clk
1
rst
0
en
1
data_in
0
odd_bit
1
even_bit
0


118.500000 us
53.500000 us
213.500000 us
Name
Value
0 us
50 us
100 us
150 us
200 us
clk
1
rst
0
en
1
data_in
1
odd_samples[7:0]
11101011
00000000
00000000


121.000000 us
54.000000 us
214.000000 us
Name
Value
0 us
50 us
100 us
150 us
200 us
clk
1
rst
0
en
1
data_in
1
even_samples[7:0]
11101011
00000000
00000000

4 RADIO FREQUENCY MEDIUM

Digital processes are realized in FPGA. Digital-to-Analog and Analog-to-Digital converters are needed for conversion between these two domains. Also IEEE 812.15.4 standard requires operating in 2.4 GHz frequency band. So we need multiply our signal with a carrier with 2.4 GHz frequency. At the receiver, the signal in carrier frequency should be filtered with a Low-pass filter, sampled and converted to digital signal again [8]. All these processes are implemented by using MATLAB Simulink tool.

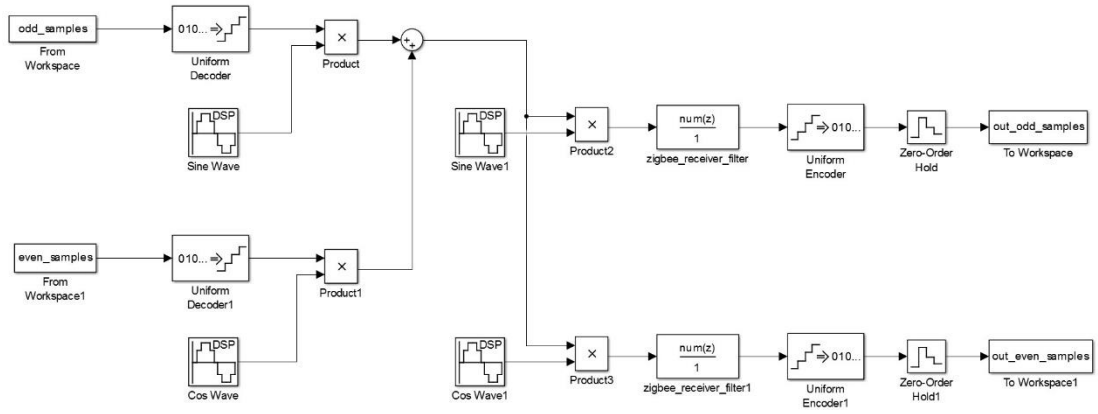


Figure 4.1: RF Medium Simulink Model

At the end of the transmitter, Pulse Shape blocks are sampling the incoming data with 1 μs period at a rate of 10 Msample/sec. So 10 8-bit samples are collected per one period. Those values are written to txt file by using "textio" library. These txt files for odd samples and even samples are given to the Simulink model as input.

Digital-to-Analog converter is implemented with Uniform Decoder model. The samples are taken as signed binary vectors. Uniform Decoder takes integers as inputs, gives floating-point output between peak values given to the block before run [9]. So in a MATLAB script signed samples are converted to signed integers. Peak value is assigned in the Uniform Decoder block as one. Outputs are obtained as can be seen in the Figure 4.2. First and third graphs shows the inputs. Even samples are coming with a delay of 0.5 μs in third graph. Second and fourth graphs shows the related analog signals whose values are changing between 1 and -1.

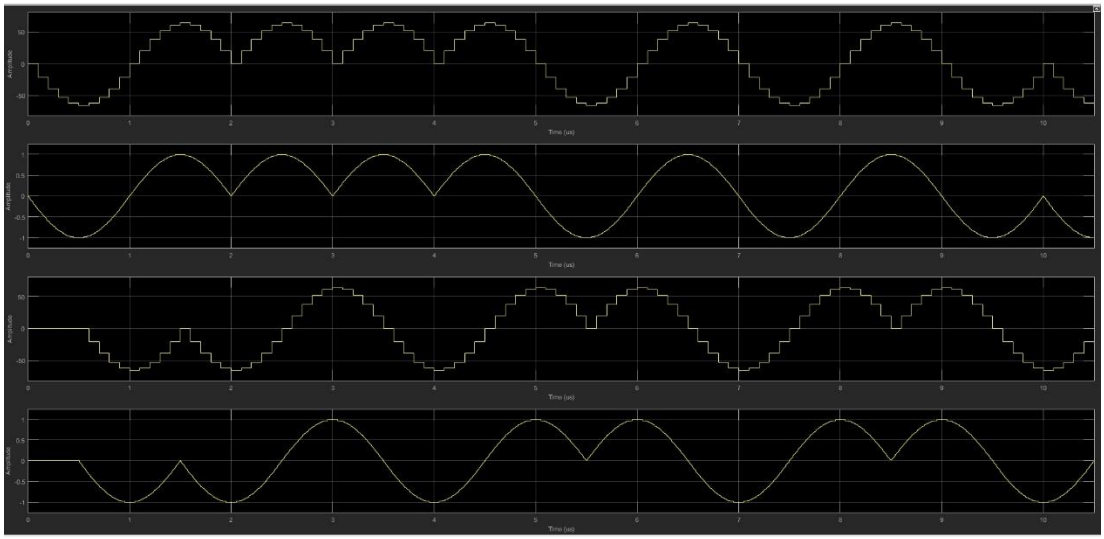


Figure 4.2: Sampled Inputs and Related Outputs

Half-sine pulse shaped and sampled digital signal is converted to analog signal. The period for a half-sine shaped analog signal is 1 μs. To transmit the data, it should be carried by sine waves with 2.4 GHz frequency. The modulation to 2.4 GHz frequency is implemented by multiplying analog converted signal with a sine wave (Figure 4.3). The upper graph shows the analog signal. The modulated signal with the envelope of data can be observed in the lower graph.

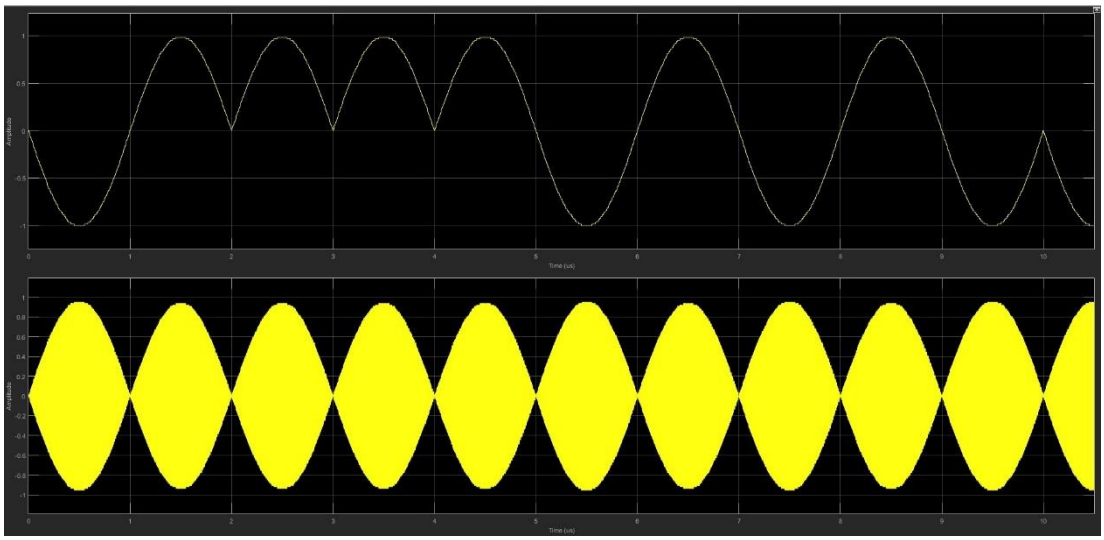


Figure 4.3: Analog Signal and Modulated Signal

Both of I and Q channels are converted to the analog signal. Then they multiplied with 2.4 GHz sine and cosine waves in order. The transmitter will add the two modulated signal and then the signal will be transmitted to the receiver. Modulated signals and sum of them can be seen in Figure 4.4.

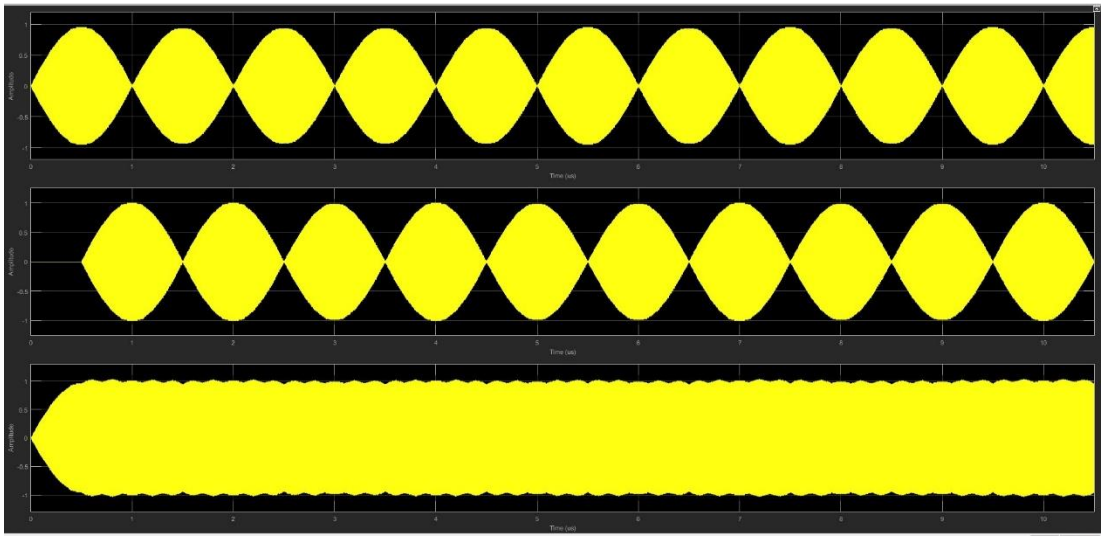
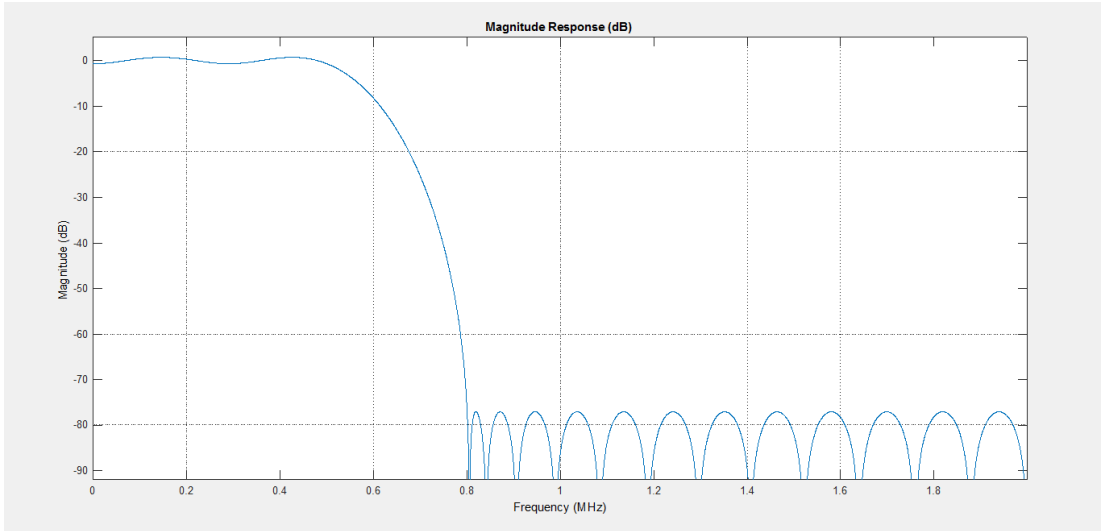
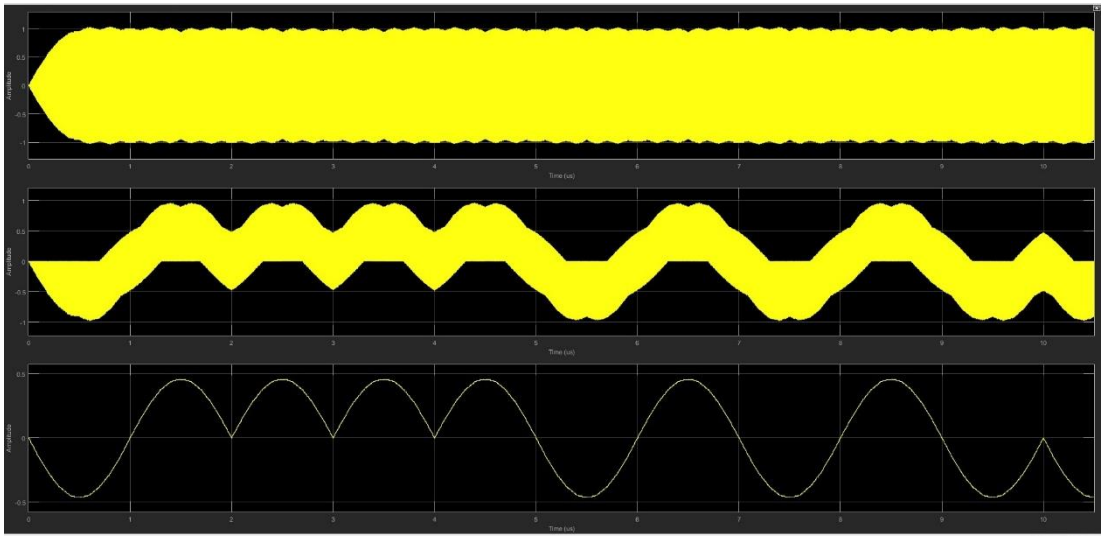


Figure 4.4: Modulated Signals and Sum of Them

The RF part of the receiver will receive the transmitted signal. Coming signal is in 2.4 GHz frequency band. To get the data from the modulated signal it should be down-converted to the lower frequency. This is realized by multiplying the signal again with the sine and cosine wave in 2.4 GHz frequency separately. By multiplying, I and Q channels are obtained. But the signals have higher frequency components. To eliminate the high frequency components it should be filtered with a proper low-pass filter. The magnitude response of the low-pass filter and the filtered signal of I channel can be seen in Figure 4.5.



(a) The Magnitude Response of Low-Pass Filter



(b) Down-Converted and Filtered I Channel Signal

Figure 4.5: Low-Pass Filter and Filtered Signal

The duration of a half-sine shaped pulse is 1 μs. So the signal may be treated as sine wave in 500 kHz frequency. Cut-off frequency of low-pass filter is chosen as 500 kHz. It attenuates the components after the cut-off frequency. Finally, filtered signal will be sampled and converted to the digital signal. Analog-Digital Converted is implemented by using Uniform Encoder block in Simulink. The block takes the analog values and converts them into the samples according to given quantization level. In this project one sample is represented with 8 bits.

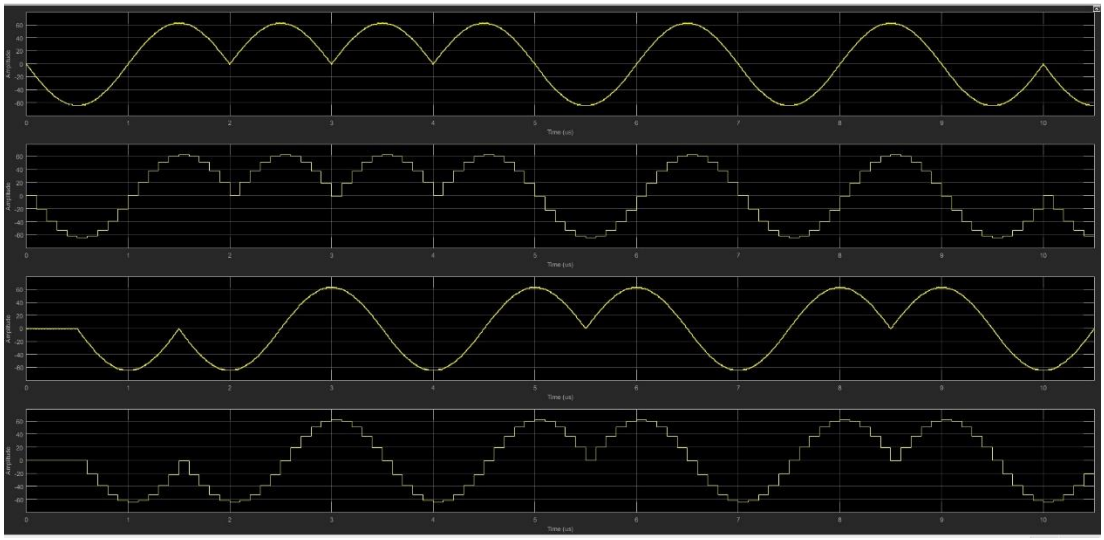


Figure 4.6: Analog-Digital Converter Inputs and Outputs

The sample values are exported to a txt file at the end of simulation in Simulink. These txt files are given to the receiver module as input.

5 DESIGN AND IMPLEMENTATION OF RECEIVER

The output values of transmitter are written to a txt file with predefined functions of VHDL. Those txt files are given to implemented Simulink model to make the RF transmission. Again the outputs are collected in txt files of odd samples and even samples. Those txt files will be the input for the receiver. Input values are read in the test bench of the receiver. Receiver includes following blocks: Odd Bit Pulse Shape Detect block, Even Bit Pulse Shape Detect block, Assist blocks, Parallel-to-Serial block, Chip-to-Symbol block, Symbol-to-Bit block and CRC block [6]. Pulse shape detect blocks are detecting the samples coming from RF part of the receiver. Assist blocks are used to regulate the timing difference due to blocks which work with different clock frequencies. I and Q channels are then serialized. The symbols are estimated according to coming bits. The highest number of match principle is used for estimation. Finally, the coming data is checked by CRC block whether there is an error or not.

5.1 Odd Bit Pulse Shape Detect Block

Table 5.1: I/O Description of Odd Bit Pulse Shape Detect Block

Pin List	**Type**	**Size**	**Description**
clk	in	1	10 MHz clock
rst	in	1	Synchronous reset
en	in	1	Active high trigger signal to operate Odd Bit Pulse Shape Detect block
data_in	in	8	Input data vector
odd_bit	out	1	Output data

The data given to this block is the samples generated by analog-digital converter. Odd Bit Pulse Shape Detect block has received the same number of samples, ten, as the pulse shaping blocks have transmitted.

The block assumes that if the sample value is greater than zero, it should be taken from a sine wave represents the bit one. It assumes the vice versa case for the bit zero. There may be some changes in the data during the transmission in the air or through the RF transceivers. So, it counts the positive and negative valued samples separately. It decides which bit will be given to the output according to the bigger counter.

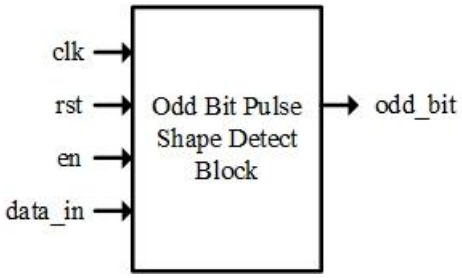


Figure 5.1: Odd Bit Pulse Shape Detect Block I/O Diagram

5.2 Even Bit Pulse Shape Detect Block

Table 5.2: I/O Description of Even Bit Pulse Shape Detect Block

Pin List	Type	Size	Description
clk	in	1	10 MHz clock
rst	in	1	Synchronous reset
en	in	1	Active high trigger signal to operate Even Bit Pulse Shape Detect block
data_in	in	8	Input data vector
even_bit	out	1	Output data

This block does the same operations and follows the same principles like the Odd Bit Pulse Shape Detect Block. It estimates the data from the samples and drives the output during 1 μs for each data.

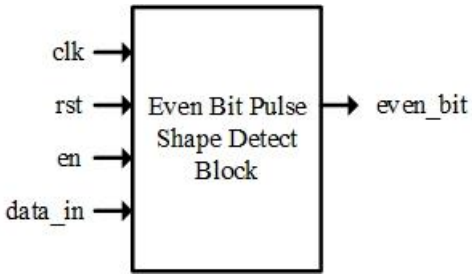
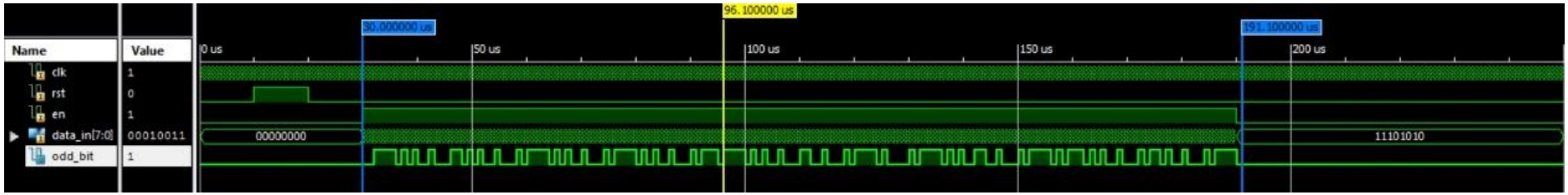
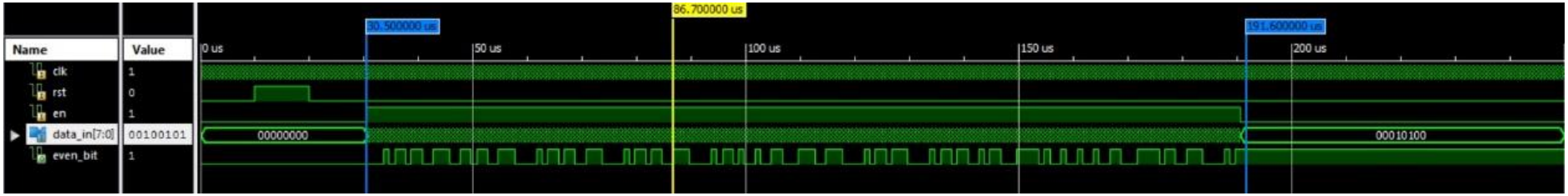


Figure 5.2: Even Bit Pulse Shape Detect Block I/O Diagram


96.100000 us
30.000000 us
191.100000 us
Name
Value
0 us
50 us
100 us
150 us
200 us
clk
1
rst
0
en
1
data_in[7:0]
00010011
00000000
11101010
odd_bit
1


86.700000 us
30.500000 us
191.600000 us
Name
Value
0 us
50 us
100 us
150 us
200 us
clk
1
rst
0
en
1
data_in[7:0]
00100101
00000000
00010100
even_bit
1

5.3 Assist Block

Table 5.3: I/O Description of Assist Block

Pin List	Type	Size	Description
clk	in	1	2 MHz/250 kHz clock
rst	in	1	Synchronous reset
data_in	in	1/4	Input data/data vector
data_out	out	1/4	Output data/data vector

The reason to give Assist Block name to this module, it really assists to the main modules working with different clock frequencies. It takes its input from previous module and gives to the output according to the clock frequency of next module.

This block is used in two different places. First one recovers the clock frequency difference between Pulse Shape Detect Blocks working with 10 MHz frequency and Parallel-to-Serial Block working with 2 MHz frequency (Figure 5.3.b). The other one is between Chip-to-Symbol Block working with 2 MHz frequency and Symbol-to-Bit Block working with 250 kHz frequency (Figure 5.3.b). The following module gets the data after some delay but proper with its clock frequency.

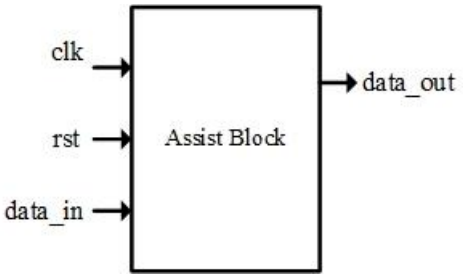
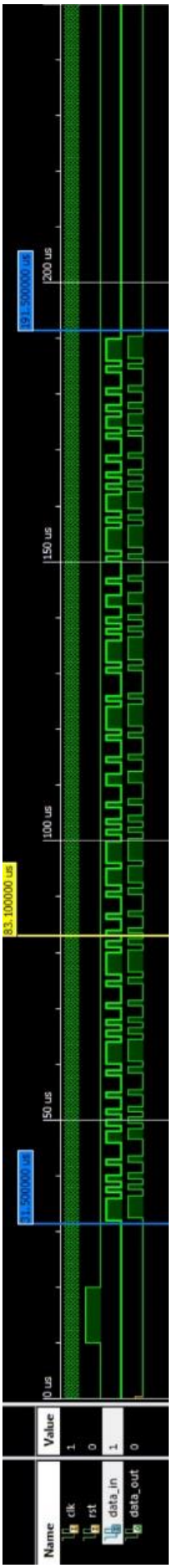
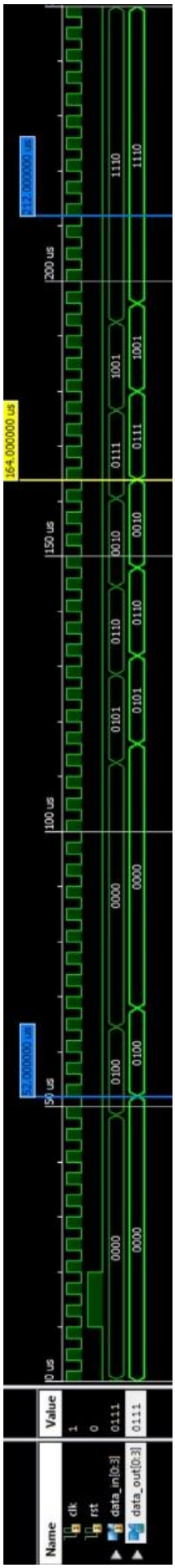


Figure 5.5: Assist Block I/O Diagram


Name
Value
clk
rst
data_in
data_out
0 us
50 us
100 us
150 us
200 us
83.100000 us
19.500000 us


Name
Value
clk
rst
data_in[0:3]
data_out[0:3]
0 us
50 us
100 us
150 us
200 us
164.000000 us
0000
0100
0101
0110
0010
0111
1001
1110

5.4 Parallel-to-Serial Block

Table 5.4: I/O Description of Parallel-to-Serial Block

Pin List	Type	Size	Description
clk	in	1	2 MHz clock
rst	in	1	Synchronous reset
en	in	1	Active high trigger signal to operate Parallel-to-Serial block
odd_bit	in	1	Input data detected from odd samples
even_bit	in	1	Input data detected from even samples
data_out	out	1	Output data

The bits estimated by Pulse Shape Detect Blocks are coming to Parallel-to-Serial Block after passing through Assist Block. The timing adjustment of the incoming bits by Assist Block is needed for Parallel-to-Serial Block to process the data properly.

Even indexed bits are coming to this block one clock cycle later than odd ones. The module firstly gives the odd bit to the output, then gives the even bit. This process is done until all the bits are given to the output. The duration of each input is reduced to 0.5 μs from 1 μs.

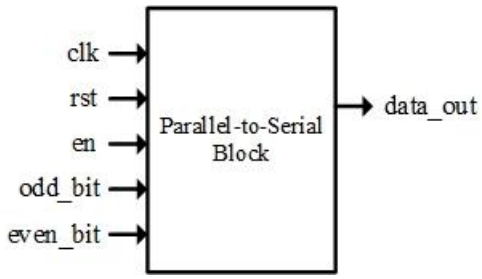


Figure 5.7: Parallel-to-Serial Block I/O Diagram

5.5 Chip-to-Symbol Block

Table 5.5: I/O Description of Chip-to-Symbol Block

Pin List	Type	Size	Description
clk	in	1	2 MHz clock
rst	in	1	Synchronous reset
en	in	1	Active high trigger signal to operate Chip-to-Symbol block
data_in	in	1	Input data
data_out	out	4	Output data vector

This block is implemented to estimate the symbol represents four bits from coming chips. At the transmitter, the symbols are converted to the chip sequences. Those chip sequences are parallelized, O-QPSK modulated and pulse shaped. During transmission in the wireless medium some errors may occur. Pulse Shape Detect Blocks may be

wrong while estimating the bits from samples. So, a reliable and proper method is needed.

In this module, an array consists of chip sequences, an array consists of symbols and an array to keep the number of matches are generated. In every clock cycle, coming bits are being compared with the related chips in the chip sequences in for loop. If a match happens, the number in that index is increased by one. Another for loop assigns the index of the highest number in the array consists of number of matches to the variable named “max”. After getting all 32 chips, the symbol in the index “max” of the array consists of symbols is given to the output.

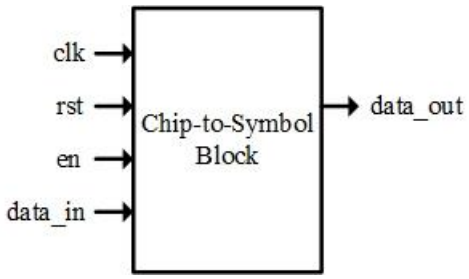


Figure 5.8: Chip-to-Symbol Block I/O Diagram

5.6 Symbol-to-Bit Block

Table 5.6: I/O Description of Symbol-to-Bit Block

Pin List	Type	Size	Description
clk	in	1	250 kHz clock
rst	in	1	Synchronous reset
en	in	1	Active high trigger signal to operate Symbol-to-Bit block
data_in	in	4	Input data vector
data_out	out	1	Output data

Symbol-to-Bit and Chip-to-Symbol blocks are working with clocks in different frequencies. So as mentioned under the Assist Block section, an Assist block has been used between these two modules. Symbol-to-Bit block gets its input synchronized with 250 kHz clock frequency. It keeps the symbol values in a register with 4 bits length. When it is enabled, it gives the least significant bit of the register to the output and shifts the register to the right. The output bits are given to CRC block as a last stage to check whether the data is taken without error.

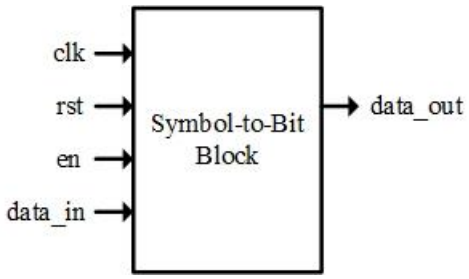
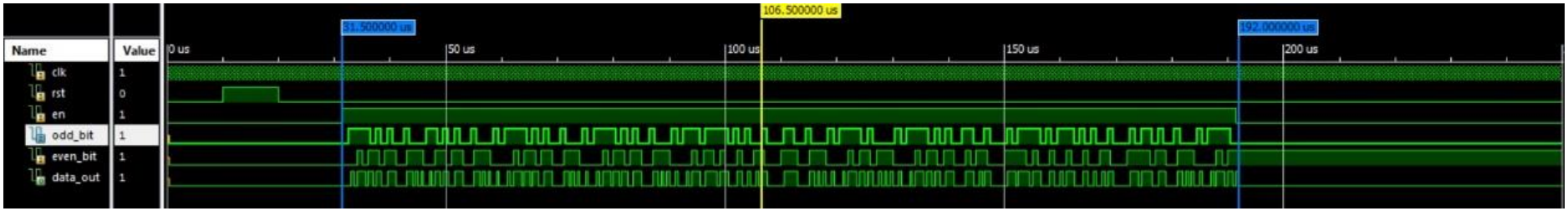
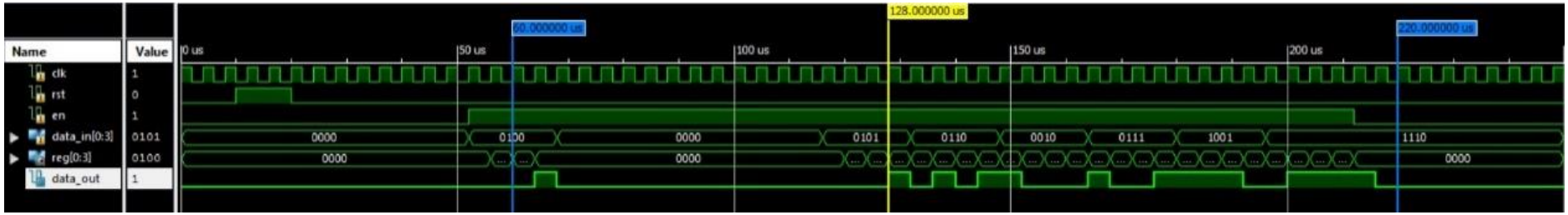


Figure 5.9: Symbol-to-Bit Block I/O Diagram


106.500000 us
31.500000 us
192.000000 us
Name
Value
0 us
50 us
100 us
150 us
200 us
clk
1
rst
0
en
1
odd_bit
1
even_bit
1
data_out
1


112.500000 us
32.000000 us
48.500000 us
192.000000 us
208.500000 us
Name
Value
0 us
50 us
100 us
150 us
200 us
clk
1
rst
0
en
1
data_in
1
data_out[0:3]
0101
0000
0100
0000
0101
0110
0010
0111
1001
1110


128.000000 us
60.000000 us
220.000000 us
Name
Value
0 us
50 us
100 us
150 us
200 us
clk
1
rst
0
en
1
data_in[0:3]
0101
reg[0:3]
0100
data_out
1
0000
0100
0000
0101
0110
0010
0111
1001
1110
0000
0000
0000

CONCLUSION

This thesis and work are implemented by using VHDL (Very High Integrated Circuit Hardware Description Language) and Vivado version 2016.2 tool. Each block has been designed, tested and verified separately, so they can be used for another projects or purposes. Final work was to compose them and make a test bench to verify whether they function correctly or not. Simulations are investigated deeply. Different cases are tested. The result was successful that implemented transmitter and receiver both are working properly.

As a future work, they can be implemented on the chip. There may be some little timing errors on FPGA. In simulation environment everything goes well. It has been seen that it is more practical and useful to have a wireless standard by using FPGA resources. There is no need for extra devices. The features of block can be modified according to usage. More complicated circuits which depends on wireless communication can be designed and implemented by using Physical Layer implementation on the thesis.

REFERENCES

[1] **IEEE Computer Society,** 2003. IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE, New York.

[2] **Kanna, R.,** 2011. Design of ZigBee Transceiver for IEEE 802.15.4 Using MATLAB/Simulink, *MSc Thesis*, National Institute of Technology Rourkela, India.

[3] **Ahmad, R., Sidek, O. and Mohd, S.K.K.,** Development of the CRC Block for ZigBee Standard on FPGA, Malaysia.

[4] **Ahmad, R., Sidek, O. and Mohd, S.K.K.,** 2009. Development of Bit-to-Chip Block for ZigBee Transmitter on FPGA, *Second International Conference on Computer and Electrical Engineering,* 492-496.

[5] **Ahmad, R., Sidek, O. and Mohd, S.K.K.,** 2011. Implementation of IEEE 802.15.4 Based O-QPSK Pulse-Shaping Block on FPGA, *International Conference on Computer Applications and Industrial Electronics,* 459-464.

[6] **Ahmad, R., Sidek, O. and Mohd, S.K.K.,** 2014. Implementation of a Verilog Based Digital Receiver for 2.4 GHz ZigBee Applications on FPGA, *Journal of Engineering Science and Technology,* Vol. 9, No. 1, 136-153.

[7] **Lee, K.T.,** 2004.Designing a ZigBee-ready IEEE 802.15.4-compliant radio transceiver, *Next Generation Wireless,* 42-50.

[8] **Muni, B.K.,** 2013. Physical Layer Implementation of a class of ZigBee Baseband Transceiver using FPGA, *MSc Thesis,* National Institute of Technology Rourkela, India.

[9] **MATLAB website**, https://www.mathworks.com/products/matlab.html

RESUME

Name Surname: Efe Emre Pazarlı

Place and Date of Birth: Lüleburgaz, 1992

High School: Kabataş Erkek Lisesi, Istanbul (2006-2011)

BSc: Istanbul Technical University, Electronics and Communication Engineering (2011-2017)