NTNU
Norwegian University of
Science and Technology

Design of Digital Baseband Transceiver for Touchscreen Devices

Ghaith Alhabbal Almujalled

Embedded Computing Systems
Submission date: June 2016
Supervisor: Lars Magne Lundheim, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

NTNU – Trondheim
Norwegian University of
Science and Technology

Design of Digital Baseband Transceiver for Touchscreen Devices

Ghaith Alhabbal Almujalled

June 2016

PROJECT / MASTER THESIS

Department of Electronics and Telecommunications

Norwegian University of Science and Technology

Supervisor 1: Professor Lars Lundheim (NTNU)

Supervisor 2: Mr. Odd Reitam (Atmel Norway)

Preface

This Master's thesis was written at the Norwegian University of Science and Technology (NTNU), under the department of Electronics and Telecommunications. It signifies the academic pinnacle of my time at higher education, and the final barrier before earning a Master of Science degree in Embedded Systems.

I have kept in mind three types of readers while writing this thesis. The first reader is an engineering student who is interested in extending and research further the work done on the thesis topic. The second reader is the academics from the assessment committee. The third reader is an electronics engineering student who is interested in transceivers architectures and digital desgin.

Trondheim, May, 2016

Acknowledgment

I would like to thank my supervisor at NTNU, Professor Lars Lundheim, and my external supervisor at Atmel, Odd Magne, for always being available when required and for providing feedback during the course of this thesis. I would like also to thank my friends here in Trondheimn Norway and in Southampton, England for their support.

Last but not the least, I would like to thank my parents for their unlimited support and encouragement of pursuing higher education.

Ghaith Alhabbal Almujalled

Abstract

With recent advances in low power electronics combined increasingly smaller integrated circuits, battery-driven smart devices are becoming more prevalent. The thesis motivation is proposed by Atmel, Norway to develop a digital baseband transceiver architecture which can be implemented to handle digital communications between an active stylus and a hand-held, touch screen devices.

This thesis presents a baseband digital transceiver architecture including transmitter and receiver. the focus of the design is on the receiver architecture. The transceiver design exploited a one bit architecture. The baseband modulation implemented direct sequence spread-spectrum communication modulation which proved effectiveness in simulations. The hardware implantation was done in simple hardware architecture using hardware description language SystemVerilog.

Contents

Chapter 1

Introduction and Motivation

1.1 Background

A capacitance touchscreen represents a mainstream technology for mobile phones today. It enables precise location sensing based on a stylus or finger contact on the screen. A key issue to be considered when designing a touchscreen system is the influence of electromagnetic interference (EMI) on the overall performance.

A main source of touchscreen interference is the switching activity from the phone charger. In basic terms, and according to a white paper published by Silicon Labs [1] interference is coupled through the finger to the touchscreen. These interference voltages cause charge movement within the touchscreen, which may be confused with the measured charge due to a stylus touch on the screen. Effective design and optimization of the touchscreen system depends on understanding communication channel properties between the active stylus and the touchscreen controller and mitigating or compensating for them as much as possible by designing a robust transceiver system.

According to the same paper, The interference waveform is complex and varies considerably between chargers, depending on circuit details and output voltage control strategy. Interference amplitude varies considerably depending on how much design effort and unit cost the manufacturer has allocated to shielding in the switching transformer. According to experimental data from Atmel, typical parameters include:

- Wave shape: Complex; consisting of pulse-width modulation square wave followed by LC ringing.
- Frequency: Varies between 1 kHz and 1 MHz with a duty cycle of 43%.
- Voltage: up to 5 V.

This thesis aimed to investigate a transceiver architecture that can handles the communications between an active stylus and capacitance touchscreen. The challenge is to overcome the noisy communication channel represented by the capacitance touchscreen within the following objectives.

1.2 Objectives

The main objectives of this Master's project are

1. The transceiver design should focus on a low-powered architecture.
2. The transmitter should only utilize the square waves generated on chip, which limits the pulse shape to either square waves or filtered square waves.
3. The receiver should have a small foot-print to reduce manufacturing costs.
4. The receiver should be as low-powered as possible since the targeted application is hand-held and powered by batteries.
5. A balance between the cost and and the performance must be achieved by exploiting the available degrees of freedom.

1.3 Approach

The first step was to investigate a proper modulation scheme that can exploit any available degree of freedom. Once the modulation scheme was chosen, verifying its effectiveness was the next step. This has been done by building a system model of the proposed modulation scheme. Moreover, a model for the studied channel was needed to be set as well to simplify the design

problem. After verifying a working model, the thesis took the direction of implementing the proposed model in hardware. The hardware modules were build and verified for logical correctness via simulations. The real-time verification of the hardware modules took place by synthesizing the transmitter module and the receiver module on two individual FPGA boards.

1.4 Structure of the Report

The rest of the report is organized as follow. Chapter two gives an introduction to the chosen modulation scheme and the theory behind it. Chapter three studies the system model of the transceiver and the communication channel. Chapter four details the hardware implementation of the transceiver based on the models discussed in Chapter three. Chapter five lists simulation results and the hardware verification done to verify the effectiveness of the proposed design. Lasly, Chapter six discusses the results from the Chapter five and gives possible future work approaches.

Chapter 2

Background Material

Some materials are presented here to give an insight into the targeted application of the proposed transceiver. The theoretical basis of direct-sequence spread spectrum are the fundamental building blocks for the proposed transceiver. Therefore, a brief theory regarding spread spectrum modulation is presented in this chapter before jumping in the system analysis and specification in the following chapter.

2.1 Choosing the Modulation Scheme

The communication channel is quite challenging due to the severe levels of interference. Moreover, it has an important property which is the wide frequency spectrum, ranging from 1 kHz to 1 MHz. Therefore, using this property to counter the interference is an advantage must be make use of when designing the transceiver. Other modulation techniques based on keying such as PSK, FSK, ASK, and QAM are also possible to explore. However, for this thesis, developing a spread-spectrum system is an attractive choice due to the wide frequency spectrum the channel allows since it allows the spreading of the data signal over a wide frequency spectrum.

2.2 Spread Spectrum Modulation

Spread spectrum modulation is a modulation method applied to digitally modulated signals that increases the transmit signal bandwidth to a value much higher than is needed to trans-

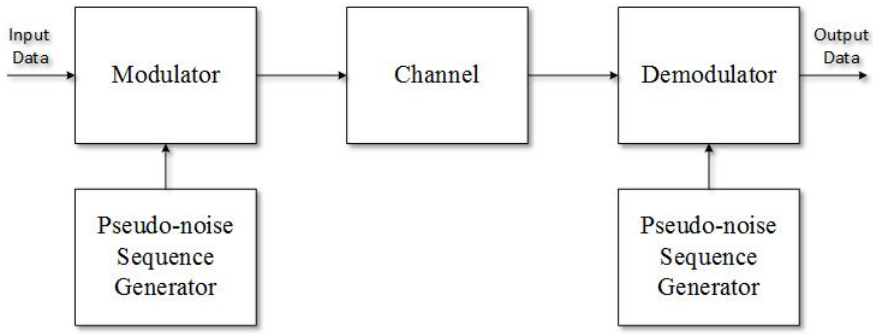


Figure 2.1: Spread Spectrum System Model.

mit the underlying information bits. There are various signaling techniques that increase the transmit bandwidth above the minimum required for data transmission such as coding and frequency modulation but they are not considered as spreading techniques. To call a signal a spread-spectrum modulated, the following conditions must be met:

1. The modulated signal occupies a bandwidth much larger than the one needed for the information bits. That is the bandwidth expansion factor $B = W/R$ is much greater than unity.

2. The spread-spectrum modulation is done using a spreading code, which must be independent of the data in the signal.

3. Dispreading at the receiver is done by correlating the received signal with a synchronized copy of the spreading code.

The block diagram shown in Figure 2.1 illustrates the basic elements of a spread-spectrum digital communications systems. It is different from other conventional digital communication systems by the inclusion of two identical pseudo-random pattern generators. One that interfaces with the modulator at the transmitting end and the second that interfaces with the demodulator at the receiving end.

The large redundancy inherited in the spread spectrum signal is essential when it comes to overcome the noise added to a channel transmitting digital information. The most important element in a spread spectrum is the pseudo randomness which makes the signals appear as noise to other unintended systems. This is achieved by using a pseudo-noise sequence (PN) which is a

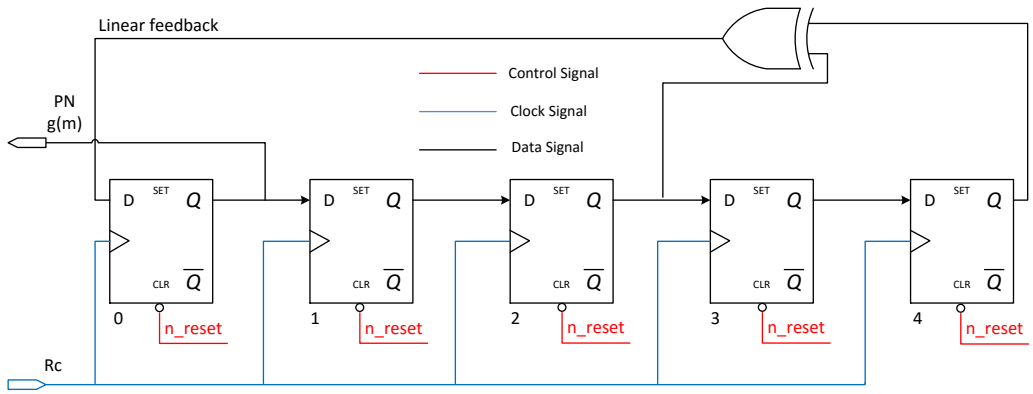


Figure 2.2: Linear-feedback shift register.

sequence of chips $C_0, C_1, C_2, \cdots, C_n$ with an autocorrelation function $R_c(m) = \sum_n C_n C_{n-m}$ largely considerate around the origin.

A common procedure for generating PN sequences is to use maximum-length sequence generators. A common hardware used to generate such sequences is a linear-feedback shift register (LFSR) illustrated in Figure 2.2.

An LFSR is a shift register whose input bit is a linear function of its previous state. The most commonly used linear function of single bits is exclusive-or (XOR). Thus, an LFSR is most often a shift register whose input bit is driven by the XOR of some bits of the overall shift register value.

2.2.1 Direct Sequence Spread-Spectrum

In direct-sequence spread-spectrum modulation, the data signal $s(t)$ is multiplied by a wide-band spreading code $g(t)$, which is constant over a time duration T_c, and has amplitude equal to 1 or -1 as illustrated in figure 2.3. The bandwidth $Bc \approx 1/T_c$ of $g(t)$ is $B_c/B = T_s/T_c$ times larger than the bandwidth B of the modulated signal $g(t)$, and the number of chips per bit, i.e. T_s/T_c , is an integer equal to G_p namely the processing gain of the system. Therefore the data signal pulse period is $T_s = G_p \times T_c$. Multiplying the data signal by the spreading signal results in the convolution of these two signals in the frequency domain. Thus, the transmitted sign $s(t) \times g(t)$ has frequency response $S(f) * G(f)$ which has a bandwidth of $B_c + B$. At the receiver side, the received spread signal is $s(t) \times g(t) + w(t)$ where $w(t)$ is the noise signal. If the receiver

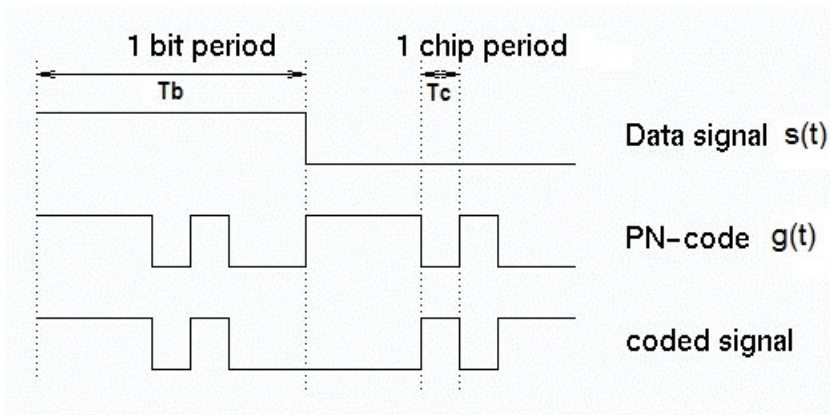


Figure 2.3: Bit period and chip period.

multiplies this signal by a synchronized replica of the spreading signal, the result is

$$s(t) \times g^2(t) + w(t) \times g(t) \tag{2.1}$$

which leads to

$$s(t) \times g^2(t) + w(t) \times g(t) = s(t) + w'(t) \tag{2.2}$$

Where $w'(t) = w(t) \times g(t)$ has the same statistics as $w(t)$ (zero mean). Therefore, the spreading and de-spreading have no impact on the transmitted signals.

2.2.2 Spectrum of Direct-Sequence Spread-spectrum

If we consider a standard binary, amplitude shift keying (ASK) where $A_n \in -1, 1$, therefore the constructed signal is

$$s(t) = \sum_{-\infty}^{+\infty} A_n \sqrt{\varepsilon_x} \text{rect}(\frac{t - nT_s}{T_s}) \tag{2.3}$$

Where:

$\boldsymbol{W_s = 1/T_s}$ is the bandwidth of signal $x(t)$.

$\boldsymbol{\varepsilon_x}$ is the energy of signal $x(t)$.

$\boldsymbol{T_s}$ is the symbol duration.

This leads to a spectrum defined by:

$$X_s(f) = \frac{T_s}{E_x} T_s^2 \text{sinc}^2 (T_s f) \tag{2.4}$$

Analogously, we build N-length ($N = T_s/T_c$) PN sequence $C = C_0, C_1, C_2, \cdots, C_{N-1}$:

$$g(t) = \sum_{k=0}^{N-1} C_k \text{rect}(\frac{t - nT_c}{T_c}) \tag{2.5}$$

Where $\boldsymbol{T_c}$ is chip duration.

By using $s(t)$ as the basic pulse shape for generating (ASK) signal, we get a spread-spectrum signal $s_g(t)$:

$$s_g(t) = \sum_{n=-\infty}^{n=+\infty} A_n \sqrt{E_s}(t - nT_s) \tag{2.6}$$

That has the spectrum $X_{s_g}(f)$:

$$X_{s_g}(f) = \frac{E_s}{T_s}|g(f)|^2 = \frac{E_s}{T_s} T_c^2 \text{sinc}^2(T_c f)|C(f)|^2 \tag{2.7}$$

Where:

$$C(f) = \sum_{k=0}^{N-1} C_k \exp(-j2\pi f k) \tag{2.8}$$

Thus:

$$\begin{aligned} |C(T_c f)|^2 &= \mathscr{F}\{R_c(m)\}(m \to T_c f) \\ &= \mathscr{F}\{N\delta_m(m \to T_c f) \\ &= N \end{aligned} \tag{2.9}$$

Where $\{R_c(m)\}$ is the autocorrelation function of the (PN) sequence that is generated by M binary registers with period $N = 2^M - 1$ such that there are (2^M) ones and ($2^{M-1} - 1$) zeros:

$$R_c(m) = \begin{cases} N, & m = \cdots, -N, 0, N, \cdots \\ -\frac{1}{N}, & \text{everything else} \end{cases} \tag{2.10}$$

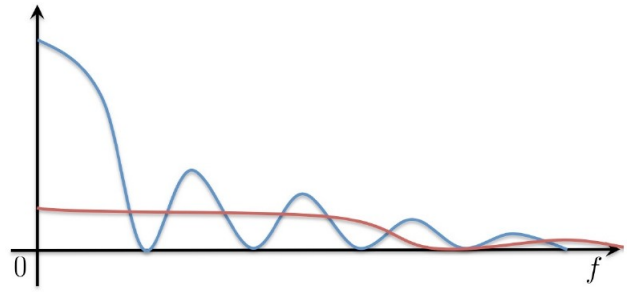


Figure 2.4: In blue, the spectrum of the ASK signal. In red, the spectrum of the spread signal.

Substituting in equation 2.7 yields:

$$\begin{aligned} X_{s_g}(f) &= \frac{E_s}{T_g} N T_c^2 \operatorname{sinc}^2(T_c f) \\ &= E_s T_c \operatorname{sinc}^2(T_c f) \\ &= \frac{E_s T_g}{N} \operatorname{sinc}^2(\frac{T_g}{N} f) \\ &= \frac{1}{N} X_s(\frac{f}{N}) \end{aligned} \tag{2.11}$$

This indicates the energy is actually spread over a much larger bandwidth than the one originally essential for the data signal as illustrated in figure 2.4.

Chapter 3

Specifications and System Analysis

Prior to developing any transceiver designs in HDL, simulations were performed in Matlab. To get a feel for how a certain transceiver architecture behaves in a real-world application. An approximated model for the communication channel was set to pass transmitted data from a transmitter model to a receiver model.

3.1 Introduction

In order to develop a digital transceiver that communicates efficiently over the communication channel, it is helpful to think about the system in a very abstract view. Starting from the data transmitter as shown in Figure 3.1. The input to the transmitter is raw binary data represented as a bit stream $a(n)$. Digital transmission over a communication channel is by definition analog, therefore, the transmitter and the receiver interfaces to the channel are analog, regardless of the digital signal processing chain in both sides. The analog signal $s(t)$ produced at the transmitter is sent over the channel and arrives at the receiver as a distorted, added-noise signal $y(t)$. The

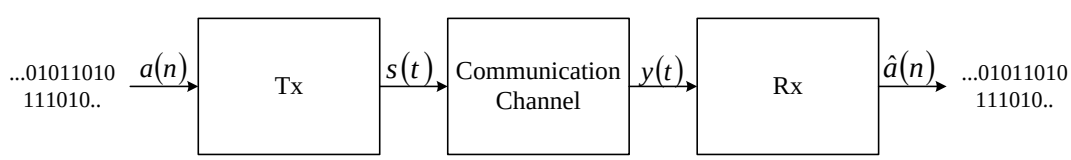


Figure 3.1: Abstract view of the digital communication system

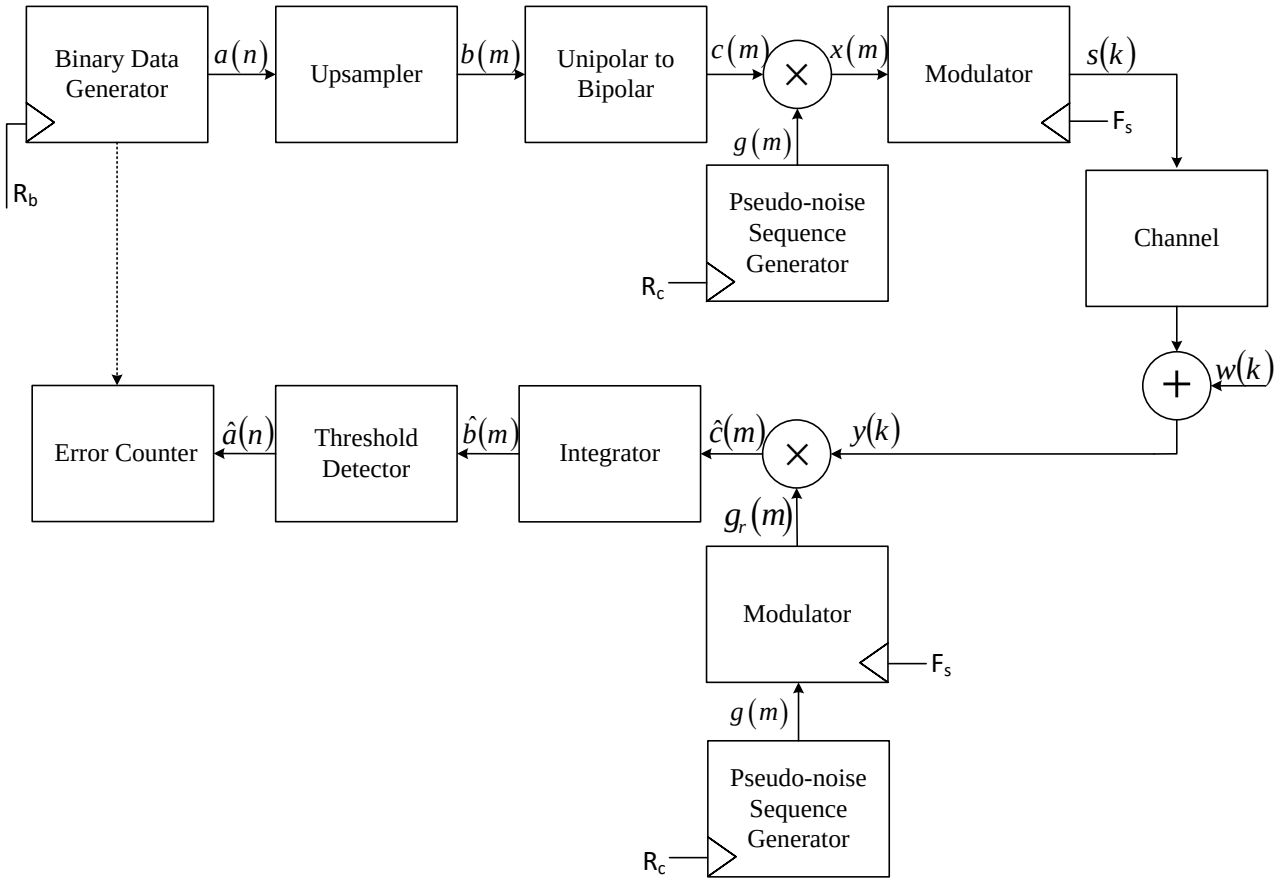


Figure 3.2: MATLAB Model of the System

receiver task is to undo the distortions introduced by the communication channel and to demodulate the received signal. Eventually, its output $\hat{a}(n)$ is a binary sequence. In an ideal case, it is identical to the sequence introduced into the transmitter.

3.2 System Modeling and Simulation

A simulation model is developed in order to get an objective measurement of the impact of given set of system parameters on the performance. Figure 3.2 illustrates the complete model used in the purpose of evaluating the system performance. The simulation model is developed in a way which would successively estimates the overall performance of the system by varying one of the two system parameters: The transmitted bit frequency R_b, and the spreading factor of the system G_p (processing gain), which is by definition is the ratio between the chip rate R_c

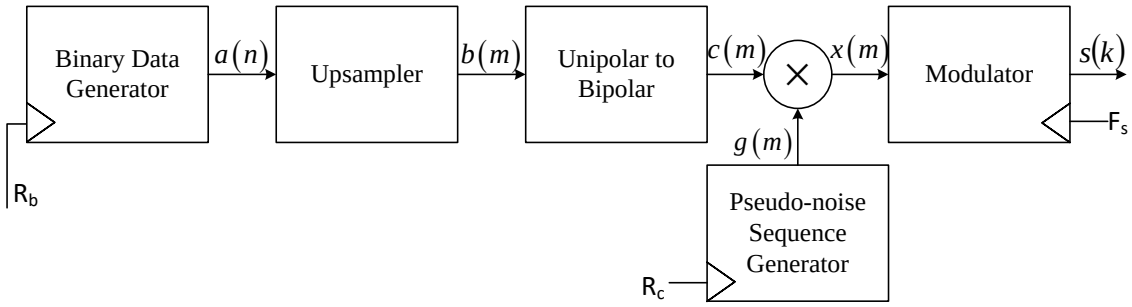


Figure 3.3: Transmitter Model

and the bit rate R_b. Therefore:

$$G_p = \frac{R_c}{R_b} \tag{3.1}$$

3.2.1 Transmitter Modeling

Common digital communications systems employs two signal waveforms, say $s_1(t) = s(t)$ and $s_0(t) = -s(t)$ to transmit binary sequence representing the information signal. The signal waveform $s_0(t)$, which is non-zero over the interval $0 \leqslant t \leqslant T$ is transmitted to the receiver if the data bit is zero. The same applies for the signal waveform $s_1(t)$, which is non-zero over the interval $0 \leqslant t \leqslant T$ is transmitted to the receiver if the data bit is one. The time interval (T) is called the signal interval, and the bit rate flowing through the channel is $R_b = 1/T$ bits per second. In practice, the signal waveforms transmitted over the channel are corrupted due to the additive noise and other types of channel distortions that ultimately limit the performance of the communication system. In order to put the previous description in practice, various MATLAB modules are used to setup the transmitter model. Figure 3.3:

- Binary Data Generator module which creates sequences of independent binary digits with equal probability $a(n)$ and a bit rate of $R_b = 4\,\text{kHz}$.

- Upsampler module which achieve the processing gain G_p of the system i.e. the spreading factor. It takes information bits $a(n)$ from the Binary Data Generator and repeats each bit G_p times yields the signal $b(m)$ with a rate of $R_c = G_p \times R_b$ where G_p is defined by equation (3.1) in the previous section.

- In order to enable the multiplication with the bipolar pseudo-noise sequence $g(m)$, the up sampled stream from the Upsampler module is converted from the unipolar representation to the bipolar representation $b(m)$ by replacing the zeros on the stream to (-1) and keeping the ones as they were. No changes in the rate in this module hence the same symbol (m).

- Pseudo-noise Sequence Generator: To ensure a long, non-repeating sequence, an LFSR structure of the order of twenty is chosen for the simulations. Therefore, the pseudo-random sequence has a period of $p = 2^{20} - 2$. The operational frequency of this module is governed by the spreading factor G_p which is defined by equation (3.1) in the previous section.

- The signal $x(m)$ is the product of multiplying the pseudo-noise sequence $g(m)$ with the signal $c(m)$:

$$x(m) = c(m).g(m) \tag{3.2}$$

  which is passed to a modulator module that maps a binary digit of one into a sequence of consecutive $+1$, and maps a binary digit of zero into a sequence of consecutive -1. This mapping (repetition) is determined by the sampling frequency F_s. Thus the signal $s(k)$ represents a sampled version of a rectangular pulse with a sampling frequency of F_s.

3.2.2 Transmitter Simulations

To get the simulation running, the first task is to generate a certain number of random, independent bits $a(n)$. Although the requirement of the project specified the data bit rate to be $4\,\text{kbit}\,\text{s}^{-1}$, the bit rate chosen in the simulation is $R_b = 4\,\text{kbit}\,\text{s}^{-1} \times 8 = 32\,\text{kbit}\,\text{s}^{-1}$ in order to satisfy the requirement that states the channel should be occupied 1/8 of the time or less.

The sampling frequency F_s defines the number of samples within a certain signal produced by the Modulator module $s(k)$. it is set to $F_s = 40\,\text{MHz}$ since the upper frequency limit of the channel is $F_{max} = 10\,\text{MHz}$.

As mentioned earlier in the previous section, once the bit stream is converted from the unipolar representation $b(m)$ to the bipolar representation $c(m)$, it is multiplied by a bipolar

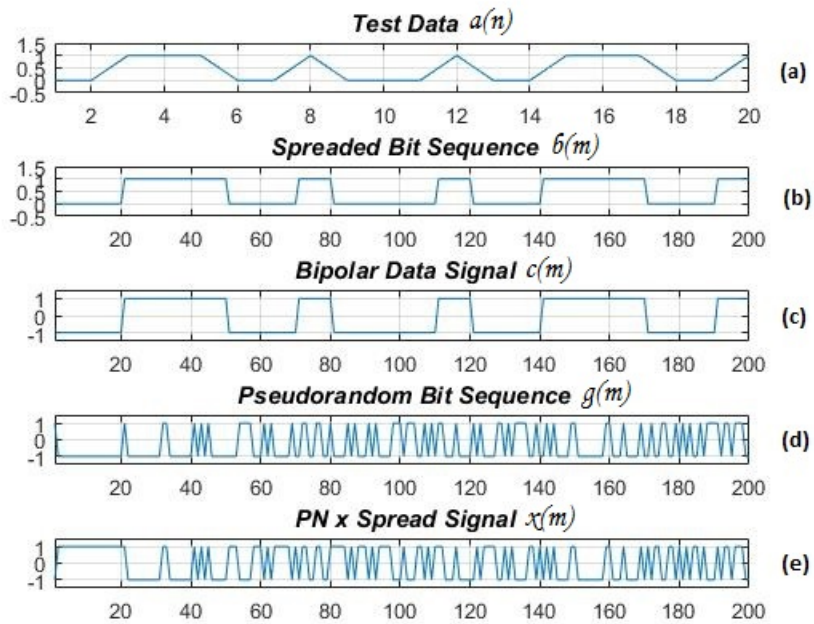


Figure 3.4: Generated signals at the transmitter (a). Random bits (b). Spread bits (c) Bipolar from (d) Generated PN (e) spread signal

pseudo-random sequence $g(m)$. Figure 3.4 illustrates the process starting from generating a random, independent bits in Figure 3.4.a, till the final form of the transmitted signal in figure 3.4.e $\big(x(m)$ at the Transmitter Model 3.3$\big)$.

The resulted signal of multiplying the pseudo-random sequence $g(m)$ with the bipolar signal $c(m)$ is passed to a modulator module that outputs $x(m)$ as defined in equation (3.2). Thus the product signal represents a sampled version of the rectangular pulse, which is the direct-sequence spread spectrum signal that shall be transmitted through the channel.

3.2.3 Communication Channel Modeling

The communication channel allows a wide frequency spectrum to pass through it, specifically from 200 kHz to 10 MHz. Moreover, a unique noise signal is introduced to this channel due to the switching activity of the charger. According to Atmel, the channel can be modeled as an RC network which has the property of a low cutoff frequency of $F_{min} = 200\,\text{kHz}$ and a high cutoff frequency of $F_{max} = 10\,\text{MHz}$, therefore it can be modeled as a bandpass filter. Butterworth

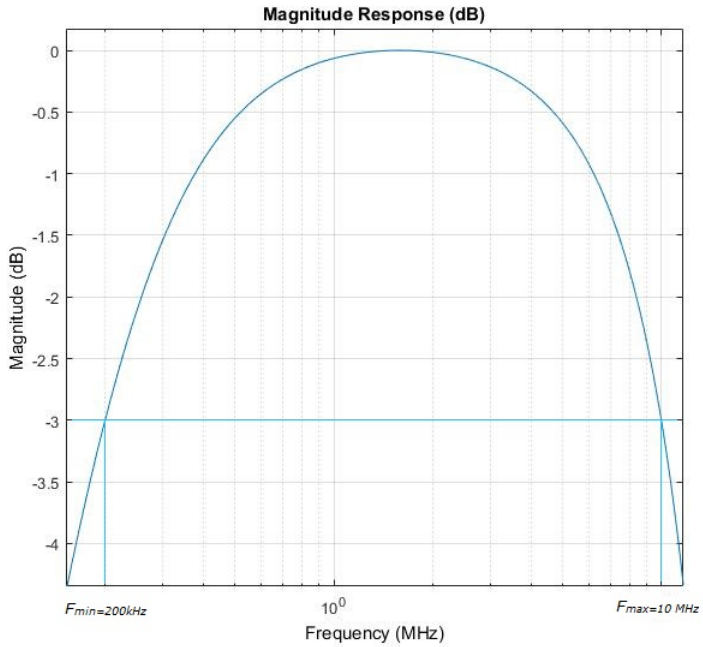


Figure 3.5: Magnitude frequency response of the channel model

filter is chosen as modeling choice due to its property of having a flat frequency response in the passband region, which is convenient for the studied channel. Figure 3.5.

Regarding the noise introduced by the charger to the channel, it was modeled at the beginning of the project as a square, periodic, signal with an amplitude of 10 V and duty cycle of 43%. At a later point, to produce a closer results to a real-world scenario, Atmel provided a real sample of the charger noise. Figure 3.6 illustrates the two noise signals used in simulations.

Summing up the above, the mathematical representation of the communication channel is:

$$y(k) = \big(h(k) * s(k)\big) + w(k) \tag{3.3}$$

3.2.4 Receiver Modeling

MATLAB receiver model is illustrated in Figure 3.7. To enable the recovery of the transmitted signal, a multiplication of the pseudo-noise sequence with the received signal $y(k)$ must be done. The pseudo-noise sequence must be synchronized between the transmitter and the receiver, therefore, an exact replica of the pseudo-noise sequence used in the transmitter is used in the receiver. However, the pseudo-noise sequence must be modulated in the same manner where the

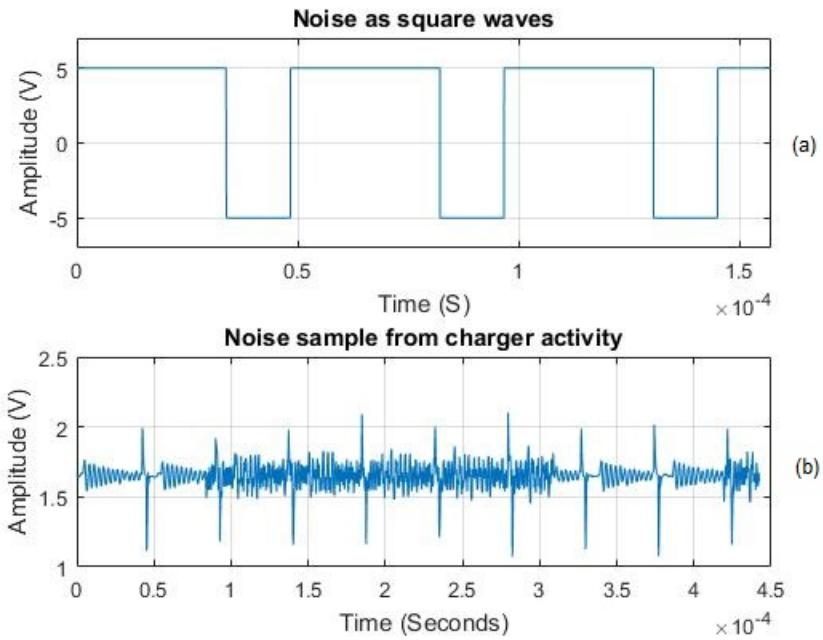


Figure 3.6: Noise Modeling. (a) Noise modeled as periodic square waves. (b) Actual Noise Sample

transmitter signal is modulated. Therefore, the pseudo-noise sequence $g(m)$ is passed through the same modulator module found in the transmitter which will output $g_r(m)$. Therefore:

$$\hat{c}(m) = g_r(m).y(m) \tag{3.4}$$

The later signal is passed to the Integrator module. It integrates (sums) samples blocks of length G_p that each should represent a single spread information bit $\hat{a}(m)$. The integrated bits are passed to the Threshold Detector module. It compares the results of the integration with a predefined threshold (η) which is in the simulation defined as (0). At this point, the decision is made whether the transmitted bit is a zero or a one as follows: in the case of the result of the integration is bigger than zero. the estimated bit is one:

$$\left(\sum_{n=(k-1)G_p+1}^{k=G_p} \hat{c}(m) \right) > 0 \xrightarrow{yields} \hat{a} = 1 \tag{3.5}$$

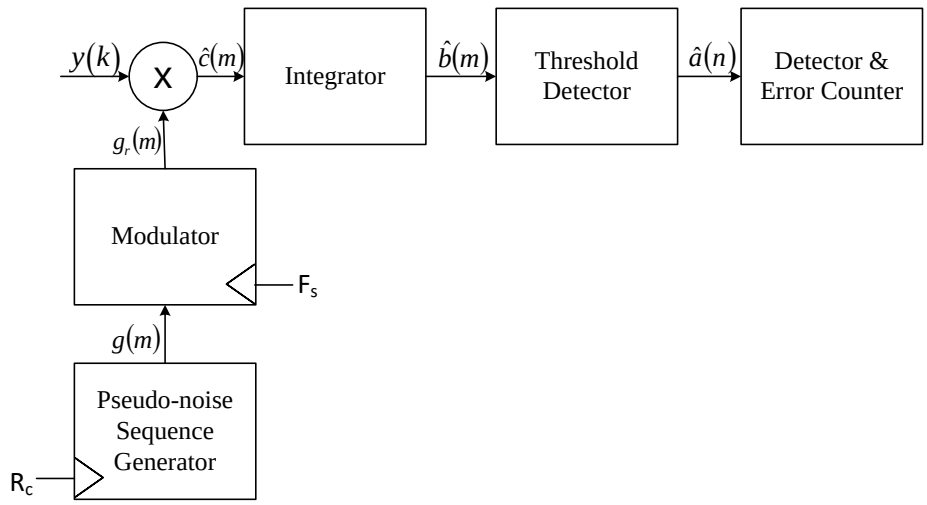


Figure 3.7: Receiver Model

In the other case when the integration is smaller than zero, the estimated bit is zero:

$$\left(\sum_{n=(k-1)G_p+1}^{k=G_p} \hat{c}(m)\right) < 0 \xrightarrow{yields} \hat{a} = 0 \tag{3.6}$$

The last module of the receiver is a Detector module which allows measurements of performance of the system. The average probability of an error is calculated by a simple comparison between the estimated bit sequence produced by the receiver and the original bit sequence produced by the transmitter, specifically the random bit generator.

3.2.5 Receiver Simulations

The simulation is developed to allow three options regarding the noise added while passing through the channel. It is possible to receive the signal with the charger noise added, or without any noise added, or directly from the transmitter to enable code debugging. The received signal $y(k)$ which has gone through the noisy channel is illustrated in figure 3.8.a.

The distortion that occurred on the transmitted signal is an expected result of going through the noisy channel. $b(m)$ is the product of multiplying the pseudo-random sequence $g(m)$ and the received signal $y(k)$:

$$b(k) = g(m).y(k) \tag{3.7}$$

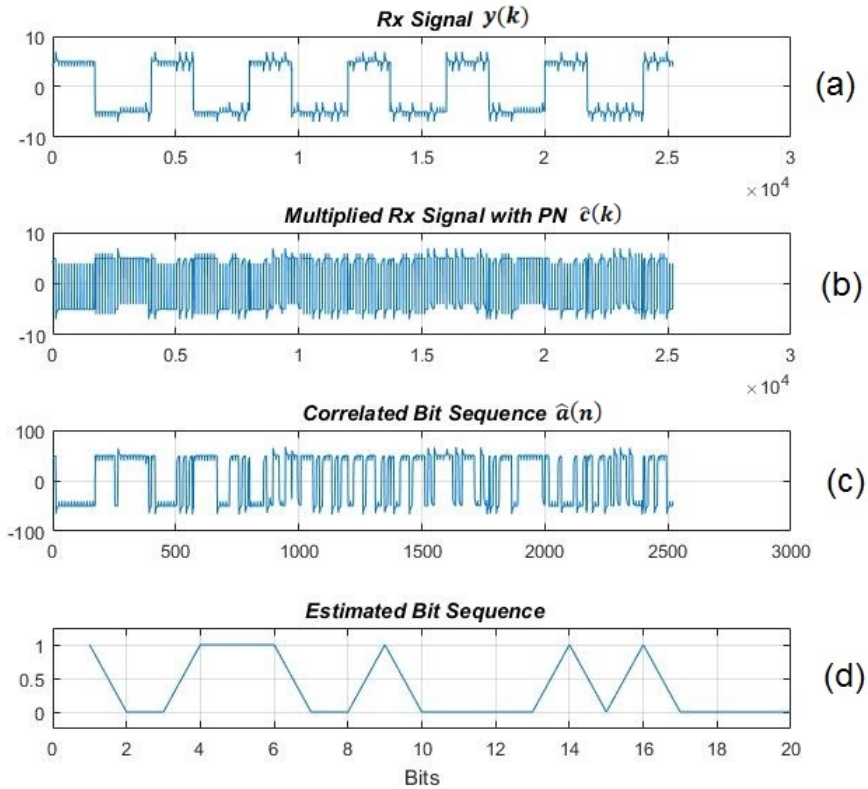


Figure 3.8: Signals at the receiver (a). Received signal (b). Dispread signal (c). Integrated signal (d). Estimated transmitted bits

Therefore, $b(m)$ is the input of the Detector module which will integrates (sums) the data samples according to the two equations (3.5) & (3.6). Once the integration is done, an estimation of the received bit can be done by comparing the integration result to a threshold $(\eta) = 0$.

Chapter 4

Architectural Design

This chapter provides implementation details for the digital baseband transceiver. The discussion starts with the detailed architecture, the simulation of the transmitter, then it moves on to detailed implementation of the receiver which is the main focus of this thesis.

4.1 Introduction

The design proposed here is intended to be used with a 1-bit ADC at the interface of the receiver with the communication channel. It detects whether the input signal is above or below a predefined value represented by a one or a zero respectively. The implantation of the transceiver is divided into two main modules; a transmitter and a receiver. The SystemVerilog hardware description language is used for implantation and verification. The implanted design is verified through simulation using ModelSim. The synthesis and the real-world verification is done using Altera development boards, which will be discussed in the following chapter. The following sections further explain the implemented design.

4.2 Transmitter Architecture

The architecture is quite simple. It is based on the spread spectrum modulation discussed in chapter two, and implemented in SystemVerilog according to the models in chapter three. The hardware architecture of the transmitter consists of two modules as illustrated in Figure 4.1: a

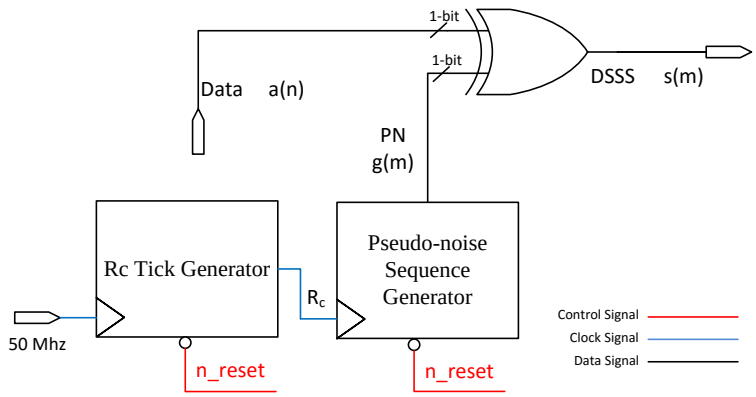


Figure 4.1: Transmitter RTL architecture.

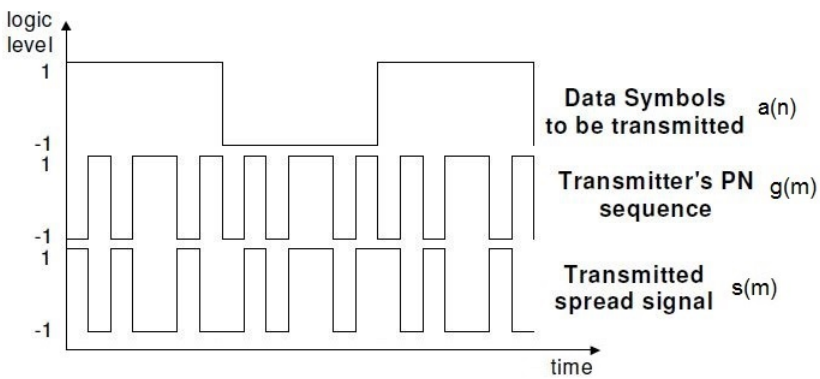


Figure 4.2: Direct sequence spreading with exclusive OR operation

Tick (clock) Generator and a Pseudo-noise sequence generator. The data stream $a(n)$ clocked-in at a rate R_b which is parametrized throughout the HDL implementation. However, for the connivance of the implementation is it chosen to be $R_b = 4\,\text{kHz}$ rather than $32\,\text{kHz}$ as it was in chapter three. The PN sequence stream $g(m)$ generated by the pseudo-noise sequence generator module which is set to a chip rate of R_c. As a reference architecture, the processing gain is chosen to be $G_p = 5$ which yields a chip rate of:

$$R_c = G_p \times R_b = 5 \times 4 = 20\,\text{kHz} \tag{4.1}$$

Since the data input $a(n)$ to the transmitter module is a 1-bit at a time, the multiplication operation is done by an exclusive OR operation as illustrated in Figure 4.2. The next following sections will discuss the implementation of those two modules.

4.2.1 Parametrized Tick Generators

Commonly, development boards are equipped with one or two oscillator clocked at a high, fixed frequency such as 50 MHz. Moreover, a number of lower clocking frequencies are required thorough out the whole design. Therefore, a tick (clock) generator that can produce any required clocking frequency is essential for the synthesis on the development boards. All the tick generators presented in this thesis have been parametrized to serve the purpose mentioned earlier. If the required clock frequency is a power of two, the logic to produce such clock is easy, the conventional solution is to use of a counter to divide the main clock. However, if the ratio between the main clock to be divided and the required frequency is not an integer producing a clock ticks at arbitrary frequency requires more math than a simple devision.

Let (C1) be the main clock at frequency F_1, (C2) the lower required clock at frequency F_2, and (Acc) is a counter accumulates the ratio F_2/F_1 at every tick of (C1). Once (Acc) crosses one, we make a tick on clock (C2) as detailed in the pseudo-code below:

```
Acc = 0;
forever
Wait for clock C1.
if (Acc < 0)
  Acc += (F2/F1);
else
  Acc += (F2/F1) - 1;
```

However, the forever loop in the previous listing assumes the variable (Acc) and the ration F_2/F_1 are at infinite precision. To present them as integers a simple multiplication by F_1 will solve it:

```
Acc = 0;
forever
Wait for clock C1.
if (Acc < 0)
  Acc += F2;
else
  Acc += F2 - F1;
```

Regarding SystemVerilog implementation, the length of the (Acc) register should be chosen to represent the largest negative value $Acc = F_2 - F_1$ plus an extra bit to represent the sign. For

instance, if the frequency of the main clock at the development board is $F_1 = 50\,\text{MHz}$ and the required clock frequency is $F_2 = 20\,\text{kHz}$ the length of the (Acc) register is 27-bits as illustrated in the parametrized Verilog code below:

```
//-----------------------------------------------------
// Design Name: TickGen
// Function: Arbitrary Clock Frequency Generator
//-----------------------------------------------------
module TickGen #(parameter ReqClk = 20000) (
  input logic Clk_in_F1,
  input logic n_reset,
  input logic Acq_reset,
  output logic Clk_out_F2
);
  logic [26:0] Acc;
  logic [26:0] dN;
  logic [26:0] dInc;
always_comb
  begin
    dInc = Acc[26] ? (ReqClk) : (ReqClk - 50000000);
    dN = Acc + dInc;
  end
always_ff @(posedge Clk_in_F1, posedge Acq_reset, negedge n_reset)
  if (Acq_reset || !n_reset)
    Acc <= 0;
  else
    Acc<= dN;
always_comb
  Clk_out_F2 = !Acc[26];  // Clock C2 tick whenever Acc[26] is zero
endmodule
```

As can be noticed from the listing above, the sequential part of the code (`always_ff` block) is sensitive for the `Acq_reset` signal which resets the counter (Acc) to zero as soon as it is asserted, i.e. asynchronous reset signal. The ability to reset (sync) the tick generator at arbitrary moments is crucial when it comes to implementing the synchronization functionality at the receiver side which will be explained further in the synchronization subsection.

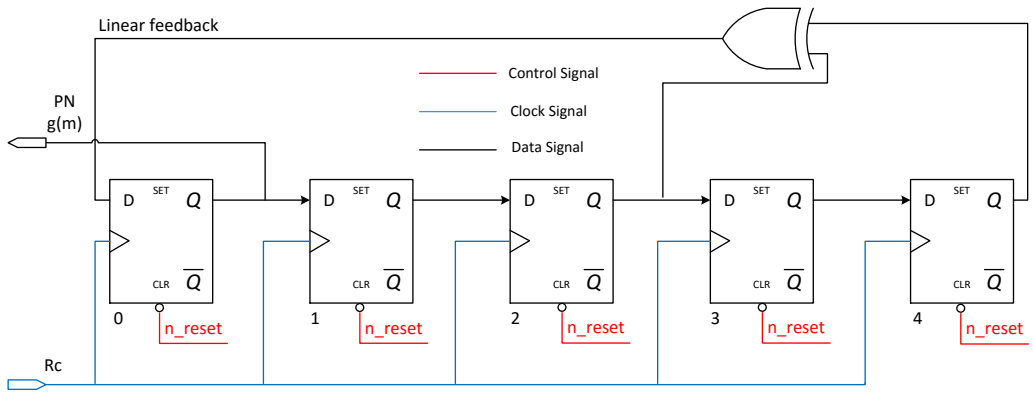


Figure 4.3: Fibonacci LFSR RTL implementation.

4.2.2 Pseudo-noise Sequence Generator

The Pseudo-noise Sequence Generator is based on Fibonacci linear-feedback shift registers logic which is illustrated In Figure 4.3. The bit positions that influence the next state are called taps. The taps in this implantation are [2,4]. The leftmost bit of the LFSR is the output bit $g(m)$. The taps are exclusively ORed sequentially (at the clock tick R_c) with the output bit $g(m)$ and then fed back to the leftmost bit. A maximum-length LFSR produces m-sequence since it cycles through all possible $2^m - 1$ states within the shift register. In this implementation, the number of shift registers m which determines the period of the pseudo-random code is set to $m = 5$ according to the following:

$$L = 2^m - 1 = 2^5 - 1 = 31 \tag{4.2}$$

The Systemverilog implementation is parametrized to enable code reuse. The main component of the LFSR structure is the shift register itself. The taps and the linear feedback line are defined by combinational assignment block (`always_comb`).

Similarly to the tick generator module, since the Pseudo-noise generator is identical in both the receiver and the transmitter, the `Acq_reset` signal is crucial to reset (synchronize) this module at any arbitrary moment once the signal (`Acq_reset`) is asserted. Therefore, the sequential block (`always_ff`) will force a reset upon the assertion of the (`Acq_reset`) signal no matter what is the state of the module clock.

```
//-----------------------------------------------------
```

```
// Design Name  : LFSR
// Function     : Fibonacci Linear feedback shift register
//-----------------------------------------------------
module LFSR #( parameter m = 5)(
  output  logic out,
  input   logic clk,
  input   logic n_reset,
  input logic Acq_reset
);
  logic linear_feedback;
  logic [m-1:0] lfsr;
always_comb
  begin
    linear_feedback = (lfsr[m-1] ^ lfsr[m-3]);
    out = lfsr[m-1];
  end
always_ff @(posedge clk, posedge Acq_reset, negedge n_reset)
  if (!n_reset)
    lfsr <= {{(m-1){1'b0}},1'b1};
  else
    begin
      if (Acq_reset)
        lfsr <= {{(m-1){1'b0}},1'b1};
      else
        lfsr <= {lfsr[m-2:0],linear_feedback};
    end
endmodule
```

4.2.3 Simulation

The ModelSim simulation tool is used for all the simulations in HDL. A testbench was created for the behavioral verification of the transmitter functionality. The testbench generates the appropriate stimuli to the design under test (DUT), which is the transmitter in this case. To judge a correct behavior from the implementation, the output signal (DSSS) should be the result of an exclusive OR operation between the (Data) signal and the (PN) signal defined at rising edge of

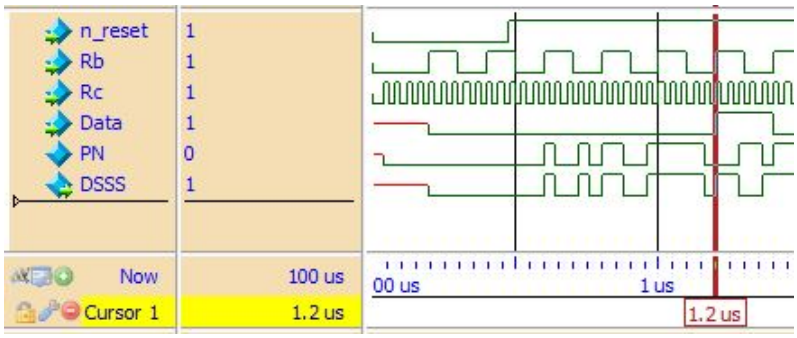


Figure 4.4: This simulation diagram shows different waveforms of transmitter module.Clocking signals denoted as R_b & R_c, transmitter output signal denoted as (DSSS), data signal denoted as (Data).

the clock (R_c). The simulation shown in Figure 4.4 confirms a correct behavior. Once the reset signal is asserted (active low reset) the simulation starts. When the (Data) signal was low, the output of the transmitter (DSSS) was identical to the (PN) signal. As soon as the (Data) signal goes high at the red indicator, the (DSSS) signal is a clear result of an exclusive OR operation between the (Data) signal and the (PN) signal.

4.3 Receiver Architecture

The baseband receiver is implemented as in Figure 4.5. It illustrates the top-level architecture of the proposed design. The synchronization between the transmitter clock and the receiver clock is a critical part of the receiver design. The acquisition is performed by detecting a unique word that is generated by the transmitter once a preamble sequence. The design choice for the preamble code is a stream of t zeros. Once the data to be transmitter from the transmitter is set to a stream of zeros, the transmitted DSSS signal is basically the local PN sequence itself due to executing exclusive OR operation between the PN sequence and the data to be transmitted as illustrated earlier in the transmitter architecture, Figure 4.1. Since the length of the PN sequence is thirty one chips with a chip rate $R_c = 20\,\text{kHz}$ therefore the chip period is

$$T_c = \frac{1}{R_c} = \frac{1}{20000} = 50\,\mu\text{sec} \tag{4.3}$$

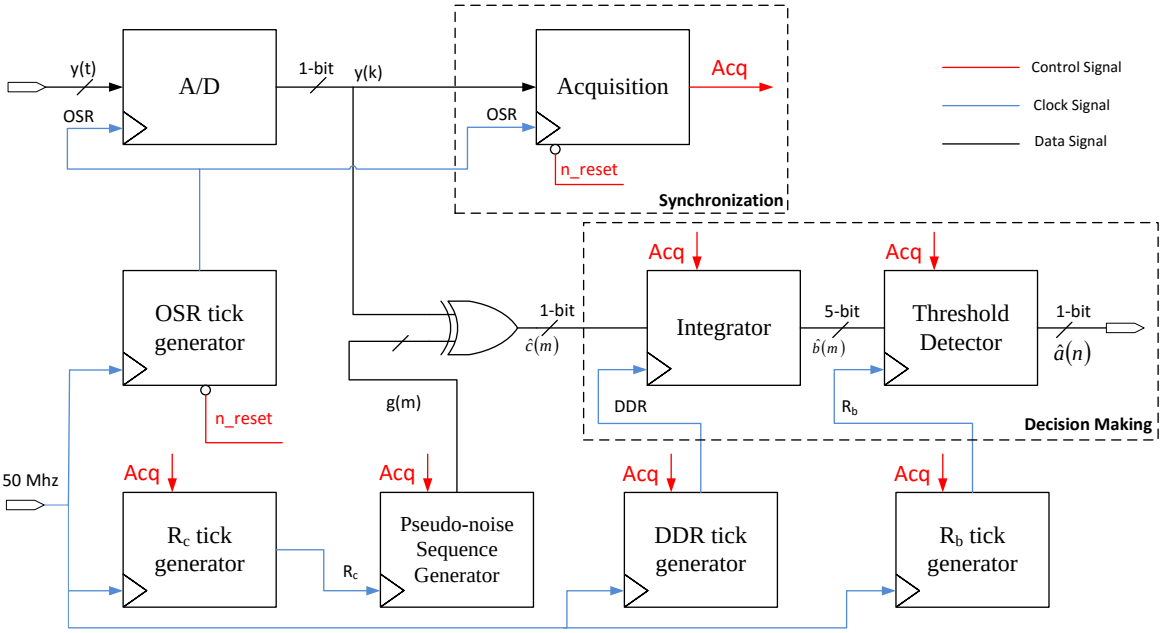


Figure 4.5: Receiver RTL Architecture.

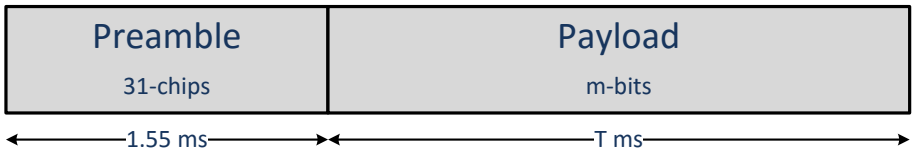


Figure 4.6: Baseband transceiver packet format.

Therefore, the length of the preamble stream is:

$$
\begin{aligned}
L_{pre} &= \text{Number of chips} \times \text{Chip period} \\
&= 31 \times 50\,\mu\text{sec} \\
&= 1.55\,\text{m sec}
\end{aligned}
\tag{4.4}
$$

Hence, the packet format is illustrated in Figure 4.6. It has been found in the real-time demonstrations that the length of the data payload influence the performance of the proposed architecture; It will be explained later that the transmitter only implemented an acquisition logic to serve for the synchronization. Therefore, once the preamble sequence is received by the receiver, there is no implemented logic to maintain the synchronization which is commonly

referred to as tracking. That being said, the length of the data payload is a design choice determinate the overall performance of the system. In chapter six, different data payload lengths were tested in real-time to have more insight in the system performance.

The Acquisition module asserts a control signal Acq once the unique word is detected which force the whole receiver into synchronizing its clock to the one in the transmitter. While the synchronization is maintained, the other modules of the receiver works on de-spreading and estimating the received data bits.

The A/D works as a simple threshold detector since it converts the analog received signal $y(t)$ to either a 1-bit logic high, or 1-bit logic low. It is worth mentioning that the incoming spread-spectrum signal is oversampled by an oversampling rate of OSR to enable the synchronization. Therefore, the A/D is being clocked by an oversampled version of the PN sequence clock namely OSR. The chosen oversampling ratio for the reference implementation is five which means:

$$OSR = R_c \times 5 = 100\,\text{kHz} \tag{4.5}$$

Those received bits $y(k)$ are multiplied by the local PN sequence at the receiver $g(m)$ producing $\hat{c}(m)$. Similarly to the transmitter, the multiplication is done by an exclusive OR operation.

Prior to estimating the received bit if it is a one or zero, an integration operation is needed as explained in chapter three. To recover a single bit, the integrator module sums up twenty five samples of $\hat{c}(m)$ at a rate of ($DDR = 200\,\text{kHz}$) and output an integration result to the threshold detector module. Its input is the 5-bit integration result $\hat{b}(m)$ and its output a 1-bit estimated received bit $\hat{a}(m)$.

4.3.1 Synchronization

In chapter three, the design problem is approached with the assumption of an ideally synchronized clocks between the transmitter and the receiver which is not the case for a real-world scenario. A sensitive part of a direct sequence, spread spectrum communication system is the synchronization of the transmitter's pseudo-random sequence to that of the receiver.

Synchronization is generally achieved by two functions: acquisition and tracking. Acquisition is responsible for the alignment of the PN sequence, whereas the tracking is responsible for

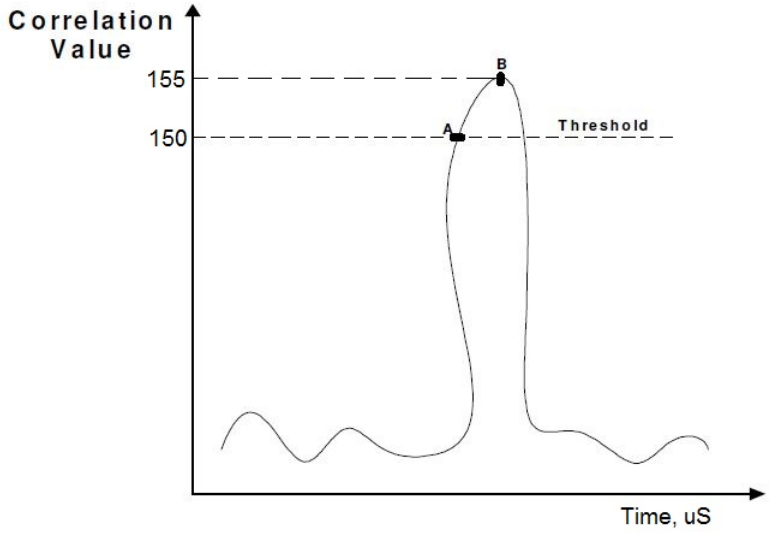


Figure 4.7: Correlation of received signal to PN

maintaining the alignment. Synchronization architectures use correlators to estimate the correlation of the received signal to the local copy of the transmitted PN sequence. Acquisition is said to be achieved once a high correlation value is detected as illustrated on Figure 4.7. The value at which this decision is made (A) depends on a predetermined threshold value which depends mainly on the characteristics of the communication channel.

If we assume a direct sequence, spread spectrum communication system without a tracking mechanism, the receiver would sample the correlation value at symbol intervals of point (A) on Figure 4.7. However, any delay originated from any part of the communication system (transmitter, channel, or the receiver) can cause the system to fall out of acquisition. Therefore, a tracking algorithm adjusts the sampling time in which the sampling of the correlation value happens as close as possible to point (B). Since the specification of the project states that the data will be transmitted in small bursts, therefore only the acquisition part of the synchronization is implanted. The implementation details are discussed in the following sections.

Acquisition

Acquisition architectures are commonly based on matched filters which are in turn based on finite impulse filter structure (FIR) with the PN sequence serving as the filter tap coefficients. The FIR structure convolves the PN coefficient sequence X_c with the samples of the received

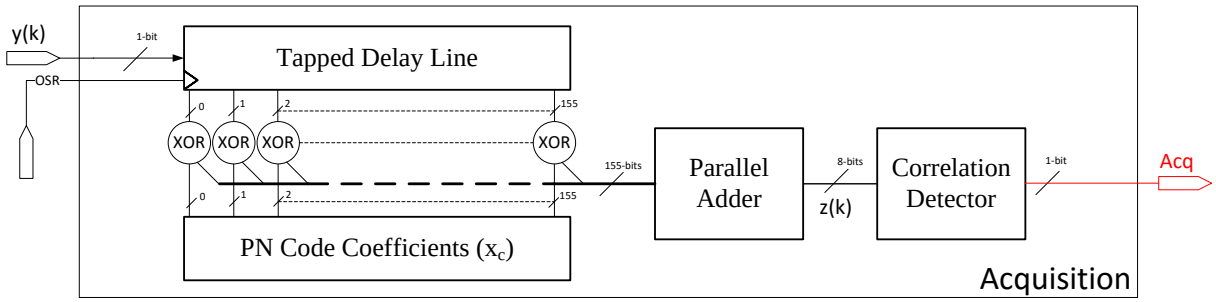


Figure 4.8: Implementation of the Acquisition module

spread spectrum $y(k)$:

$$z(k) = \sum_{l} X_c(l)\, y(k-l) \tag{4.6}$$

As the received sequence $y(k)$ slides through the tapped delay line, the multiply and accumulate process calculates the correlation value. Therefore, a new correlation value is calculated at each clock cycle, which is in this case is the oversampling rate (OSR).

Taking into account the input sequence is a stream of ones and zeros, the implemented design utilizes exclusive OR operation instead of multiplications. As mentioned earlier in equation (4.2), the length of the PN sequence used is $L = 31$. However, the oversampling rate OSR is chosen to be five times higher than the chip rate R_c. Therefore, a 155-tap digital matched filter (31×5) is used as illustrated in Figure 4.8. This is done with the use of four components: Firstly, a 155-bit static register that stores an oversampled version of the PN sequence X_c. Secondly, a 155-bit shift register (Tapped Delay Line) to hold and shift the incoming samples $y(k)$. Thirdly, 155 exclusive OR gates. Finally, a 155-bit parallel adder which is implanted by using three onboard Altera's full adders as illustrated in Figure 4.9. The output range of the 155-tap digital matched filter is $z(k) = \{0, +155\}$. A in the peak correlation $z(k)$ (which is zero in this implementation) indicates the reception of a valid preamble code due to the fact of executing an exclusive OR operation between an equal number of zeros and ones. This is verified by simulation in ModelSim as Figure 4.10 illustrates. It is worth mentioning that the parallel adder implementation is fully combinational logic. Therefore, the moment the correlation between the stored PN code coefficients and the incoming samples is achieved, the parallel adder detects it instantly.

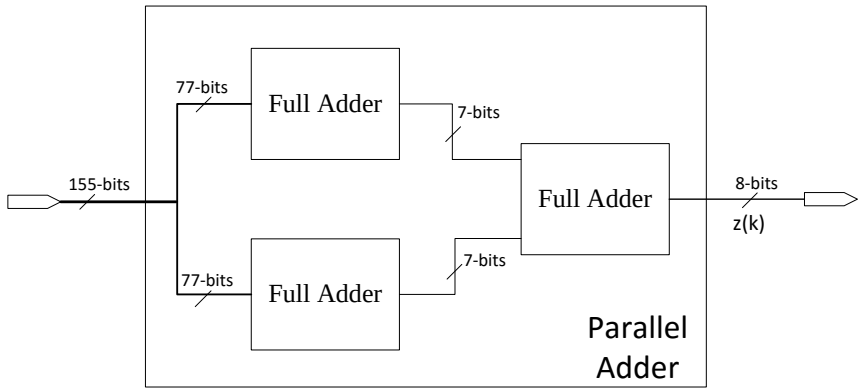


Figure 4.9: Implementation of Parallel Adder

4.3.2 Decision Making

The decision making process is achieved by two modules, the Integrator and the Threshold Detector. The integration operation is needed in this implementation to reconstruct the received, sampled, spread-spectrum signal. Since the oversampling rate OSR is five times higher than the chip rate R_c which is in turn five times higher than the bit rate R_b therefore, each twenty five samples produced by the A/D map into a one bit of the originally transmitted data from the transmitter. The integration operation is implemented in HDL by an accumulation logic. The same logic used for holding and shifting the sample in the Acquisition module is used for the Integrator. As Figure 4.11 illustrates, a 25-bit shift register (TDL) is used to shift and hold the samples at a rate of

$$DDR = 2 \times OSR = 200\,\text{kHz} \tag{4.7}$$

The double data rate, hence the acronym DDR, is a value chosen by experimenting different speeds of the integration operation in real-time demonstrations. In an ideal scenario, if the transmitter transmitted logic one, we would expect the output of the integrator is twenty five since the output range is $\hat{b}(m) = \{0, +25\}$, which is the maximum value it can output. Same argument holds true in the case of transmitting a zero, the ideal output from the integrator is zero. Therefore, a threshold detection logic is required to determine whether the received bit is a one or a zero. The implementation of the threshold detector is quite simple. It involves a

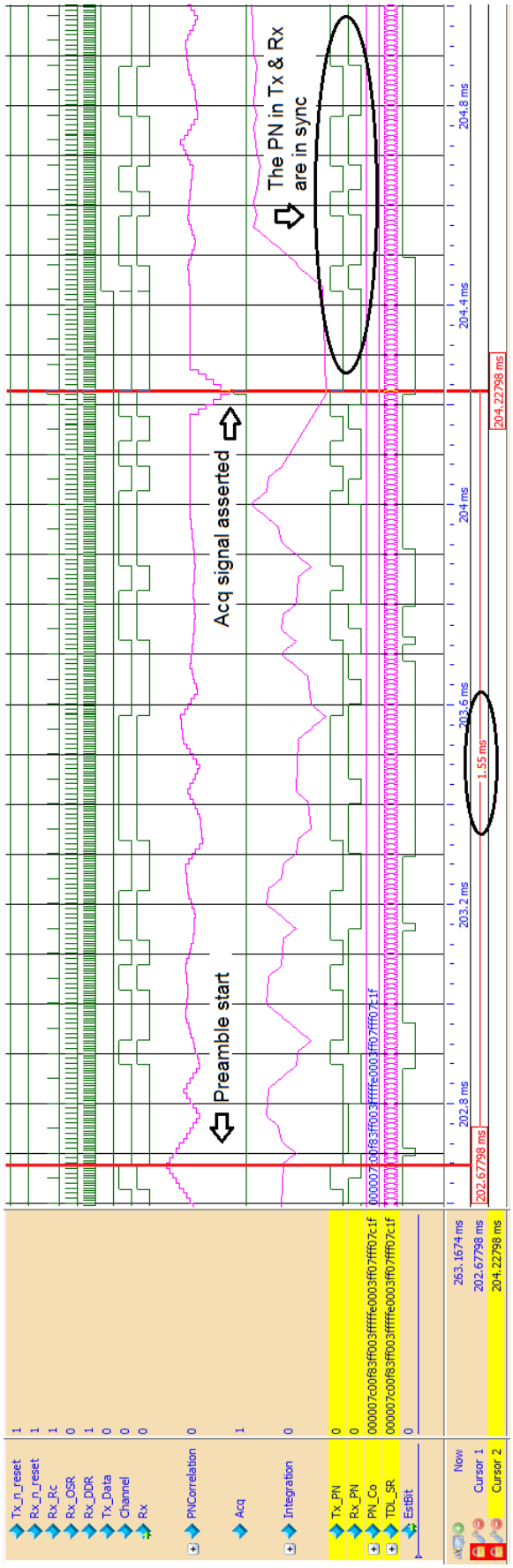


Figure 4.10: This simulation diagram shows different waveforms of the receiver module illustrating the acquisition and estimation process. Starting from sending the preamble of thirty one zeros at the first red highlighted cursor, detecting the preamble once the PN coefficients register PN_Co and the TDL shift register of the Acquisition module TDL_SR at the second red cursor. (Both highlighted in yellow, represented in HEX). Asserting the Acq signal, and forcing the R_x PN code to be in sync with T_x PN code. The time needed to achieve synchronization defined between the start of the preamble code and the detection of the correlation is 15.5 msec as expected from equation **??** and highlighted in this simulation diagram.

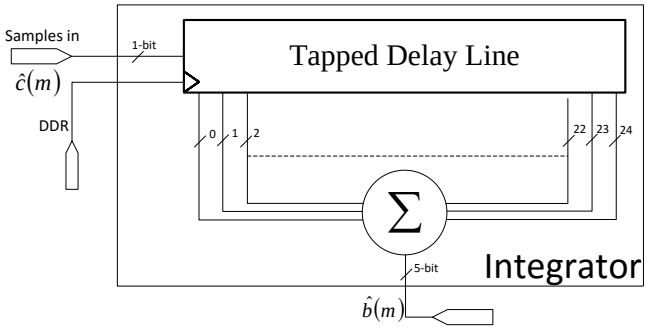


Figure 4.11: Implementation of Integrator

comparator asserting $\hat{a}(m)$ signal once the integration value surpassed twelve. (Mid value of the range $\{0, +25\}$). The opposite is true, if the integration value is less than twelve, the estimated bit $\hat{a}(m)$ is decided to be a zero. Therefore, the threshold detector module de-asserts the $\hat{a}(m)$ Signal. As can be seen in Figure 4.9, once the Acq is asserted, the estimated received bit signal ($\hat{a}(m)$ =EstBit) begin to matching the transmitted bits from the transmitter T_x_Data.

Chapter 5

Simulation Results and Hardware Verification

This chapter illustrates the performance of the proposed modulation scheme (Spread-spectrum modulation) regarding its effectiveness when it comes to transmitting and receiving digital data through the studied communication channel in chapter two. Moreover, verifying logical correctness of the Verilog modules that are based on the Matlab modules is done in real-time by demonstrating the proposed transmitter architecture and the receiver architecture on two separate FPGAs that are connected only through a wire between the transmission port of the transmitter and the receiving port of the receiver.

5.1 Simulation Results

The performance evaluation of the proposed architecture is based on average bit error rate (BER). All simulations are evaluating the error in receiving the bit stream in the case of sending a 1000 bits at a rate of $R_b = 32\,\text{kHz}$. Four simulation runs were conducted to calculate the BER. For three of them, either the bit rate R_b, the processing gain G_p, or the noise frequency F_{noise} is changed in order to evaluate the individual impact on the overall system performance. Moreover, the noise signal used in simulations is the square wave signal described in Chapter three which the square wave signal. The forth simulation run estimated the system performance using a noise sample provided by Atmel.

The signal-to-noise in all simulations is by definition the ratio between the transmitted signal power to the noise power:

$$SNR = \frac{P_{signal}}{P_{noise}} \tag{5.1}$$

The amplitude of the transmitted signal is fixed to one Volt in all simulations to come. On the other hand, the amplitude of the noise signal is A_{noise} which varies from one simulation to the other. Therefore, and since the signal and noise signals are measured in Volts, the signal-to-noise ratio in decibels is :

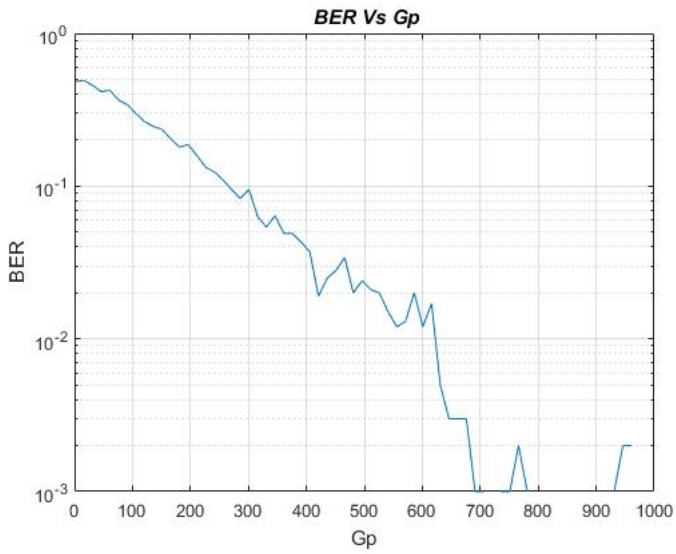
$$SNR_{dB} = 10\log_{10}\left[\left(\frac{1}{A_{noise}}\right)^2\right] \tag{5.2}$$

5.1.1 BER Against Spreading Factor (G_p)

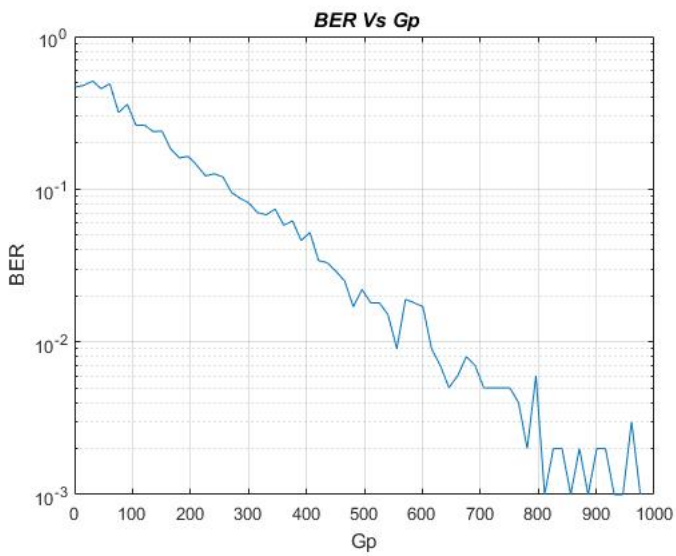
In this simulation run, the BER is computed as a function of the spreading factor of the system G_p. The spreading factor of the system took the values $G_p = 1$ to $G_p = 1000$ with incremental steps of 25 each. The noise amplitude is set to $A_{noise} = 5\,\text{V}$ for all trails in Figures 5.1a, 5.1b, and 5.1c. The frequency of the noise signal (square wave signal) is set to $F_{noise} = 500\,\text{kHz}$ in 5.1a. Whereas in Figure 5.1b and 5.1c the frequency is set to $F_{noise} = 1\,\text{kHz}$ and $F_{noise} = 1\,\text{MHz}$ respectively. The noise amplitude A_{Noise} is fixed in this simulation run. Hence, the signal-to-noise ratio in this run is:

$$SNR_{dB} = 10\log_{10}\left[\left(\frac{1}{5}\right)^2\right] = -13.97\,\text{dB} \tag{5.3}$$

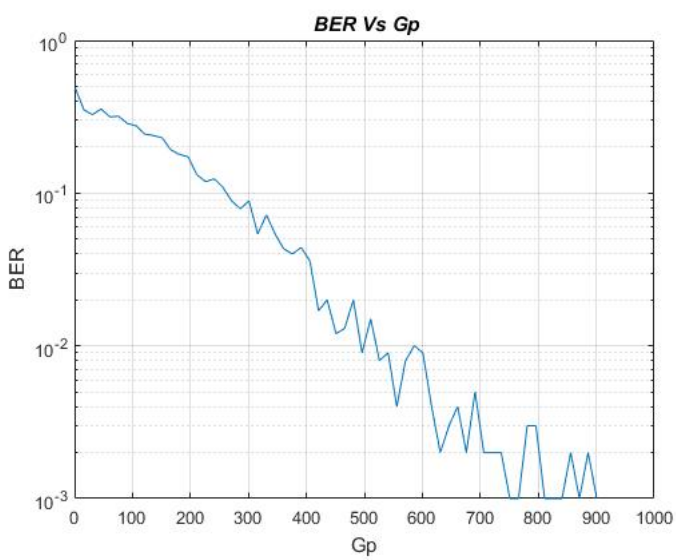
As one might expect, and the simulations graphs illustrates the higher the spreading factor of system the lower the bit error rate (BER). For instance, to reach a $BER \approx 10^{-2}$ a high spreading factor $G_p \approx 630$ is required when the frequency of the noise signal is $F_{noise} = 500\,\text{kHz}$ and $F_{noise} = 1\,\text{kHz}$ as sub-figures 5.1a, 5.1b illustrates. Interestingly, the required spreading factor $G_p = 600$ is slightly lower than the previous two cases for a $BER \approx 10^{-2}$ when the frequency of the noise signal is $F_{noise} = 1\,\text{MHz}$ as Figure 5.1 illustrates.



(a) $F_{noise} = 500\,\text{kHz}$, $SNR = -13.97\,\text{dB}$



(b) $F_{noise} = 1\,\text{kHz}$, $SNR = -13.97\,\text{dB}$



(c) $F_{noise} = 1\,\text{MHz}$, $SNR = -13.97\,\text{dB}$

Figure 5.1: BER as a function of the spreading factor G_p

5.1.2 BER Against Noise Frequency (F_{noise})

The second simulation run investigated system's performance (BER) as a function of the frequency of the noise signal F_{noise}. The signal-to-noise ratio in this run is same as the one from the previous run since neither the amplitude of the signal nor the amplitude of the noise is changed. Therefore $SNR_{dB} = -13.97\,\text{dB}$.

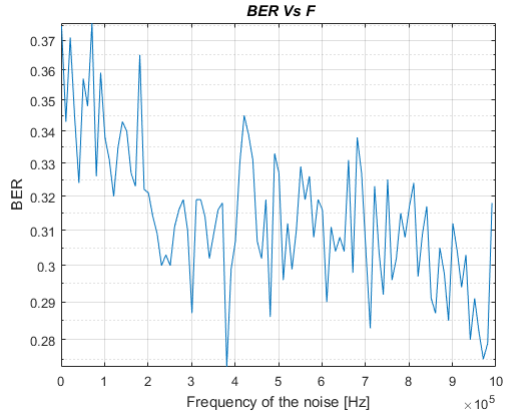
The set of frequencies investigated took the values between $F_{noise} = 1\,\text{kHz}$ to $F_{noise} = 1\,\text{MHz}$ with an incremental step of 10 kHz at each run. Moreover, the spreading factor of the systems G_p took the values of $G_p \in \{100, 200, 400, 500, 600\}$ in figures 5.2a, 5.2b, 5.2c, 5.2d, and 5.2e respectively.

For relatively low values of the processing gain of the system $G_p \in \{100 : 400\}$, the performance of the system degraded since the average bit error rate (BER) could not exceed $BER \approx 0.04$ which is in line with the results from the previous run since the average bit error rate is approximately $BER \approx 10^{-2}$ once the processing gain G_p is raised to $G_p = 600$.

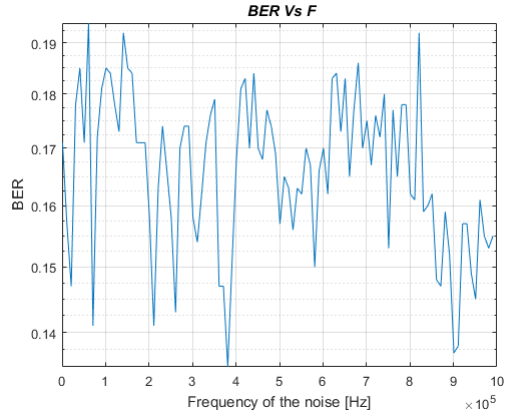
5.1.3 BER Against Signal-to-noise ratio (SNR)

The third simulation investigated the system's performance for a given set of interference amplitudes as illustrated in Figure 5.3. Similarly to first simulation, the processing gain G_p is assigned a different value at each run. Noise amplitudes took the values from $A_{noise} = 1\,\text{V}$ to $A_{noise} = 5\,\text{V}$ with incremental steps of 0.1 V. The BER is plotted as a function of the signal-to-noise ratio SNR.

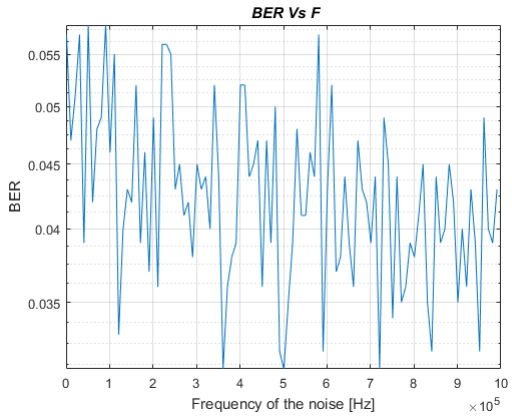
The spreading factor G_p took the values of $G_p \in \{100, 200, 300, 400, 500, 600\}$ in Figures 5.3a, 5.3b, 5.3c, 5.3d, 5.3e, and 5.3f respectively. As expected, the signal-to-noise ratio is improving by increasing the processing gain of the system G_p. As can be seen in Figure 5.3a, the processing gain is set to $G_p = 100$ which resulted in a low system performance. As the processing gain G_p is scaled up to a higher values, the signal-to-noise ratio is being improved. As can be seen in Figure 5.3e and Figure 5.3f, once the processing gain is $G_p > 400$, the signal-to-noise ratio (SNR) improved significantly compared to the simulations with lower processing gains than 400. Unfortunately, higher SNR values couldn't be extracted due to the high demanding computing power the simulations required.



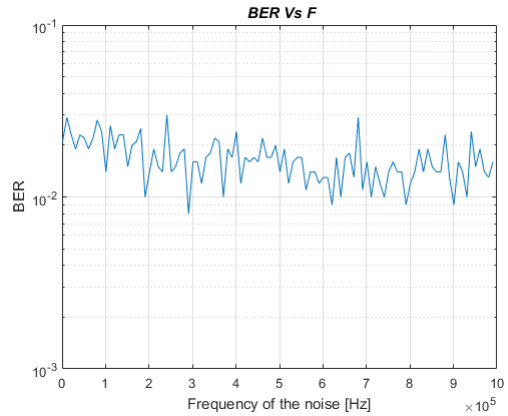
(a) $G_p = 100$



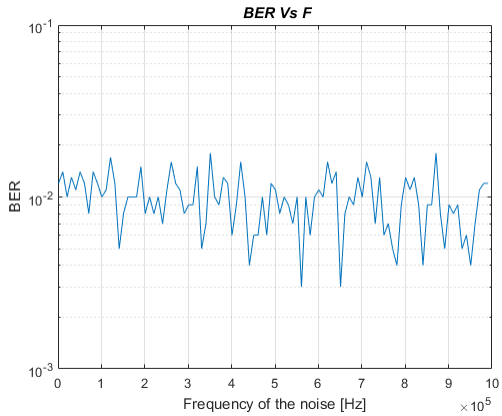
(b) $G_p = 200$



(c) $G_p = 400$

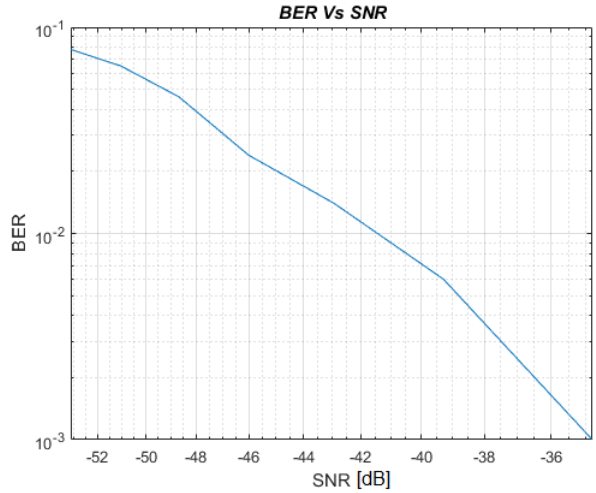


(d) $G_p = 500$

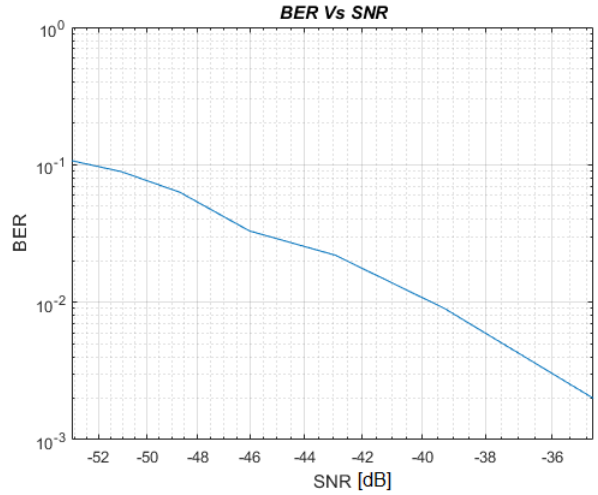


(e) $G_p = 600$

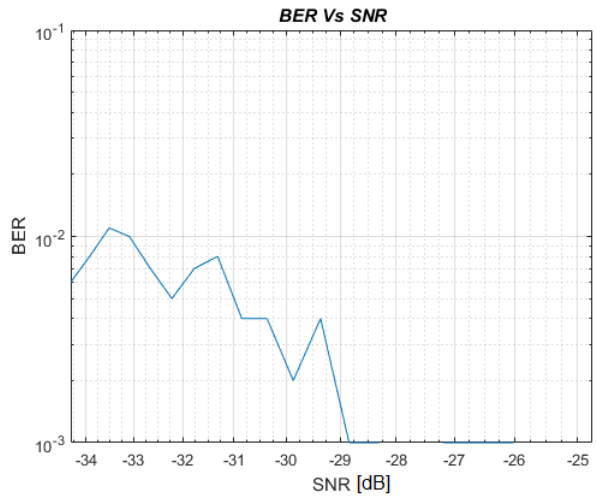
Figure 5.2: BER as a function of the interference frequency F_{noise}



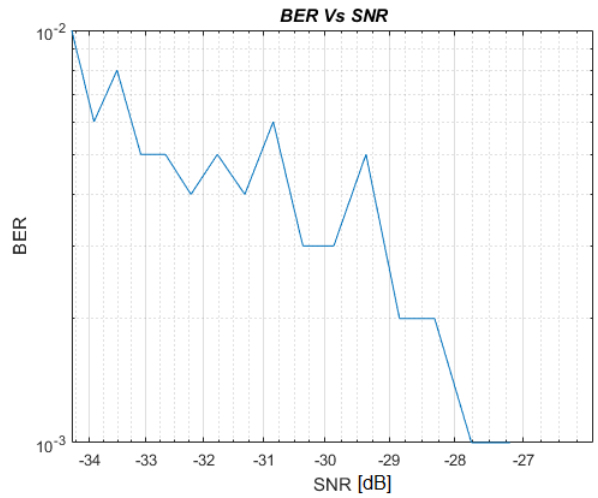
(a) $G_p = 100$



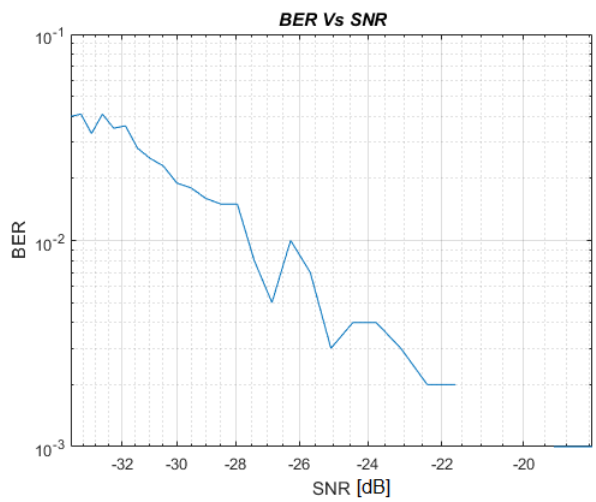
(b) $G_p = 200$



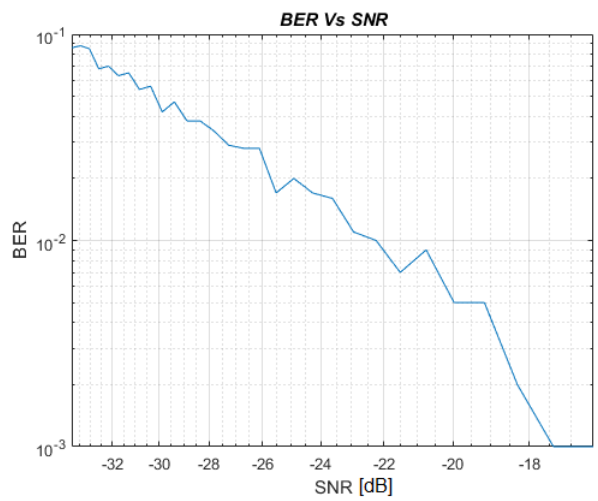
(c) $G_p = 300$



(d) $G_p = 400$



(e) $G_p = 500$



(f) $G_p = 600$

Figure 5.3: BER as a function of the signal-to-noise ratio SNR

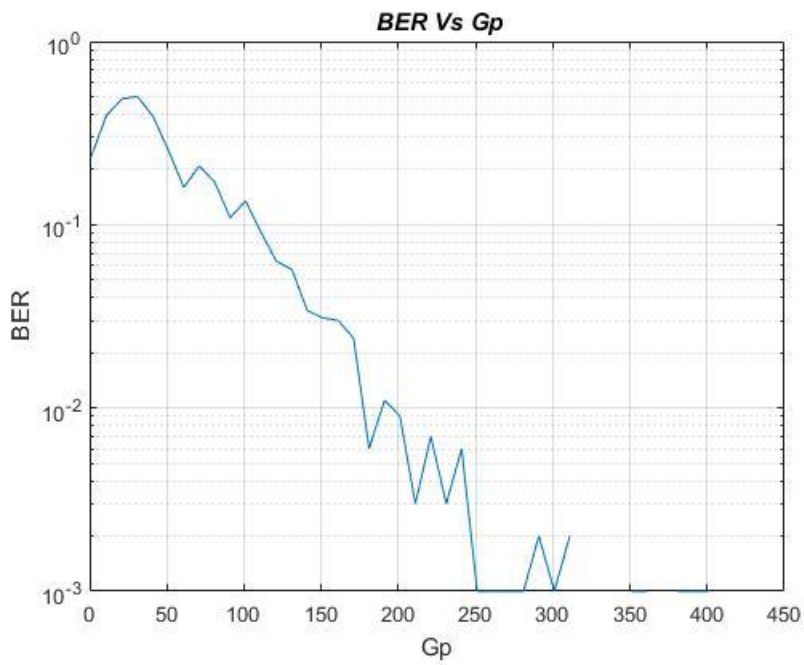


Figure 5.4: BER as a function of the spreading factor G_p, $SNR = -4.36\,\text{dB}$

5.1.4 BER Against Spreading Factor (G_p)

The fourth and final simulation utilized a real-world noise sample [1] rather than the square wave signal used in the previous simulations. Whereas all other parameters are the identical to the ones in the first simulation. The insights such signal give is illustrated in Figure 5.4. Modeling the noise signal as a square wave signal is considered to be a worst-case scenario noise signal for a baseband transceiver system such the one studied in this thesis. The simulation result illustrated in Figure 5.4 states that clearly. The processing gain of the system dropped almost to half $G_p = 250$ compared to all previous simulations in order to achieve an average bit error rate of $BER \approx 10^{-2}$. The signal-to-noise ratio can be calculated by finding the power of the noise sample provided which is in this case is the RMS of the noise sample, which is found to be $Noise_{RMS} = 1.6519$ Therefore, the signal-to-noise ratio is:

$$SNR_{dB} = 10\log_{10}\left[\left(\frac{1}{Noise_{RMS}}\right)^2\right] = -4.36\,\text{dB} \tag{5.4}$$

[1] The noise signal used here was provided by Atmel, Norway.

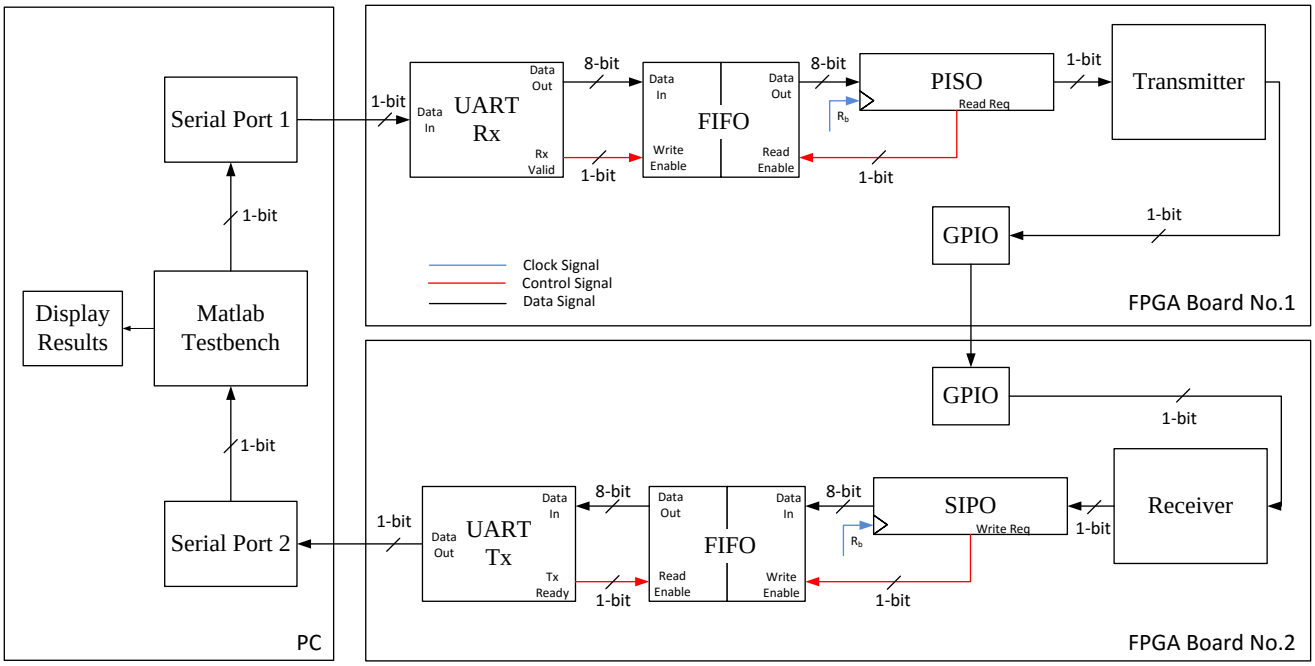


Figure 5.5: Demonstration Setup

5.2 Hardware Verification

Verifying logical correctness of the SystemVerilog modules was done by a SystemVerilog testbench which confirmed the behavioral correctness of the modules as described through out the sections in chapter four and summarized in Figures 4.4 and 4.9.

The real-time verification is done by comparing transmitted and received data packets. Those packets were MATLAB-generated and consisted of a preamble sequence (stream of zeros) and a payload consists of arbitrary bits similar the packet structure presented in chapter four in Figure 4.6. The length of the transmitted payload is fixed through out the verifications to $m = 1024$ bit.

The data packet is fed to the transmitter FPGA board through a serial port. The transmitter performs the spread-spectrum modulation as described in chapters three and four and transmit the data packet to the receiver through the GPIO pins located on the two FPGA boards. The receiver which is synthesized at the second FPGA board receives the transmitted packet and process it as described in chapters three and four. The estimated received packet at the receiver's end is sent back to MATLAB via a second serial connection. The whole setup of carrying out the

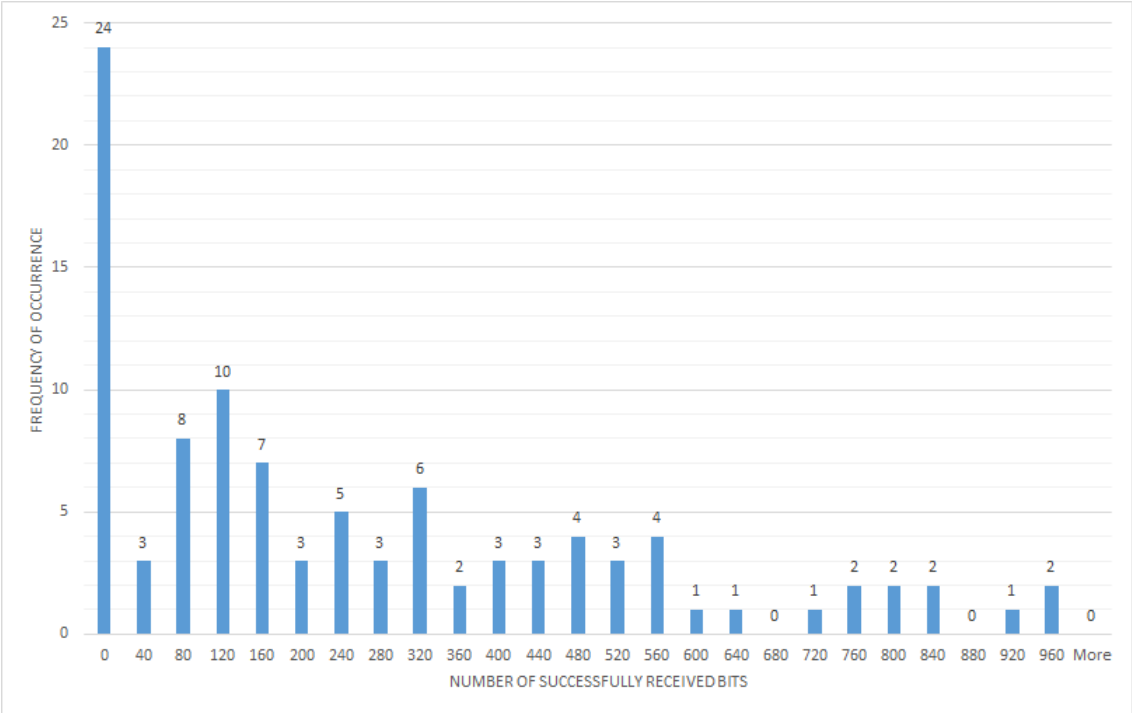


Figure 5.6: Histogram illustrating the number of successfully received bits, each bin has a width of 40 bits.

verification is illustrated in Figure 5.3, whereas the modules implemented are explained at the Appendix.

Since the receiver's synchronization with the transmitter is only based on the acquisition without the any tracking, the receiver will certainly fall out of synchronization at some point. Determining how reliable is the proposed architecture is based on how much useful data can be send and received during the time window in which both the transmitter and the receiver are in sync. A MATLAB testbench is created to determine the probability of the receiver falling out of sync. This is done over one hounded trails, at each trail the number of the successfully received packets is recorded.

The information collected over the trails are formated into a histogram chart as illustrated in Figure 5.6. As can be seen, two interesting points can be noticed from the histogram. The first bin, which illustrates how frequent the system falls out of sync even before sending any packet, represents a high probability of receiving no data at all almost 25% of the time. The second interesting observation is the 120-bit bin which has a higher frequency of occurrence compared to the other bins (except the zero bin of course). Clearly, the occurrence of successfully receiving

more bits than one hundred twenty bits means implicitly that for sure a hundred bits has been received successfully.

To get a better understanding of the system's performance, the collected data can be illustrated as a cumulative percentage graph as illustrated in Figure 5.7. This graph is a rough estimate of the packet error (PER) rate of the system. As conclude earlier from the histogram, the longer the transmitted packet, the lower the chance of receiving it. More precisely, transmitting a 120-bit packet will have a a 65% chance of being received successfully. Moreover, transmitting a packet with a length between 120-bits and 160-bits can have the chance of 50% of being successfully received.

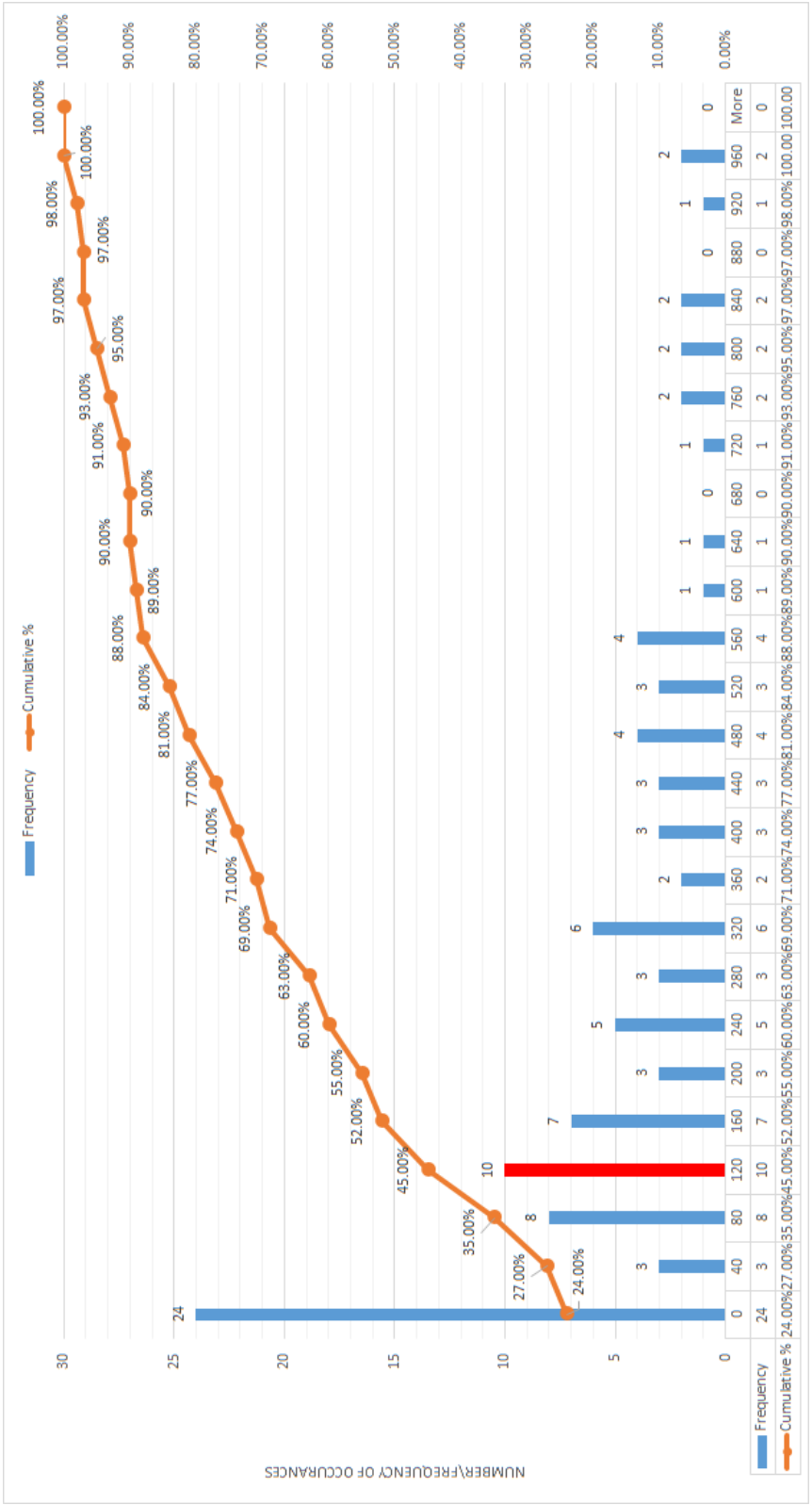


Figure 5.7: Cumulative percentage curve.

Chapter 6

Summary and Recommendations for Further Work

In this final chapter, the conclusions of the thesis are presented. Moreover, some possible suggetions for future work are provided.

6.1 Summary and Conclusions

In touch-sensitive application the problem associated with interfering noise will always be present. In this thesis, the effect of electromagnetic interference is studied for the case of a hand-held device with a capacitance touchscreen interface. Short-range wireless communication technologies such as the baseband transceiver presented in this thesis can be useful in mitigating noise effects in certain application scenarios. The proposed transceiver system utilized direct baseband transmission which is less power-hungry compared to other short-range which use high-frequency carrier for transmission.

The design and hardware implementation of the digital baseband transceiver for touchscreen devices is discussed in this thesis. The baseband modulation of the proposed architecture utilized direct sequence spread-spectrum modulation. The spread-spectrum modulation was the choice to explore as a solution for the presented problem mainly for its effectiveness in countering the noise found in the studied communication channel. Other modulation schemes are also a possibility to explore such as binary frequency shift keying due to its desired implan-

tation simplicity in this application. However, due to time shortage other modulation schemes could not be studied.

A simulation model of the proposed transceiver system is developed and putted under test to create an objective opinion regarding the effectiveness of spread-spectrum modulation in carrying the communications between the two transceiver's ends through the communication channel. The simulation model assumed a perfect synchronization between the transmitter and the receiver. Moreover, the simulation model verified the possibility of using a simple, one-bit comparator at the receiver side to successfully estimate the transmitted bits by the simplest architecture possible.

Turning the simulation models into a hardware implementation was the choice made for this thesis. The results from the simulations provided a starting point of implementing the proposed system in hardware. Moreover, the hardware implementation targeted a reference design of the modeled system. Developing the hardware implementation of the transceiver was done in two main steps. Firstly, building the hardware modules and verifying logical correctness through simulations. Secondly, synthesizing the designed hardware modules into two individual FPGA boards. The hardware modules consisted of a transmitter module and a receiver module. The transmitter module implemented a simple, 1-bit, direct sequence, spread-spectrum spreading circuit. Whereas the receiver, which is the focus of this thesis, implemented a de-spreading logic, a tracking logic, and an estimation logic.

6.2 Discussion

In this thesis, the system design and circuit implementing of a digital baseband transceiver for touchscreen devices is presented by two approaches. The first approach is to investigate the effectiveness of spread-spectrum modulation in transmitting the and receiving digital binary data over a noisy channel. This approach assumed an ideal case where the synchronization between the transmitter and the receiver is achieved. The second approach implemented the spread-spectrum modulation in hardware assuming non-noisy channel and a non-synchronized receiver.

The effectiveness in the context of the first approach is having a reasonable bit error rate.

However, the word "reasonable" doesn't apply in this setting due to the fact that different applications have different requirements. For instance, if a certain application has a specified channel that is limited to $SNR \approx -14\,\text{dB}$ and the application must have a $BER >= 10^{-3}$ or less, therefore the user should expect this architecture will require a spreading factor of $G_p = 600$. In an another application, the "reasonable" average bit error rate might be $BER \approx 10^{-1}$ for the same channel. Therefore, the user can expect a smaller circuit due to a smaller spreading factor however he/she will require a more robust receiver to compensate for the high error rate. This might come in the cost of the circuit but it is open to further research.

The second approach as mentioned earlier took another ideal case, which is a non-noisy channel. This approach investigated the implementation of spread-spectrum modulation with a synchronization logic. Due to implementing just an acquisition logic without a tracking logic, the synchronization will defiantly fall at some point in time after it has been established by the acquisition circuit. The real-time verification of the circuit investigated how reliable such a system. Once again, calling a communication system "reliable" is a relative term. The verification results showed that a data payload with an average length of $m = 120\,\text{bit}$ will allow the system to send and receive data with a 65% chance of receiving all of them. Such setup could be ideal for systems based on burst transmission. However, other systems with continuous transmission could make use of this setup but might need an error detection and correction logic. In this case, the trade-off between implementing such logic and the tracking logic is a topic to be studied regarding which direction is less costly and more reliable in terms of the target application.

The processing gain G_p plays a major role in both approaches. In the first approach, the higher the processing gain the better the average bit error rate. In other words, it directly affected the performance of the spread-spectrum modulation. In the second approach, the processing gain was fixed to a value of $G_p = 5$ to build a reference hardware module of the transceiver. In this case, the processing gain has no influence on acquisition circuit simply because the length of PN sequence has no association with the spreading factor of the system. However, the higher the processing gain, the better the average bit error rate as the simulations indicated. This needs to be checked in within the hardware implantation when applying a realistic, noisy channel rather than the ideal one used.

6.3 Recommendations for Further Work

As mentioned earlier, the design problem took two approaches. As a first recommendation for a future work, the two approaches can be combined in the framework of verifying the system under a realistic-channel along with a realistic spreading factor that is higher than five. Moreover, a tracking logic could be needed, which is can be verified by further research.

Those recommendation could fall under the category of short-term recommendations. For a medium-term recommendation, designing a link layer could be an option. The design presented in this thesis falls under a physical layer implantation, The link layer implementation could enhance the proposed communication system specially if it utilized error correction mechanisms.

Due to the fact that the proposed design in this thesis is application-specific, a long-term future work recommendation might not be in place. However, choosing a different modulation scheme rather than the spread-spectrum modulation such as binary shift keying could be an interesting approach to pursuit in the long-term.

Appendix A

FPGA Demonstrations

A.1 Introduction

In this appendix the implementation of the proposed digital baseband transceiver on the FPGA boards is discussed. Although the ModelSim tools is used for the behavioral verification of the transceiver in chapter four, a real-time verification must take place to verify the performance of the proposed architecture. The is done by using two Altera DE2-115 FPGA borads, one of which implemented the transmitter, the other implemented the receiver. The Quartus II tools are used for the synthesis, and place & route. Matlab is used to send and receive test data to and from the two FPGA boards. The boards provide 50 MHz clock signal, the various clocking signals R_b, R_c, OSR, DDR are provided by the parametrized tick generators discussed at chapter four.

A.2 Demonstration Modules

As can be seen in Figure 5.5, the demonstration setup consist of three parts

1. PC part which provided the Matlab software and two serial ports enabling the communications with FPGA boards.

2. First FPGA board implemented the proposed baseband transmitter and data transfer module.

3. Similarly, the second FPGA board implemented the proposed baseband receiver and data transfer modules.

A.2.1 The Transmitter FPGA Board

Firstly, the test packet data is generated in a MATLAB testbench. The test packet consists of a preamble sequence plus a data payload. The data payload contains for testing and validation perpuses 1024 bits. The test packet is send serially at a baud rate of 2400 bit s^{-1} via a serial connection controlled by MATLAB. The UART module handles the serial connection from the FPGA part. The code for this module is quite common and easy to understand. The UART module de-serialize the incoming test packet into a stream of words (8-bits each). A FIFO is a first in first out memory module. It is utilized due to the fact of having two clocking domain. The first domain is on the writing side of the FIFO in the transmitter board. The write side of the FIFO is clocked by the same clock of the UART module which is[1]:

$$wr_{clock} = UART_{clock} = \text{Baud Rate} \times 8 = 19\,200 \tag{A.1}$$

The UART module provided a R_x Valid signal which is used for enabling the write operation in the FIFO Whereas the read side of the FIFO must be clocked at the desired bit rate for the test which is set to $R_b = rd_{clock} = 4\,\text{kHz}$. Hence, the write side of the FIFO must be four time higher than the reading side to avoid data being overwritten:

$$\frac{\text{Clock rate on the fast FIFO side}}{\text{Clock rate on the slow FIFO side}} = \frac{19200}{4000} = 4 \tag{A.2}$$

The PISO module stands for parallel-in serial-out handles the task of feeding the transmitter module the queued test packets in the FIFO within the rate of $R_b = 4\,\text{kHz}$. It requests the data from the FIFO's read side by asserting the read enable signal.

The transmitter module which is explained in Chapters three and four performs its task of modulation and transmits the DSSS signal to the receiver module via a direct connection through the available GPIO connections on both boards.

[1] The UART module uses an oversampling of eight when it comes to receiving the serial data

A.2.2 The Receiver FPGA Board

The receiver receives the DSSS signal via one of the available GPIO ports. The receiver's output is fed to a serial-in parallel-out module (SIPO) which handles the task of de-serializing the estimated bits to an 8-bit words. Similar to the reasons mentioned in the transmitter board, the FIFO is required here as well. In this case, the read side of the FIFO is the slow part. The expected rate of the estimated bits (receiver's output) is $R_b = 4\,\text{kHz}$ Therefore, the $wr_{clock} = Rb = 4\,\text{kHz}$. the SIPO module is also responsible for asserting the write enable signal. Once that data are requested by the UART module, the T_x Read signal is asserted by the UART module which enables the reading logic on the FIFO. The UART module handles the serial communications between the receiver's board and the MATLAB testbench via a serial connection clocked at $2400\,\text{bit}\,\text{s}^{-1}$. The MATLAB tesbench reads the received data and compares it to the one originally send to the transmitter. The comparison results are discussed in chapter five.

Bibliography

[1] Silicon Labs: How to Minimize Touchscreen Electromagnetic Interference, 2012.

[2] J. Proakis and M. Salehi, Digital communications. Boston: McGraw-Hill, 2008.

[3] A. Goldsmith, Wireless communications. Cambridge: Cambridge University Press, 2005.

[4] Couch, Leon W, Digital and analog communication systems: Upper Saddle River, N.J., Pearson, 2013.

[5] Chaparro, Luis F, Signals and systems using MATLAB.

[6] Ingle, Vinay KProakis, John G, Digital signal processing using MATLAB, Stamford, Conn: Cengage Learning, 2012.

[7] Proakis, John GIngle, Vinay K, Student manual for digital signal processing with MATLAB, Upper Saddle River, NJ: Pearson & Prentice Hall, 2007.

[8] Oppenheim, Alan Verghese, George C, Signals, systems & inference.

[9] Ching-Hung Chiou, and Chao-Wang Huang, and Kuei-Ann Wen, and Mau-Lin Wu, A programmable pipelined digital differential matched filter for DSSS receiver, IEEE J. Select. Areas Commun, 2001.

[10] Ming-Luen Lieu,Tzi-Dar Chiueh, A low-power digital matched filter for direct-sequence spread-spectrum signal acquisition, IEEE J. Solid-State Circuits, 2001.