Radiation Tolerant Satellite Communication Modem

Moysis Tsamsakizoglou

Master of Science Thesis
Stockholm, Sweden 2012
Radiation Tolerant Satellite Communication Modem

June 25, 2012

Author:
Moysis Tsamsakizoglou

Academic Supervisor:
Nickolay Ivchenko

Industrial Supervisors:
Fredrik Bruhn
Jan Schulte

Master of Science Thesis
Stockholm, Sweden 2012
XR-EE-SPP 2012:006
Abstract

Low Earth Orbit (LEO) satellites due to their close distance to Earth have the advantage of low power communications and the propagation time of the transmitted data is negligible. However, these advantages don’t come without cost. The satellites need to reach very high speeds in order to manage to stay in orbit because the Earth’s gravitational field is pulling them down. These speeds can be up to approximately 7 km/s which makes the Doppler effect very intense. As a result, the communication systems of such satellites are more complex since they require proper circuitry that will have to deal with the Doppler effect in order to ensure a reliable and uninterrupted communication. An other challenge, the satellite communication systems face in space environments, is the effect of radiation which affects semiconductor devices and can have disastrous consequences in the functionality of communication equipments. This Master thesis analyzes and implements digital modulation schemes which are the core of modern digital communication systems nowadays. Throughout the design process the Doppler effect and the impact of radiation in semiconductors have been taken into account and proper solutions have been applied to solve them. Four phase modulations (BPSK, SDBPSK, QPSK, OQPSK) and two frequency modulations (AFSK, MSK) have been implemented which will be used to send Telemetry data from the satellite to the ground station on Earth. One phase demodulation (BPSK) has been also implemented that will receive Telecommand data from the ground station for the proper control of the satellite. The BPSK modulator and demodulator constitute together a complete satellite modem solution.
Acknowledgements

I would like to express my immense gratitude to Dr. Nickolay Ivchenko for his supervision and guidance while I was working with my thesis. Despite his very limited free time, he was always willing to support me.

Similarly, I would like to thank my industrial supervisors, Dr. Fredrik Bruhn and Jan Schulte, since this thesis has been a collaboration between KTH and ÅAC Microtec. They provided me with all the necessary tools for the successful completion of the thesis. During my thesis work I learned a lot about the exciting field of space technology and satellite communications and their contribution to that was significant.

The completion of this thesis finalizes my Master of Science studies. I feel, it will be an omission if I do not mention two people that helped me a lot make this dream come true. The first is Professor Christos Tzikas who supported me a lot during my bachelor studies, initiated my interest in digital hardware development and he was the first who suggested me continue further with my studies in a more advanced level. The other person I would like to thank is Professor Jari Mustajärvi who was my supervisor during my exchange studies in HAMK University of Applied Sciences, in Finland.

Also I would like to thank Theodor-Adrian Stana for the very long study hours we have shared throughout our studies in KTH and during our theses work in Alfvén Laboratory.

Last but not least I would like to thank my family for being supportive for these two years during my studies.

Moysis Tsamsakizoglou
Contents

1 Introduction 1
  1.1 Satellite Communications ......................................................... 1
    1.1.1 Types of Orbits ............................................................ 1
    1.1.2 Telemetry and Telecommand .............................................. 2
    1.1.3 CCSDS File Delivery Protocol (CFDP) ................................ 2
  1.2 Analog versus Digital Telecommunications .................................. 3
    1.2.1 Digital Modulations ....................................................... 3
    1.2.2 RF Front-Ends .............................................................. 4
  1.3 Need for Radiation Tolerant Systems ........................................ 5
  1.4 Master Thesis Goals ............................................................ 6

2 Background 7
  2.1 Radiation Tolerant Hardware .................................................. 7
    2.1.1 Flash based FPGAs ......................................................... 7
    2.1.2 nanoRTU 21X ................................................................. 8
    2.1.3 Radiation Tolerant Design Techniques ................................ 9
  2.2 Doppler Effect ................................................................. 10
  2.3 Digital Modulation ............................................................. 10
    2.3.1 Modulation Techniques .................................................... 11
      2.3.1.1 Amplitude Modulation ................................................. 11
      2.3.1.2 Frequency Modulation ............................................... 11
      2.3.1.3 Phase Modulation .................................................... 12
    2.3.2 Modulation Performance Metrics ...................................... 12
      2.3.2.1 Bandwidth Efficiency ............................................... 12
      2.3.2.2 Bit Error Rate (BER) ............................................... 13
    2.3.3 Modulation Schemes ....................................................... 13
      2.3.3.1 Binary PSK (BPSK) .................................................... 13
      2.3.3.2 Symmetrical Differential Binary PSK (SDBPSK) .................. 14
      2.3.3.3 Quadrature PSK (QPSK) ............................................... 14
      2.3.3.4 Offset Quadrature PSK (OQPSK) ..................................... 16
      2.3.3.5 Audio Frequency Shift Keying (AFSK) ................................ 17
      2.3.3.6 Minimum Shift Keying (MSK) ........................................ 18
  2.4 Digital Demodulation .......................................................... 19
    2.4.1 Types of Demodulation ................................................... 19
    2.4.2 BPSK Demodulation ......................................................... 19
    2.4.3 Carrier Recovery Techniques .......................................... 20
      2.4.3.1 Costas Loop .......................................................... 20
    2.4.4 Timing Recovery Techniques ........................................... 22
      2.4.4.1 Early-Late Gate Algorithm ........................................ 22
3 BPSK Modulator

3.1 BPSK Block Diagram

3.1.1 Carrier Generation

3.1.1.1 Quantization Effect on Spectrum

3.1.1.2 Look-Up Table

3.1.1.3 Carrier Generation State Machine

3.1.2 Inverter

3.1.3 DA Controller

3.1.4 PLL and Reset Block

3.2 Testing

3.2.1 UART Controller

3.2.2 Data Latch

3.2.3 Data State Machine

3.2.4 Hardware Results

3.3 Synthesis Results

4 SDBPSK Modulator

4.1 SDBPSK Block Diagram

4.1.1 Carrier Generation

4.1.2 Differential Encoding

4.1.2.1 Encoder

4.1.2.2 Delay

4.1.3 Hardware Results

4.1.4 Synthesis Results

5 QPSK & OQPSK Modulators

5.1 QPSK Block Diagram

5.1.1 Carrier Generation

5.1.2 I / Q Channels

5.1.3 Inverter

5.1.4 Adder

5.1.5 Testing

5.1.6 Hardware Results

5.1.7 Synthesis Results

5.2 OQPSK Modulation

5.2.1 Hardware Results

5.2.2 Synthesis Results

6 AFSK & MSK Modulators

6.1 AFSK Block Diagram

6.1.1 Carrier Generation

6.1.2 Frequency Selection

6.1.3 Hardware Results

6.1.4 Synthesis Results

6.2 MSK Block Diagram

6.2.1 Carrier Generation

6.2.2 Frequency Selection

6.2.3 Phase Control

6.2.4 Hardware Results

6.2.5 Synthesis Results
# List of Figures

1.1 Front-End transmitter .............................................. 4  
1.2 Front-End receiver ................................................. 4  
2.1 SRAM cell ............................................................. 8  
2.2 Flash cell ............................................................ 8  
2.3 nanoRTU ............................................................... 9  
2.4 Triple Module Redundancy .......................................... 9  
2.5 BPSK modulator ...................................................... 13  
2.6 SDBPSK modulator .................................................. 14  
2.7 QPSK modulator ...................................................... 15  
2.8 QPSK constellation diagram ....................................... 15  
2.9 OQPSK modulator ................................................... 16  
2.10 AFSK modulator .................................................... 17  
2.11 BPSK demodulator .................................................. 20  
2.12 Costas Loop .......................................................... 20  
2.13 Early-Late Gate algorithm ........................................ 22  
3.1 BPSK Block Diagram ................................................. 25  
3.2 64 Samples - 16 Quantization steps .............................. 26  
3.3 64 Samples - 64 Quantization steps ............................. 26  
3.4 256 Samples - 16 Quantization steps ............................ 27  
3.5 256 Samples - 256 Quantization steps ........................... 27  
3.6 Carrier generation state machine ................................. 27  
3.7 8-bit Full Adder structure ......................................... 29  
3.8 DA Controller state machine ...................................... 29  
3.9 Reset block ........................................................... 30  
3.10 BPSK bitstream generation ....................................... 31  
3.11 Data state machine ................................................ 31  
3.12 BPSK hardware results ............................................ 32  
4.1 SDBPSK Block Diagram .............................................. 35  
4.2 Encoder ............................................................... 36  
4.3 Encoder simulation .................................................. 36  
4.4 SDBPSK hardware results .......................................... 37  
5.1 QPSK Block Diagram ................................................. 40  
5.2 I / Q Channels block ............................................... 41  
5.3 QPSK bitstream generation ........................................ 42  
5.4 QPSK hardware results - I / Channels ........................... 43  
5.5 QPSK hardware results - Phase shifts .......................... 43  
5.6 Staggered I / Q Channels block ................................... 45  
5.7 OQPSK hardware results - I / Q Channels ..................... 45  
5.8 OQPSK hardware results - Phase shifts ........................ 46  
6.1 AFSK Block Diagram ................................................. 47
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2</td>
<td>AFSK hardware results</td>
<td>49</td>
</tr>
<tr>
<td>6.3</td>
<td>MSK Block Diagram</td>
<td>51</td>
</tr>
<tr>
<td>6.4</td>
<td>Data delay</td>
<td>52</td>
</tr>
<tr>
<td>6.5</td>
<td>Phase control</td>
<td>53</td>
</tr>
<tr>
<td>6.6</td>
<td>MSK hardware results</td>
<td>53</td>
</tr>
<tr>
<td>7.1</td>
<td>AD Controller state machine</td>
<td>57</td>
</tr>
<tr>
<td>7.2</td>
<td>Costas Loop Block Diagram</td>
<td>58</td>
</tr>
<tr>
<td>7.3</td>
<td>Phase Detector simulation results</td>
<td>59</td>
</tr>
<tr>
<td>7.4</td>
<td>Phase Detector simulation results - Frequency and Phase mismatch</td>
<td>59</td>
</tr>
<tr>
<td>7.5</td>
<td>FDA Tool settings</td>
<td>60</td>
</tr>
<tr>
<td>7.6</td>
<td>FIR Filter response</td>
<td>61</td>
</tr>
<tr>
<td>7.7</td>
<td>FIR Datapath analysis</td>
<td>62</td>
</tr>
<tr>
<td>7.8</td>
<td>Error in lock condition when NCO is set to 7.812KHz</td>
<td>63</td>
</tr>
<tr>
<td>7.9</td>
<td>Error in lock condition when NCO is set to 7.031KHz</td>
<td>64</td>
</tr>
<tr>
<td>7.10</td>
<td>Error in lock condition when NCO is set to 8.593KHz</td>
<td>65</td>
</tr>
<tr>
<td>7.11</td>
<td>Numerically controlled oscillator</td>
<td>65</td>
</tr>
<tr>
<td>7.12</td>
<td>Phase accumulator</td>
<td>66</td>
</tr>
<tr>
<td>7.13</td>
<td>AD Controller hardware results</td>
<td>67</td>
</tr>
<tr>
<td>7.14</td>
<td>Costas Loop hardware results</td>
<td>68</td>
</tr>
<tr>
<td>7.15</td>
<td>Early-Late Gate Block Diagram</td>
<td>69</td>
</tr>
<tr>
<td>7.16</td>
<td>Phase errors in symbol timing recovery</td>
<td>70</td>
</tr>
<tr>
<td>7.17</td>
<td>Early-Late Gate simulation results</td>
<td>70</td>
</tr>
<tr>
<td>7.18</td>
<td>Early-Late Gate hardware results</td>
<td>71</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Look-Up table values and inversion .............................................. 28
3.2 BPSK synthesis results ................................................................. 33
4.1 SDBPSK testing pattern ................................................................. 38
4.2 SDBPSK synthesis results ............................................................... 38
5.1 Symbol to phase mapping ............................................................... 42
5.2 QPSK synthesis results ................................................................. 44
5.3 OQPSK synthesis results ............................................................... 46
6.1 AFSK synthesis results ................................................................. 50
6.2 MSK synthesis results ................................................................. 54
7.1 Filter coefficients ............................................................... 62
7.2 Demodulator synthesis results ....................................................... 72
Chapter 1

Introduction

Satellites constitute a very special and complex field of space technology which should not be treated independently but as a part of a larger system. Undoubtedly, one important part of this system is the launching process which aims to set the spacecraft into an orbit around the Earth. According to the satellite’s mission, the orbit requirements may vary affecting also the launching process. Common missions include Meteorology, Military and Defence, Astronomy and Navigation. Even though the broad range of applications makes the complexity of every mission unique with each of them facing different challenges from the launching process to the spacecraft design, most of them share something in common. All of these applications require some type of communication with the Earth for the proper control of the spacecraft and the collection of data or other types of information. That makes satellite communications a major task with significant importance to the successful completion of a mission. This Master thesis will focus on satellite communications and the challenges which arise compared to conventional terrestrial communications.

1.1 Satellite Communications

A satellite communication system can be seen as two different segments: The Space segment and the Ground segment [1].

The Space segment contains the satellite or a satellite network and all the facilities on the Earth for their proper control which are called Telemetry Tracking and Command stations (TT&C) [1]. Their task is to monitor the correct operation of the satellite and all the crucial functions such as power management, ability to stay in the required orbit and many others which will guarantee the success of the mission.

The Ground segment contains all the end-users or stations on the Earth which communicate with the satellite. Some end-users use the satellite’s services in order to receive information only like navigation systems or television signals and some others in order to both transmit and receive like mobile communications.

1.1.1 Types of Orbits

The orbit requirements are determined in big proportion by the mission and the applications which the satellite aims to cover. There are generally several types of orbits which the satellite can be set to follow but not all of them are proper for communication purposes. In this paragraph we review some of those which are well suited for communications.

Geostationary Earth Orbit (GEO) is a circular orbit above the Earth’s equator which means that every satellite is set to a specific longitude according to the area that is required to cover and a latitude of 0° [2]. A feature that satellites, which are set to GEO orbit have, is that they look stationary (in the same position) when they are observed from the Earth. In fact this occurs because they are moving with the same speed the Earth is rotating. This makes satellites in GEO orbit ideally for telecommunications and point to point data transmission without facing problems related to the Doppler effect. The fact that their distance is more than 42,000 km [3] from ground makes them also have a very good coverage of the Earth’s surface. Three satellites placed in 120° around the Earth’s equator can cover most of the
Earth. Notable disadvantages of the GEO orbit satellites are their high costs, their inability to cover the polar parts of the Earth (all areas above 76° latitude due to the Earth’s geometry) and the propagation delay of the transmitted data due to the long distance to Earth.

Low Earth Orbit (LEO) satellites follow orbits much closer to the Earth. Most of the LEO orbit satellites are between 180 km to 2000 km [3] from the Earth’s surface which makes the propagation delay of the transmitted data negligible compared to GEO orbits and the required transmission power much lower. However, the coverage of LEO satellites is smaller and usually a network of satellites is used to cover large areas on the Earth’s surface. Due to the fact that LEO satellites are relatively close to earth, in order to stay in orbit and deal with the effect of the Earth’s gravity, they have to move at very high speeds which can be around 7.5 km/s [3]. Their tracking is essential and the complexity of their communication systems higher since the Doppler effect now is very intense. The International Space Station (ISS) is set to follow a LEO orbit.

Medium Earth Orbit (MEO) satellites follow orbits with a distance range of 2,000 km to 42,000 km to Earth, between the LEO and GEO orbits [3]. For that reason, their advantages and disadvantages are a combination of the ones that have been already mentioned. Very notable examples of MEO satellites are GPS, Glonass and Galileo navigation systems [4]. These systems use MEO orbits because a smaller number of satellites are required for global coverage compared to LEO orbits and at the same time the end-user’s transceiver can be more low power with smaller antennas compared to the ones that would be required in case of GEO satellites.

As a final remark in this section we can add that not all orbits are circular [5]. In circular GEOs for example, the satellite to Earth distance is stable. On the other side, MEO and LEO satellites sometimes are set to follow an elliptical orbit. As a result, in some point during the orbit the distance between the satellite and the Earth gets decreased and reaches a minimum which is called perigee. In that point the spacecraft’s speed is increased to overcome the gravity. In another point during the orbit the distance between the satellite and Earth is increased reaching a maximum called apogee. In that point the spacecraft’s speed is decreased because the effect of gravity is less. During apogee the Earth’s coverage is larger.

1.1.2 Telemetry and Telecommand

During the launching process the communications are limited to very critical functions like battery management and temperature monitoring in various systems. Once the spacecraft gets into the required orbit the communication systems are fully enabled and the interaction with the Telemetry Tracking and Command stations starts. The communication with the stations can be split in two categories according to the direction of the data.

Telemetry or downlink is the data transmission from the spacecraft to the Telemetry Tracking and Command stations down to Earth. The types of data that are carried through Telemetry can be categorized as following [2]:

- **Housekeeping data** contain information about the spacecraft. Through Telemetry all the functions and the subsystems of the spacecraft can be monitored for correct operation and maintenance.
- **Attitude data** come from accelerometers, gyroscopes or other sensors which are used to monitor the interaction of the spacecraft with the environment and its position in it.
- **Payload data** usually depend on the mission. In telecommunication or observation satellites the amount of data which need to be sent back to Earth may be considerably high so in these cases extra equipment is used for data transmission.

Telecommand or uplink is the data transmission from the Earth to the spacecraft. The control team on Earth based on the Telemetry data have the ability through Telecommand to fully control the spacecraft from simple operations to more complex ones.

1.1.3 CCSDS File Delivery Protocol (CFDP)

Even though with Telecommand the Telemetry Tracking and Command station being able to fully control the spacecraft, the complexity of the systems on-board makes it inefficient to be realized only at the bit
level. There is strong requirement for robust and reliable bulk transfers of data organized in files. In order to achieve this, the development of reliable communication protocols is essential. A protocol that serves that need is the CCSDS File Delivery Protocol (CFDP) [6] developed from the Consultative Committee for Space Data Systems and has been widely adopted. It can reliably create, handle and manipulate file transfers allowing remote management of on-board file systems. That allows to take the control of the spacecraft in a higher level of abstraction.

1.2 Analog versus Digital Telecommunications

The nature of space applications and the harsh environments where the telecommunications equipment is assigned to operate raise the need for reliable communication methods. Digital telecommunication methods dominate over the analog ones since they have a number of advantages. Below, the most important ones are presented.

- Digital communication methods are much less susceptible to nonlinearities, distortion and noise. Discrete signals are quantized and that makes them more immune to the effect of noise compared to signals which are continuous in time (Analog ones).

- In satellites the transmission bandwidth is limited and very expensive. It is common for many satellite telecommunication systems to use switching techniques to share their bandwidth with many end-users or ground stations. These switching techniques are much more efficient and low cost to implement with digital communication methods.

- Protocols such as CCSDS File Delivery Protocol, can only be realized in digital communications and they give great flexibility.

- In telecommunications and more specific in Military and Defence applications it is usually essential to encipher and decipher information in order to ensure a level of security. In digital communications complex encrypting algorithms are easier to be applied.

- Large number of the systems on the spacecraft are digital, like on board computers and other systems for motion control. Interfacing these systems to digital communication equipment is a direct, low cost and more reliable solution.

1.2.1 Digital Modulations

Digital data which are sampled in the different subsystems of the spacecraft in order to be sent to the base station down to Earth through Telemetry, need to be properly processed from the communication systems for transmission. This process is called digital modulation. The need for digital modulation arises from the fact that the digital data can not be sent directly to the antenna for transmission. These data are usually low frequency baseband signals which normally would require antennas with unrealistic dimensions to be transmitted. To get a good idea of the problem, we can take as an example a normal dipole antenna which in order to transmit a signal with frequency $f$, needs to be $\frac{\lambda}{2}$ meters long where $\lambda$ is the wavelength of the transmitted signal. For a 10KHz signal, a 15km long antenna would be appropriate for transmission which obviously is not practical. On that point, a question that would naturally arise is: “What would be the case in high frequency baseband signals?”. This looks like an ideal scenario because more data could be transferred and smaller antennas would be required. However, in practice it is not a feasible solution because high frequency digital signals have large spectral content which in most of the cases doesn’t comply to the strict regulations.

Purpose of Digital modulation is to shift the low frequency baseband signal in the spectrum to a higher frequency for transmission, keeping at the same time the bandwidth requirements low. In order to achieve this a high frequency sinusoidal signal, called carrier, is used to modulate the digital data and “carry” them. The advantage of modulation is that since the carrier is a tone, it has zero spectral content and it is used to shift the spectrum of the digital signal to higher frequencies. As a result, low bandwidth requirements can be achieved since the transmitted signal’s bandwidth is close to the baseband’s and at the same time small and efficient antennas can be used.
On the other side the digital demodulation is assigned with the task to reject the carrier signal and bring the digital information back to baseband frequency.

### 1.2.2 RF Front-Ends

Digital modulators convert the digital Telemetry data to a sinusoidal signal of higher frequency but usually that frequency is not the transmitted one. It is an Intermediate Frequency (IF) which needs to be up-converted for transmission. Figure 1.1 shows a diagram of a typical Front-End transmitter. The IF frequency is up-converted in the mixer with a high frequency produced from the Local Oscillator (LO). After Up-Mixing sometimes a band-pass filter is used to remove any unwanted frequency components. Next, the power amplifier whose output impedance should be properly matched with the antenna’s, amplifies the signal which is send to the antenna for transmission.

![Figure 1.1: Front-End transmitter](image)

On the other side, the Telecommand data in the form of RF waves which propagate through the space environment arrive to the spacecraft’s antenna. In the antenna, these RF waves are converted to a signal which is amplified by a Low Noise Amplifier (LNA). A band-pass filter is necessary before the LNA in order to pass only the necessary RF Frequency for amplification. After the amplifier, a down-mixer converts the signal to baseband using a frequency produced from a Local Oscillator (LO). A low-pass filter is used after down-mixing to remove any high frequency components.

![Figure 1.2: Front-End receiver](image)

In that point, it is good to mention that there are two main receiver techniques [7]. The first one is called Direct-Conversion Receiver (DCR) or Homodyne Receiver or zero-IF Receiver. In DCR the Local Oscillator has the same frequency as the received RF signal. From signal theory we know that signal multiplications have as a result two frequency components. That means after down-mixing the two resulting frequencies are

\[ f_{\text{high}} = f_{RF} + f_{LO} \]  

(1.1)
The $f_{\text{high}}$ is removed from the low-pass filter but the $f_{\text{low}}$ is almost zero. As a result the Telecommand data can be directly retrieved back with a DCR.

The other receiver technique is called superheterodyne receiver. In this technique the Local Oscillator frequency is different from the incoming RF signal and usually smaller. From (1.2), it is obvious that an Intermediate Frequency (IF) is retrieved back and needs further down-mixing in a second stage to get the baseband signal.

### 1.3 Need for Radiation Tolerant Systems

Deep space environment creates many challenges on the design of communication equipments and electronic systems in general since radiation can have a significant impact on the correct functionality of semiconductor devices and unwanted effects can cause serious malfunctions in the system. These unwanted effects of radiation are called Single Event Effects and they can be split in three categories depending on the type of failure they create [8]:

- **Single Event Upsets** occur when radiation has an effect to overcharge the depletion region of a $p$-$n$ junction which can cause voltage transients that are able to change the state of a register or a memory element in the design [9]. These errors are not permanent since only the element’s data are affected and if new data are written it will come back to correct operation.

- **Single Event Interrupts** are more serious and can cause permanent malfunction. Reset or device re-configuration is necessary for correct operation in that case. They are a special type of Single Event Upsets with the difference that configuration or memory cells are now affected from radiation. These errors lead to device malfunction and they can affect its complete functionality.

- **Single Event Transients** mainly refer to radiation effects on combinational logic circuitry. The effect of radiation on combinational circuits appears as instant voltage transients with very small duration since there is not any storage capability. They can create temporal erroneous data but their very small duration makes them not a serious concern.

**Single Event Effects** constitute a very important problem in satellite communications and the exact causes which are responsible for their existence need to be understood.

The first cause for Single Event Upsets is due to Atmospheric Neutrons which are created when high energy subatomic particles from Sun or deep space hit the Earth’s atmosphere. As a result of this interaction high energy neutrons arise. When these neutrons hit silicon atoms, heavy ions are created which can create transient voltages on the gate of CMOS transistors which consist the basic construction element for SRAM cells in current FPGA technologies. These transient voltages can lead to the state transition of memory cells or logic elements in FPGA devices. It is important to note here that Single Events based on Atmospheric Neutrons do not only exist in space. They exist also on Earth and they are highly considered in safety critical applications like medical and communications equipment. In space environments the problem is just more intense which makes the design of the communications equipment more complex.

The second cause for Single Events is due to Alpha particles which are emitted due to radioactive isotopes in the packages of the semiconductor devices. The most common trace impurities in packaging materials are Uranium $^{238/235}U$ and Thorium $^{232}Th$ and they emit particles as long as they decay [9]. These particles travel through the material and lose kinetic energy [8] but they achieve to leave heavy ions on their path which can affect the state of logic elements.

In communication equipment where FPGA devices are used for hardware development, two types of errors can be created from Single Event Effects [10]. The first are called Soft errors and occur when logic elements or registers momentary transit to another state which instantly can have some unpredictable effect in the functionality. The other types of errors are more serious and they are called Firm errors. We
call \textit{Firm errors} the unwanted effects that occur in the configuration data or the interconnection matrix of the FPGA device and these errors can have a disastrous effect on the functionality of the hardware because they are very difficult to be detected.

It is obvious that the need for radiation tolerant systems in space applications is critical and proper care should be taken during the design process. As mentioned in a previous section, through \textit{Telecommand} the control team in the ground station can take full control of the spacecraft. A temporary or permanent malfunction on the spacecraft receivers can have as a result to lose complete control of it. Considering also the cost and the importance of many missions, radiation tolerant techniques and proper hardware should be used to avoid these unwanted effects.

1.4 Master Thesis Goals

This Master thesis focuses on the communication systems used for \textit{Telemetry} and \textit{Telecommand} in LEO Satellites. The result of this work will be part of the \textit{Space segment} and will consist of the communication system of the spacecraft. The aim of the thesis is to study and create a solid background for the digital modulation schemes which are most commonly used in LEO satellites and find efficient ways to implement them on hardware using radiation tolerant flash based FPGAs. The expected outcome of the thesis can be summarized as following:

- A list of digital modulations will be implemented that are able to accept telemetry data and modulate them properly in order to feed them in a later stage to the RF Fron-Ends and then to the antenna. The modulations chosen are two frequency modulations (AFSK and MSK) and four phase modulations (BPSK, SDBPSK, QPSK and OQPSK). All of them are widely used in satellite communications.

- A BPSK demodulator will be also implemented. The BPSK demodulator together with the BPSK modulator will make up a complete modem solution that will be able to receive and transmit \textit{Telecommand} and \textit{Telemetry} data respectively. The implemented modem will set the base for future development of the CCSDS File Delivery Protocol on top of it.

- Given specifications for \textit{Telemetry} data require a bitrate of at least 4800 b/s and for \textit{Telecommand} 1200 b/s.

- The fact that the thesis aims for LEO satellites makes it necessary to deal properly with the Doppler effect in the demodulator’s design. In terrestrial applications this effect is negligible. However, in satellite applications it may result in completely losing the spacecraft. For that reason, proper tracking loops will be implemented in the demodulator that will allow to keep continuously track of the \textit{Telecommand} frequency variations.

- The hardware implementations will be done on radiation tolerant hardware using also radiation tolerant design techniques. That will make the communication system proper for space applications and tolerant to Single Event Upsets.
Chapter 2

Background

In this chapter the main topics of the thesis work are explained in detail in order to give the reader the proper theoretical background that will allow him to follow later the implementation details in the next chapters. In the first section, the solutions which exist for radiation tolerant hardware are presented focusing on FPGA devices which are the main platform for digital hardware development nowadays. The hardware tools that have been provided by ÅAC Microtec for testing the implemented design are presented, highlighting their features which make them a proper choice for aerospace applications. Design techniques which are used to prevent Single Event Upsets are also described. The second section refers to the basic principles of the Doppler effect which has been taken into consideration during the development of the demodulator. In the third section, the principles of the modulation techniques are described paying extra attention on six modulation schemes which are used in the thesis. In the fourth and last section the demodulator is described and solutions how solve the Doppler effect issues are proposed.

2.1 Radiation Tolerant Hardware

The first and most important step for reliable and robust satellite communication systems is the proper choice of the hardware components since semiconductors generally suffer from radiation in space environments. Field Programmable Gate Arrays (FPGAs) are the best choice for digital hardware development since they combine good performance with the advantage of re-programmability. However, not all FPGA technologies are proper for space environments. The understanding of radiation effects in each technology helps for the proper device selection.

2.1.1 Flash based FPGAs

The two dominant FPGA technologies today are the SRAM FPGAs which use as a basic switch or storage element SRAM cells and Flash FPGAs which use Flash cells. There is also a third technology called Antifuse-based FPGAs which offer good immunity against radiation but lack the advantage of re-programmability. In the forthcoming paragraphs we analyze the structure of SRAM and Flash based FPGAs to understand their radiation immunity.

In SRAM based FPGAs, a cell is made up from six transistors as can be seen in figure 2.1. The four transistors in the middle are two cross-coupled inverters. They are connected in a proper way so that they can never have their outputs in the same logic level. When the Write Line is 0 the SRAM cell is storing its value to the cross-coupled inverters. When it turns to 1 the value stored is released to the Bit lines. In the same way the Bit lines can be used to write a value to the cell. As technology evolves and fabrication processes are shrinking in nanometers the operation voltages are also decreasing. That means the SRAM technologies which are powered with 0.8, 1.2 or 2.4 volts, have cells which can be turned on and off with smaller voltages as well. That makes SRAM cell technologies sensitive to voltage transients which are created when ions hit the silicon atoms. In the SRAM cell of figure 2.1 if an ion hits one of the cross-coupled transistors it can change the stored value.

Flash based devices use Flash cells which have a different structure as they can be seen in figure 2.2. The stored value now is on the Floating Gate. The Word Line is separated from the Floating Gate with
a thin layer of oxide which has very high impedance. That means that in order to change the stored value in the Floating Gate, a high voltage needs to be applied through the Word Line.

In Actel Flash FPGAs for example this value is $\pm 17.5\text{V}$. The value stored in the Floating Gate controls the switching CMOS transistor which turns on and off accordingly. It can be easily understood that transistors in Flash cells need a much larger charge to change state compared to SRAM cells. That makes them more radiation tolerant since the small transient voltages created from ions can not easily affect them.

Another feature of Flash technology FPGAs is their very low power consumption which in satellite applications is very important. From figures 2.1 and 2.2 it is visible that a Flash cell uses fewer transistors compared to a SRAM cell.

The nanoRTU 21X hardware is suitable for aerospace applications since all the components on board are selected according to the European Space Agency (ESA) ECSS standards [11]. It has been designed to support very low power, radiation tolerant processor based systems to control the payload in the spacecraft.

The nanoRTU is equipped with a Flash technology FPGA from the ProASIC3 series product portfolio of ACTEL [12]. The FPGA device support designs up to 600k system gates and due to its very low power capabilities and radiation tolerant Flash technology is an ideal choice for such applications. The nanoRTU comes also with two 12-bit Digital to Analog converters and one 12-bit Analog to Digital converter on board. The DAC will be used to convert the digitally modulated signal in analog in order to be sent in the RF Front-End in a later stage and the 12-bit ADC will be used to receive the modulated signal and convert it to digital for data recovery and baseband processing.
2.1.3 Radiation Tolerant Design Techniques

The choice of a Flash technology FPGA is a very good solution and sets the base for a radiation tolerant design but it is not the only thing that the designer can do. There are proper design techniques which also help in that direction and with the help of existing hardware synthesis tools achieve more radiation immune systems.

The design tools used during the thesis work are the Libero v9.1 suite from Actel which comes with the Synplify synthesis tool from Synopsys. The Synplify tool which takes advantage of the underlying FPGA existing structure automates the process of radiation tolerant designs offering three methods to choose from. The one that has been used is the Triple Module Redundancy method (TMR) [13].

In TMR method every register in the design is replaced with three registers and a majority voter. A register replacement for the TMR method can be seen in figure 2.4. Assuming that an ion hits one of the three registers which changes its state, the other two registers will have the correct value. A majority voter propagates the value which most of these three registers have. This method provides a very good protection over Single Event Upsets but needs a lot of hardware resources which in practical designs can even lead to twice the consumed resources. Except from that, the majority voter adds an extra delay in the data path which should be verified with timing analysis that it does not affect the functionality of the design.
with very small duration and they can hardly have serious effect in functionality.

The second method is based on that concept. All the registers in the design are replaced with combinatorial cells with feedback. That makes the created storage element more radiation immune. The third and last method realize the TMR implementation but with the difference that here the three registers are replaced with combinatorial cells with feedback. It is a combination of the first and the second method. However, it requires a lot of hardware resources and that makes it not applicable to all designs.

2.2 Doppler Effect

Doppler effect is very important and needs to be taken into consideration in telecommunications when there is relative motion between the transmitter and the receiver. This motion has as a result the frequency transmitted to arrive different on the receiver. This could lead to serious problems in satellite communications and that raises the need for proper tracking circuitry that will allow the receiver to accept and demodulate the incoming *Telecommand* signal that will lie between a frequency range \((f_{\text{receding}}, f_{\text{approaching}})\). In order to design that loop a proper understanding of the phenomenon itself is necessary and the frequency range should be specified.

Having a stationary transmitter which transmits a signal with frequency \(f_0\) and a receiver which is moving relative to it with a speed \(v\), the doppler formula for light waves is given from [14]

\[
\lambda_{\text{receiver}} = \lambda_0 \sqrt{\frac{1 - \frac{v}{c}}{1 + \frac{v}{c}}} \tag{2.1}
\]

where \(\lambda_{\text{receiver}}\) is the wavelength of the signal which arrives to the receiver and \(c\) is the speed of light. To derive the (2.1) as a function of frequency, using the

\[
\lambda = \frac{c}{f} \tag{2.2}
\]

we can rewrite it as

\[
f_{\text{receiver}} = f_0 \sqrt{\frac{1 + \frac{v}{c}}{1 - \frac{v}{c}}} \tag{2.3}
\]

When the receiver is approaching the transmitter, the relative speed \(v\) is positive which means that it sees a higher frequency than the transmitted one. In the other case where the receiver is receding from the transmitter, the relative speed \(v\) is negative and a signal with lower frequency arrives to the receiver.

Knowing the transmitter’s frequency and the speed of the spacecraft, with the equation (2.3), we can define the frequency range that the demodulator should be able to keep track in order not to lose the communication with the Telemetry Tracking and Command stations. This range is the \((f_{\text{receding}}, f_{\text{approaching}})\) where \(f_{\text{receding}} < f_0 < f_{\text{approaching}}\). It becomes clear why the problem is so intense in LEO satellites. The higher the relative speed between the receiver and the transmitter, the wider this range is.

In our implementation we will define the doppler range as \(\pm 10\%\) of the expected incoming modulated signal to our demodulator. This percentage has been given as a requirement and has been mainly extracted empirically in order to demonstrate the demodulator’s capabilities. In order to specify accurately the actual doppler range in the demodulator’s input, the speed of the spacecraft and the receiver Front-End specifications are required which is out of the scope of this thesis.

2.3 Digital Modulation

The process of converting digital information into a signal suitable to be transmitted through a medium is called digital modulation and can be splitted into two main categories, baseband and broadband. In baseband modulation the data are represented with proper pulse waveforms and transmitted without any frequency transformation. The lack of frequency transformation in baseband modulation limits the
transmission of these signals through a wire because otherwise antennas with unrealistic dimensions would be required to transfer information through air. In telecommunications where information needs to be transferred in long distances broadband modulation is used. The low frequency digital information is transformed to a high frequency signal with the use of a sinusoidal high frequency signal called carrier. Since the goal of this thesis is satellite communications in this section we focus only on broadband modulation covering the basic principles and the required theory the reader needs to follow the implementation details which are presented in the following chapters.

2.3.1 Modulation Techniques

In broadband digital modulation the data are usually a bitstream which is modulated onto a sinusoidal signal called carrier which has the form

\[ f_c(t) = A_c \cos(2\pi f_c t + \phi_0) \]  

(2.4)

To completely describe a sinusoidal signal three parameters need to be specified. Those are the amplitude \( A \), the angular frequency \( \omega \) \((2\pi f)\) and the initial phase \( \phi_0 \). The data bitstream changes one of these three parameters of the carrier, producing the modulated signal. According to which parameter is modified we can distinguish three modulation techniques which are widely used.

2.3.1.1 Amplitude Modulation

The variation of the carrier’s amplitude from the data bitstream is called amplitude modulation (AM). In telecommunications theory AM is described as a non-constant envelope modulation technique. Having a data signal \( p(t) \) and a carrier like (2.4) the modulated output signal will have the form

\[ s(t) = A_c p(t) \cos(2\pi f_c t + \phi_0) \]  

(2.5)

The modulated signal will have the same frequency as the carrier but its amplitude will change according to the logic level of the bitstream in every time instant. The digital modulation based on this basic principle is called ASK (Amplitude Shift Keying). In cases where the data logic level set is \( \{0,1\} \) we have a special form of ASK modulation known as OOK (On-off Keying). In OOK the presence of the carrier means a binary one and the absence of it a binary zero.

Amplitude modulation used to be a dominant choice in the past mostly because it is based on a simple concept which is easy and low cost to implement. However, the fact that information is hidden on the amplitude of the modulated signal, makes AM prone to high levels of noise.

2.3.1.2 Frequency Modulation

The part \( 2\pi f_c t + \phi_0 \) of the carrier is called angle. The techniques where the bitstream changes the angle are called angle modulations. Frequency modulation is such an angle modulation where the data bitstream changes the angle frequency \( \omega \) of the carrier signal. The modulated signal in such case takes the form

\[ s(t) = A_c \cos(2\pi (f_c + \Delta f p(t)) t + \phi_0) \]  

(2.6)

The term \( \Delta f \) is called frequency deviation and represents how the carrier’s frequency changes in respect to the data bitstream. One popular frequency modulation is FSK (Frequency Shift Keying). In FSK the data bitstream takes values from the logic level set \( \{1,-1\} \) and the modulated signal can be expressed as a combination of two signals with different frequencies.

One question that naturally arises is what the value of \( \Delta f \) should be. It is obvious that the larger the deviation frequency the bigger will be the difference between the frequencies for the two logic levels. The modulation index \( \langle m \rangle \) gives a measure of \( \Delta f \) and is defined as
\[ m = \frac{\Delta f}{f_{\text{bitstream}}} \] (2.7)

According to the value of \( m \), frequency modulations can be split in two categories. When \( m \ll 1 \), then in that case we have narrowband frequency modulation. In the other case where \( m \gg 1 \) we have wideband frequency modulation. Usually the choice of \( m \) depends on the medium or any bandwidth restrictions. However, there are some types of frequency modulation that require a specific value of \( m \). For example, Sunde’s FSK modulation requires that the space between \( f_c + \Delta f \) and \( f_c - \Delta f \) is exactly the same as the symbol rate. That leads to a modulation with \( m = 1 \). As we will see also in a later section, MSK is another special case of frequency modulation with a requirement of \( m = 0.5 \).

### 2.3.1.3 Phase Modulation

Phase modulation is another type of angle modulation where the data bitstream changes the phase of the carrier and the modulated signal can be expressed as

\[ s(t) = A_c \cos(2\pi f_c t + K_p p(t)) \] (2.8)

The factor \( K_p \) is called phase sensitivity and shows how the phase of the carrier signal changes with respect to the bitstream. The amplitude and the frequency of the modulated signal are the same with the carrier’s. The phase \( \phi_i = K_p p(t) \) now contains the information data.

### 2.3.2 Modulation Performance Metrics

There are three main factors that influence the choice of a modulation scheme. These are the Bandwidth efficiency, the Bit Error Rate (BER) and the system complexity.

#### 2.3.2.1 Bandwidth Efficiency

In (2.8), if the information bitstream \( p(t) \) takes values from the logic level set \( \{1, 0\} \) the modulated signal will have the same frequency as the carrier but it will switch phase between \( K_p \) and 0. In case \( K_p = \pi \) rad then we have the BPSK modulation scheme. One important thing to consider is that until now we mentioned that \( p(t) \) takes values from a two level logic set. That means each time instant only one bit of information is transmitted. The need arises for a term called Bandwidth efficiency, which describes the number of bits per second that can be transmitted per Hertz of channel bandwidth. The amplitude modulation scheme OOK and the frequency modulation scheme FSK which have been mentioned above are also examples of modulations where only 1 bit of information is transmitted each time instant.

The importance of the Bandwidth efficiency concept in modulations like BPSK can be more obvious if it is realised in comparison with the Quadrature Phase Shift Keying (QPSK) scheme which is able to map the value of two bits in a specific phase shift. A modulation with this characteristic can achieve twice bandwidth efficiency than BPSK. Given a specific amount of data and the fact that they require the same time to be transferred in both schemes, QPSK requires half the bandwidth of BPSK. This is a critical factor when selecting a modulation scheme and becomes even more important as the bandwidth of the transmitted data increases.

The above can also be interpreted in a different way. Given a specific bandwidth, QPSK can transfer the same amount of data twice faster than BPSK. However, in reality most of the modulations aim to reduce the bandwidth.

In telecommunications theory the data information transmitted every time instant is called symbol. It can be precisely said that symbol is the unit of transmission and bit is the unit of information. The symbol’s transmission speed is measured in bauds per second and the transmission speed of bits in bits per second. For some modulation schemes like BPSK, OOK and FSK the baud rate is equal to the bit rate since the symbol is equal to one bit.
2.3.2.2 Bit Error Rate (BER)

Bandwidth efficiency is a critical factor when choosing a modulation scheme. However, this does not come without cost. Increasing the bits per symbol number makes the detection in the receiver harder and increases the possibility of error in the data information retrieval. For example, in the 16PSK modulation scheme the symbol is consisted of four bits. Compared to BPSK, the receiver should be able to detect sixteen distinct phase shifts. That of course increases the complexity of the system and the error probability. That raises the need for a metric called Bit Error Rate (BER) which shows the number of bit errors over a total number of sent bits in a specific time period.

2.3.3 Modulation Schemes

2.3.3.1 Binary PSK (BPSK)

Binary phase shift keying is the simplest type of phase modulation. The fact that the symbol to be transmitted is 1 bit in BPSK, makes the modulated signal to switch between two distinct phases [15]. For that reason the phase sensitivity \( K_p \) is chosen to be \( \pi \) rad. The data bitstream changes the phase of the carrier according to the logic level of the transmitted bit. The equation (2.8) can be transformed for BPSK as in

\[
s(t) = A_c \cos(2\pi f_c t + \pi p(t)) \quad (2.9)
\]

We can express BPSK as a combination of two sinusoidal signals with the same frequency but a difference of \( \pi \) rads in phase. These signals are called antipodal. Even though from (2.9) is derived that a logic '0' is mapped to 0 phase shift and a logic '1' to \( \pi \) rads phase shift, in practical systems usually it is the opposite because a cosine wave oscillator is used for the generation of the carrier signal and an arrival of a bit with logic '1' is represented with no phase shift. However, both are correct without any effect on bandwidth efficiency or bit error rate. Summarizing the above, the formula for BPSK can be expressed as

\[
s(t) = \begin{cases} 
A_c \cos(2\pi f_c t) & \text{for } 1 \\
A_c \cos(2\pi f_c t + \pi) & \text{for } 0 
\end{cases} \quad (2.10)
\]

The fact that the signals are antipodal allows BPSK modulation to be realised as a simple multiplication of the cosine carrier signal with the information bitstream encoded as non-return-to-zero(NRZ). The figure 2.5 depicts this concept.

![Figure 2.5: BPSK modulator](image)

In order to examine the bandwidth requirements for the BPSK modulator, we need to analyze the modulation from the spectrum perspective. A low frequency bitstream with frequency \( f_D \) is modulated with a carrier which has frequency \( f_c \). A multiplication of two signals in the time domain, corresponds in their convolution in the frequency domain. The spectrum of the carrier will be an impulse located in
With zero bandwidth since it is a tone. The data spectrum will be shifted around the carrier. As a result the bandwidth of the modulated signal will be equal to the bandwidth of the data bitstream. The bandwidth of the data bitstream is more complicated. A spectrum analysis of a pulse wave would have as a result an infinite number of frequencies which lead to an infinite bandwidth. However, it can be seen in practice that the bandwidth of a pulse can be considered as $f_D$ since most of the frequencies which play an important role in shaping the pulse are located between 0 to $f_D$. In order to make the comparison easier with the modulations which will be described in the next sections, we simply define the bandwidth requirements for BPSK as $R$ which in this case are equal to the bit rate of the data.

### 2.3.3.2 Symmetrical Differential Binary PSK (SDBPSK)

The SDBPSK modulation scheme can be viewed as an extension of the BPSK modulation. However, they have one very important difference. In SDBPSK the incoming bitstream is encoded with the previously encoded data. In the figure 2.6 this idea is made more clear.

![Figure 2.6: SDBPSK modulator](image)

Initially, $d_{K-1}$ has a reference value. When the bitstream $a_K$ arrives, the first bit is encoded with the reference value and the result ($d_K$) is forwarded to the modulator's input. The second bit of the bitstream which arrives after time $T$ will now be encoded with the previous result $d_K$ which has been also delayed for time $T$. The delayed value $d_K$ is shown in figure 2.6 as $d_{K-1}$. The encoder in SDBPSK is a combinational circuit with the following boolean function [15]

$$d_K = a_K \oplus d_{K-1}$$ (2.11)

Another difference with BPSK and conventional DBPSK is the phase of the carrier which now switches between ($-\pi/2$ to $\pi/2$) instead of (0 to $\pi$). Detailed information about what are the effects of this can be found in [16]. Generally, we can summarize the advantages of SDBPSK below:

- The phase of the carrier doesn’t change only with respect to the bitstream but also to the previous encoded data. Their difference decides the change of the carrier phase. SDBPSK is less susceptible to random phase changes that can occur in the transmission channel, a phenomenon which is called phase ambiguity.
- The SDBPSK demodulator is simpler than the one of BPSK because it allows the data to be demodulated non-coherently without a carrier recovery structure.

### 2.3.3.3 Quadrature PSK (QPSK)

Both modulation schemes described above, BPSK and SDBPSK, have 1 bit as a symbol which makes them poor in terms of bandwidth efficiency. QPSK belongs to a family of modulation schemes called MPSK schemes or M-ary PSK schemes. Common feature of all of them is that they represent a symbol with $\log_2 M$ bits [15]. That means in SPSK the symbol is 3 bits, in 16PSK the symbol is 4 bits and QPSK (4PSK) the symbol is 2 bits. The advantage of QPSK over the other MPSK modulation schemes is that it doesn’t suffer from bit error rate degradation. In the 16PSK scheme for example, the fact that 16 phase shifts need to be detected increases the error probability and of course the complexity of the receiver. In figure 2.7 the structure of a QPSK modulator is shown.
Chapter 2. Background

2.3. Digital Modulation

The incoming data bitstream \( p(t) \) with bit period \( T \) and bitrate \( R \) is passing through the non-return-to-zero conversion and from the logic levels \{1, 0\} is converted to the logic levels \{1, -1\}. A serial to parallel converter receives the data and splits them in two channels. The upper channel is the I (In-Phase) arm and down channel is the Q (Quadrature) arm. Due to the fact that the data arrive serially, the serial to parallel converter needs time \( 2T \) to release them simultaneously in the channels. As a result the bitrate on the channels is half of the incoming bitstream (\( \frac{R}{2} \)). In that way, a QPSK modulator requires half the bandwidth to transmit a specific amount of data comparing to the BPSK scheme.

In BPSK, the symbol to phase mapping is straightforward since there are only two possible symbol values and two phases which are anti-diametric in the constellation diagram are chosen. Phases like 0 and \( \pi \) rad are the most common because we achieve the best possible bit error rate. The modulated signal switches between two peak values which makes the information easily detectable from the receiver. On the other side, the symbol in QPSK is 2 bits and there are four possible bit combinations in the channels which lead to four distinct phase shifts. The symbol to phase mapping is a bit more complex here and needs a good understanding. For that reason we will base our description on the constellation diagram in figure 2.8 which shows a widely used symbol to phase mapping [15] and it is the one also which has been used in our implementation.

The four black spots depict the four symbols and their assignment to specific phases. The horizontal axis is the In-Phase component and the vertical axis is the Quadrature component. The distance of each spot from the reference and its phase from the positive horizontal axis show the amplitude \( A \) and the phase \( \theta \) of the modulated signal, respectively. The symbol to phase mapping has been done having in mind the following two considerations:
The four symbols split the constellation diagram in four regions. The receiver detects an incoming phase shift and according to the region it lies, translates it back to binary information. Due to noise or the propagation through the transmission medium a signal transmitted may be slightly shifted in constellation diagram and for that reason it is very important for the receiver to understand the start and the end of each region. With the current symbol to phase mapping it is simple because the regions lie on the I and Q axes. In a different mapping we would need a more complex receiver. Only BPSK and QPSK schemes have this advantage and it is a factor which contributes to their good bit error performance.

Noise can also force the modulated signal to move to a different decision region by altering the phase of the carrier. In this case the receiver demodulates wrong data. However, there are ways to reduce the effect that this can have on the bit error rate. We map the symbols around the constellation diagram according to Gray encoding. The idea is that each symbol must differ only one bit with the symbol in the neighbouring regions. As a result, only one of the two bits is now wrong instead of two which would be without Gray encoding.

**2.3.3.4 Offset Quadrature PSK (OQPSK)**

OQPSK modulation scheme is an improved variation of the QPSK scheme which is more constant envelope. In that point it is useful to focus a bit on the constant envelope term used in telecommunications theory and the importance of it.

A modulation is called constant envelope when the amplitude of the modulated signal is not changing with the bitstream. Amplitude modulation schemes are non-constant envelope since the amplitude of the carrier varies with respect to the bitstream. On the other side, phase and frequency modulations are constant envelope since the information lies on the phase and frequency of the modulated signal, respectively. However, a close look on a BPSK or QPSK modulated signal reveals that this is not fully correct for phase modulations. The fact that phase shifts of $\pi$ rad exist result in the inversion of the carrier’s amplitude and makes these modulation schemes non-constant envelope.

A question which naturally arises is what are the advantages of a constant envelope modulation. The answer to that question is related to the power amplifier stage of the modulator. Constant envelope modulations allow power amplifiers to work very near to their saturation or non-linear levels. On the other side, non-constant envelope modulations restrict the operation of the power amplifiers only to their linear region. As a result, in non-constant envelope modulations the power amplifiers are larger, consume more power and they are less efficient.

OQPSK modulation scheme adds significant improvement to QPSK regarding this aspect. In QPSK the transition from symbol 11 to symbol 00 leads to a phase shift of $\pi$ rad. That happens because we have simultaneous change on both channels. In OQPSK, the Q channel is delayed by time $T$ and considering that the period in each channel is $2T$ only one channel can change every time $T$. In figure 2.9 is depicted the structure of the OQPSK modulator and the delay in the Q arm can be seen.
As a result, only $\frac{\pi}{2}$ phase shifts can occur which leads to a more constant envelope signal since we don’t have now these abrupt amplitude changes. Except from this, the QPSK and OQPSK modulations are identical leading to the same bit error rate and bandwidth efficiency.

2.3.3.5 Audio Frequency Shift Keying (AFSK)

AFSK belongs to the type of angle modulations where the bitstream changes the frequency of the carrier. This type of modulation is called Frequency modulation and in digital communications is known as Frequency Shift Keying (FSK) modulation. AFSK differs from regular FSK because it performs modulation in baseband frequencies like audio tones.

An example of AFSK modulation is the Bell 202 modem [17] and the structure of it is depicted in figure 2.10. Two local oscillators produce two sinusoidal signals with frequencies $f_1$ and $f_2$ respectively. These two signals are important to have the same phase for a coherent modulation. According to the Bell 202 specifications the $f_1$ is the Mark frequency and it is 1200Hz ±10Hz. The $f_2$ is called Space frequency and it is 2200Hz ±10Hz. A logic ‘1’ in the binary data results to the propagation of the Mark frequency and a logic ‘0’ to the propagation of the Space frequency. This is easily achieved with a multiplexer where the binary data are the selection line and the sinusoidal signals its inputs.

![AFSK Modulator Diagram](image)

**Figure 2.10: AFSK modulator**

In this thesis work an AFSK modulator is implemented based on the Bell 202 specifications. However, it is worth to analyse the relation between the $f_1$ and $f_2$ frequencies and the bitrate of the binary data and how this affects the characteristics of the modulated signal in order to understand the general concept behind FSK modulations.

Let us specify the two FSK frequencies as

$$
\cos(2\pi f_1 t) \quad \text{for Mark} \quad (2.12)
$$

$$
\cos(2\pi f_2 t) \quad \text{for Space} \quad (2.13)
$$

where in both cases $kT \leq t \leq (k+1)T$. Theoretically, these two frequencies can have any value. However, considering the complete communication system, there is strong need for coherent demodulation because it offers better performance. Coherent demodulation is achieved when the $f_1$ and $f_2$ frequencies are chosen to be orthogonal. That means they don’t interfere with each other and they can be easier detected from the receiver. The condition for orthogonality is

$$
\int_{kT}^{(k+1)T} \cos(2\pi f_1 t) \cos(2\pi f_2 t) dt = 0 \quad (2.14)
$$

From the integral theory of trigonometric functions we know that

$$
\int \cos(mt) \cos(nt) dt = \frac{\sin(m+n)t}{2(m+n)} + \frac{\sin(m-n)t}{2(m-n)} \quad (2.15)
$$

Using (2.15), (2.14) is converted as following
\[ \int_{kT}^{(k+1)T} \cos(2\pi f_1 t) \cos(2\pi f_2 t) dt = \frac{sin2\pi(f_1 + f_2)T}{4\pi(f_1 + f_2)} + \frac{sin2\pi(f_1 - f_2)T}{4\pi(f_1 - f_2)} \] (2.16)

To satisfy the need for orthogonality, (2.16) should be equal to 0. That will occur only when both nominators are also equal to 0. This requires \(2\pi(f_1 + f_2)T = n\pi\) and \(2\pi(f_1 - f_2)T = m\pi\) where \(n, m\) are integers. From the above we can derive the following formulas for \(f_1\)

\[ f_1 = \frac{n + m}{4T} \] (2.17)

and for \(f_2\)

\[ f_2 = \frac{n - m}{4T} \] (2.18)

By subtracting 2.17 and 2.18 we find that for orthogonality the separation between the two FSK signals need to be

\[ f_1 - f_2 = \frac{m}{2T} \] (2.19)

From 2.19, very important conclusions can be derived. It is obvious that the bitrate of the input data is closely related to the frequency separation. For orthogonality, the frequency separation should be an integer multiple of \(\frac{1}{2T}\). In Bell 202, the absolute value of \(f_1 - f_2\) is equal to 1000 Hz. That means, the bitrate of the incoming data should be an integer multiple of 500 bps for orthogonality.

Another issue which should be taken under consideration is phase continuity. The relation of the bitrate with the \(f_1 - f_2\) difference decides if there will be abrupt phase changes in the bit boundaries. For phase continuity the frequency separation should be equal to \(\frac{1}{2T}\). In that case, the frequency changes are very smooth.

### 2.3.3.6 Minimum Shift Keying (MSK)

MSK is a very spectrally efficient form of FSK since the frequency separation is equal to half the data bitrate [18, 19]. The formula (2.19) can be modified for MSK as following

\[ f_1 - f_2 = \frac{1}{2T} \] (2.20)

This is the minimum possible separation for orthogonality. The carrier or center frequency is given by

\[ f_c = \frac{f_1 + f_2}{2} \] (2.21)

and it can be easily extracted that the mark and space frequencies take the form

\[ f_1 = f_c - \frac{1}{4T} \] (2.22)

\[ f_2 = f_c + \frac{1}{4T} \] (2.23)

The formula (2.22) converts the general formula for frequency modulations (2.6) as in

\[ s(t) = \cos(2\pi(f_c + d_k \frac{1}{4T})t + \theta_k) \] (2.24)

were \(kT \leq t \leq (k+1)T\). In order to achieve phase continuity we must make sure that the phase doesn’t
change on the transition from \((k-1)T \leq t \leq kT\) to \((kT \leq t \leq (k+1)T)\). That means in the time instant \(kT\), the phases must be equal which brings the equation
\[
\frac{\pi d_{k-1}}{2T} (k-1)T + \theta_{k-1} = \frac{\pi d_k}{2T} kT + \theta_k
\] (2.25)

Solving (2.25) for \(\theta_k\) we get
\[
\theta_k = \theta_{k-1} + \frac{k\pi}{2} (d_{k-1} - d_k)
\] (2.26)

From equation (2.26) we can derive the following two cases:

- When two consecutive bits are equal the current phase \(\theta_k\) remains equal with the previous \(\theta_{k-1}\)
- When two consecutive bits are different, then the phase should be shifted by \(\pi\) rad when \(k\) is odd in order to achieve phase continuity of the modulated signal.

### 2.4 Digital Demodulation

In the RF Front-Ends section, the Homodyne and Super-heterodyne receivers have been discussed. In satellite applications where the Doppler effect is more intense due to the high spacecraft speeds, the Homodyne receivers are more difficult to implement due to the fact that they need perfect frequency matching between the Local Oscillator and the incoming RF signal. The Doppler effect makes the frequency of the incoming RF signal shift which rises the need for a mechanism that will allow the Local Oscillator to track it properly in order to get the baseband signal as expected after mixing. Tracking loops in that high frequencies are power hungry and more difficult to implement compared to the ones that would be required in low frequencies. On the other side, Super-heterodyne receivers usually produce an Intermediate Frequency (IF) which even though it is much smaller than the RF one, is still relatively high in order to be properly digitized with low cost Analog to Digital converters for baseband processing. A solution which is prefered in those cases is a Low-IF receiver which is similar to Super-heterodyne with the difference that now the IF frequency is very close to the baseband frequency [7]. The advantages of this are that the IF frequency can be digitized more accurately and the baseband signal can be recovered digitally. Also the digital methods for doppler shift tracking of the digitized Low-IF frequency are easier, more low power and accurate to implement in hardware.

In this thesis work a complete BPSK modem was implemented and for that reason the next sections in this chapter are focused on this scheme. The demodulator has been tested with the modulator’s output simulating a Low-IF receiver Front-End and for that reason an all digital solution was to be implemented.

#### 2.4.1 Types of Demodulation

Digital demodulation can be distinguished in two types, Coherent and Non-Coherent. Both types have advantages and disadvantages which are summarized below [15]:

- Coherent demodulators require carrier information for data retrieval. That makes necessary more complex circuitry that will be able to recover the carrier from the modulated signal. Coherent demodulators offer much better bit error rate performance compared to non-coherent ones.
- Non-Coherent demodulators don’t need carrier information for demodulation. Usually a fixed oscillator is used. The result is much simpler hardware but with higher bit error probability.

#### 2.4.2 BPSK Demodulation

The importance of the satellite’s receiver doesn’t allow for performance compromises. A reliable demodulator that will offer the best possible bit error performance is essential. For that reason, coherent demodulation has been chosen for the BPSK modem. In figure 2.11 the structure of a BPSK demodulator is shown.
A carrier recovery circuit is used to recover the carrier signal from the incoming modulated signal. Their multiplication has as a result a high frequency component and the baseband signal. A low-pass filter is used to remove the high frequency component and a discriminator is used in the output of the filter to bring the recovered data to the required logic levels \{1, 0\}.

This point is the most critical part of the demodulator. To retrieve the baseband signal, the recovered carrier needs to have the same frequency as the incoming signal and be in-phase with it. The ability of the carrier recovery circuit to track the frequency variations of the incoming signal and fine-tune the recovered carrier to be in-phase decides in a big proportion its performance and suitability for space applications.

![BPSK demodulator](image)

Figure 2.11: BPSK demodulator

### 2.4.3 Carrier Recovery Techniques

#### 2.4.3.1 Costas Loop

In order to understand the Costas Loop operation, an analysis of the structure shown in figure 2.12 is useful [15, 20].

![Costas Loop](image)

Figure 2.12: Costas Loop

Let us assume that a BPSK modulated signal \( r(t) \) with the form

\[
r(t) = \cos(\omega_c t + \phi)
\]  

(2.27)

is an input to the Costas Loop. This is point 1 in figure 2.12. The signal \( r(t) \) is multiplied with two orthogonal sinusoidal signals created from the Numerical Controlled Oscillator (NCO) which splits the loop in two channels, the In-Phase (I) and the Quadrature (Q). Initially, the NCO is set to a frequency close or identical to the one expected from the transmitter. In the In-Phase upper channel the signal generated from the NCO (point 2) is given by

\[
\cos(\omega_c t + \theta)
\]  

(2.28)

and the multiplication with the \( r(t) \) has as a result at the low-pass filter’s input a signal with two frequency components. The form of this signal is
From trigononetry, we know that the multiplication product of two cosine signals is given by

\[ \cos(x) \cos(y) = \frac{1}{2} \left[ \cos(x - y) + \cos(x + y) \right] \]  

(2.30)

From (2.29) and (2.30), we can calculate the multiplication product which takes the form

\[ \cos(\omega_c t + \phi) \cos(\omega_c t + \theta) = \frac{1}{2} \left[ \cos(\phi - \theta) + \cos(2\omega_c t + \phi + \theta) \right] \]  

(2.31)

The second term is the high frequency component which is not needed and the low-pass filter in the I channel is used to reject it. However, the other term is the baseband signal and it is our retrieved data. So, the signal in the filter’s output (point 4) is

\[ \frac{1}{2} \cos(\phi - \theta) \]  

(2.32)

Similarly we can calculate for the Q channel. The NCO generated signal for this channel (point 3) has \( \frac{\pi}{2} \) rad phase compared to the one for the I channel and is given by

\[ \sin(\omega_c t + \theta) \]  

(2.33)

For the Q channel the multiplication product has again two frequency components and is expressed by

\[ r(t) \sin(\omega_c t + \theta) = \cos(\omega_c t + \phi) \sin(\omega_c t + \theta) \]  

(2.34)

From trigononetry, we know that the multiplication product of a cosine signal with a sine is given by

\[ \cos(x) \sin(y) = \frac{1}{2} \left[ \sin(x - y) + \sin(x + y) \right] \]  

(2.35)

From (2.34) and (2.35), we can calculate the multiplication product which takes the form

\[ \cos(\omega_c t + \phi) \sin(\omega_c t + \theta) = \frac{1}{2} \left[ \sin(\phi - \theta) + \sin(2\omega_c t + \phi + \theta) \right] \]  

(2.36)

The second term is the high frequency component and it is rejected by the low-pass filter. So, the signal in the filter’s output (point 5) is given by

\[ \frac{1}{2} \sin(\phi - \theta) \]  

(2.37)

The output results of the filters in the I and Q channels are used to calculate a signal called error which drives the NCO. The error which is the product of the multiplication between (2.32) and (2.37) can be expressed as

\[ \frac{1}{2} \cos(\phi - \theta) \frac{1}{2} \sin(\phi - \theta) = \frac{1}{4} \left[ \cos(\phi - \theta) \sin(\phi - \theta) \right] = \frac{1}{4} \left[ \frac{1}{2} \left[ \sin(0) + \sin 2(\phi - \theta) \right] \right] \]  

(2.38)

\[ = \frac{1}{8} \sin 2(\phi - \theta) \]

From the analysis above we can derive the following conclusions. When the incoming signal (2.27) has
the same frequency and it is in phase with the NCO, then the error signal is 0 and from the I channel we can retrieve the digital data. In that case the loop is in lock condition. A discriminator is commonly used to bring the filter’s output to the appropriate digital levels \{1, 0\}. However, a difference in phase will result to an error which will increase or decrease the frequency of the signals (2.28) and (2.33). As a result of this, the difference $\phi - \theta$ tends to reach 0 or $\pi$ in order to bring the error to 0 and the loop back in lock condition.

The Costas Loop is preferred over the conventional PLL in wireless communication systems where the Doppler shift is intense. The reason is because the error of the Costas Loop is $\sin 2(\phi - \theta)$ instead of $\sin(\phi - \theta)$ in normal PLLs. That results in twice the sensitivity in phase changes between the NCO and received signal. Another advantage of Costas Loop is its locking capability with phase modulated signals in its input [21].

### 2.4.4 Timing Recovery Techniques

The Costas Loop demodulates the Telecommand signal and recovers the carrier. Its ability to track the frequency of the incoming signal makes it an ideal solution for the Doppler shift problem. However, the Costas Loop itself is not enough for a complete modem application. This can be easily derived from figure 2.12. The only output from Costas Loop is the demodulated data. In order to process these data in later stages of the application a clock which is synchronized with them is necessary and it is not provided.

Timing synchronizers or Symbol timing recovery circuits aim to solve exactly this problem. Given a digital signal, which in our case is the retrieved Telecommand data, they create and synchronize a clock signal with them in order to give the ability for further processing of this data with a state machine or other synchronous circuit.

There are two main types of Timing synchronizers [15]. One is the open-loop synchronizers which recover the clock from the Costas Loop retrieved data and they usually do that with non-linear ways. The second type is the closed-loop synchronizers which with repeated measurements try to match the generated clock with the incoming signal.

In this thesis work we chose the Early-Late Gate Algorithm since it is a closed-loop synchronizer with good performance in terms of synchronization speed and occupies a relatively small number of hardware resources.

#### 2.4.4.1 Early-Late Gate Algorithm

The structure of an Early-Late Gate closed-loop synchronizer is shown in figure 2.13.

![Figure 2.13: Early-Late Gate algorithm](image)

The data signal is sampled from two accumulators in two different timing periods. The accumulator which samples the time period between ($0$ to $\frac{T}{2}$) is called Early accumulator and the other which samples the period ($\frac{T}{2}$ to $T$) is called Late accumulator. The start and end of the sampling period for both
accumulators is controlled by the Numerical Controlled Oscillator (NCO) which generates also the clock signal. Three conditions can be distinguished:

- In case one clock period is synchronized with a bit period, the accumulated data of the Early and Late accumulators are equal which means that the error signal will be 0. A zero error signal will have no effect on the NCO’s frequency.

- In the case where the Early accumulator starts sampling \( \delta \) time after the rising edge of a bit, then the Late accumulator data will be smaller than the Early. That will imply a positive error signal which will tend to increase the clock frequency and restart it in order to bring it towards the timing of the data signal.

- The last case where the Early accumulator starts sampling before the rising edge of the bit leads to a negative error. Here the Early accumulator data will be smaller compared to the Late. The negative error will reduce the clock frequency and restart the clock to synchronize it with the incoming data signal.
Chapter 3

BPSK Modulator

3.1 BPSK Block Diagram

In the BPSK modulation scheme a change in the logic level of the binary information changes the phase of the carrier signal. This can be realised as a combination of two sinusoidal signals with a difference in phase of $\pi$ rad. However, thinking in hardware, the equation (2.10) can be expressed also in a slightly different way as in

$$s(t) = \begin{cases} A_c \cos(2\pi f_c t) & \text{for } 1 \\ -A_c \cos(2\pi f_c t) & \text{for } 0 \end{cases}$$ (3.1)

That means the BPSK modulation can be implemented as a sinusoidal signal which gets inverted upon an arrival of a bit of logic level '0'. The BPSK block diagram in figure 3.1 depicts the hardware components and their interconnections that realize this idea in the hardware. All of them have been implemented in VHDL and connected to the top level using the SmartDesign tool included in the Libero v9.1 Suite. In the forthcoming sections each of the components is described paying extra attention to design decisions taken and their effect on the performance of the modulator. Finally, emphasis is given on the synthesis results and the testing on the FPGA device.

![Figure 3.1: BPSK Block Diagram](image)

3.1.1 Carrier Generation

The $\text{cosine} \_\text{generation}_0$ in the block diagram is the cosine generation component and it generates the sinusoidal carrier signal. The quality of the generated signal is an important factor which affects the power
spectrum of the modulated signal but due to the limited resources on the FPGA a proper balance must be kept that will guarantee good performance and at the same time efficient use of the device resources. For that reason, a specific quantization step had to be decided for the synthesis of the cosine waveform.

### 3.1.1.1 Quantization Effect on Spectrum

The carrier theoretically is a sinusoidal signal with zero bandwidth and its spectrum is expected to be only the fundamental frequency (tone). In our application due to the low bitrates requested, the carrier can be generated digitally in the FPGA. However, a digitally synthesized sinusoidal signal doesn’t have the same spectrum as the theoretical one. There are two factors that affect its quality and these are the number of samples per period and the quantization step used to shape it. In theory an infinite number of samples per period (a feature existing in continuous time signals in contrast to discrete ones) and infinitely small quantization step would give the perfect tone. The limited number of hardware resources in the FPGA raises the need to make some compromises in performance and analyze to what extent they affect the spectrum of the generated signal.

To start with, we considered the case of 64 samples per period and a quantization of 16 steps. Analysing this case with Matlab, we can see that in the spectrum plots there is not only the fundamental frequency but also some other frequencies due to sampling and some noise in between them. These are good to be avoided in general. This case is shown in figure 3.2.

**Figure 3.2: 64 Samples - 16 Quantization steps**

These frequencies occur due to sampling and the noise between them due to quantization. We will refer to these frequencies as sampling effects. To reduce the noise we analysed the spectrum also with a larger number of quantization steps. A number of 64 steps gave the results shown in figure 3.3. We observe that the noise now is negligible compared to the sampling effects. A conclusion derived from these two results is that a number of at least 64 steps is good to be used to reduce the noise due to quantization. However, that doesn’t mean fewer number of steps will have disastrous effect in the communication system but the amplitude of the noise becomes comparable to that of the other frequencies.

Next, we tried to reduce the amplitude of the sampling effects. To achieve that we had to shape a more continuous-time signal which means a higher number of samples per period. We kept initially the quantization steps to 16 in order to see their effect in the spectrum. The results are shown in figure 3.4. As expected the sampling effects have now very low amplitude compared to the fundamental frequency but the noise due to quantization can be observed. Finally, we tested the case of 256 samples per period and 256 quantization steps. That case gave us the best results since both the sampling effects and the noise are negligible compared to the fundamental frequency which now looks more like a tone.

As a final remark we can add that the decision is all about hardware resources. Noise when is kept in low levels compared to the sampling effects, can have very little impact in the communication. In our case we chose a number of 256 samples and 128 quantization steps.
Chapter 3. BPSK Modulator

3.1. BPSK Block Diagram

3.1.1.2 Look-Up Table

Upon the decision to use a quantized 128 step cosine waveform with 256 samples per period, as a carrier, a Look-Up table was the proper choice in order to store the values that would shape it. Taking advantage of the sinusoidal signal characteristics, only the values of the first quadrant (0 to $\frac{\pi}{2}$) needed to be stored since the other three quadrants ($\frac{\pi}{2}, \pi$, $\frac{3\pi}{2}$) and ($\frac{3\pi}{2}, 2\pi$) can be easily shaped from it [22]. In this way we managed to save resources on the FPGA fabric. So, a Look-Up table with 64 samples and 8-bit values stored in it, is used in this design.

3.1.1.3 Carrier Generation State Machine

The fact that the Look-Up table stores the values for only the first quadrant makes necessary a mechanism to read the Look-Up table properly and generate a complete period of the waveform. A state machine has been designed for that purpose and the figure 3.6 shows the transition diagram of it.

Initially, when reset is asserted, the state machine is on the counting_down state and the scanning address of the Look-Up table is set to 63. This is because the carrier is cosine and the amplitude can only decrease upon start-up. When the reset is de-asserted the state machine is ready to scan the Look-Up table to generate the first quadrant (0, $\frac{\pi}{2}$). The Look-Up table is scanned from address 63 down to 0. When it reaches address 0, it sets a flag to false which means that it is entering the negative cycle of
the cosine waveform and changes the state to \textit{change\_up}. The state then from \textit{change\_up} transits to \textit{counting\_up}. The Look-Up table is now scanned upwards but the values passed to the output are the 2's complement of the ones in the table. The scanning starts from address 0 up to 63. When it reaches address 63, then the next state is \textit{change\_down}. During \textit{change\_down} the flag remains to false because the third quadrant lies on the negative cycle. The state now changes to \textit{counting\_down} and the Look-Up table is scanned again backwards down to address 0. When it reaches address 0 the state transits to \textit{change\_up} and the flag sets to true which means that we are entering the positive cycle and the Look-Up table should be scanned again upwards. The state now transits to the \textit{counting\_up} state and the Look-Up table is scanned from address 0 up to 63. When the address 63 is reached, the state transits to \textit{change\_down} with the flag remaining unchanged because we are still on the positive cycle. In that point a complete period has been scanned. The next state is \textit{counting\_down} and a new period of the cosine wave starts.

It is important to note that for a complete period we need 256 clock cycles. The cosine generation component is clocked with 3.125MHz which results in a carrier with frequency 12.207 KHz. The reason we chose this carrier frequency is the data bitrate. In the satellite communications where our modulator will be used, there was a requirement for a minimum bitrate of 4800 b/s for telemetry. We have finally chosen a bitrate of 6103 b/s since it was hardware resources efficient to generate all the required clocks, based on the on board oscillator and the device’s PLL, for that frequency. In BPSK scheme, to take advantage of the low bit error rate, it is important for the carrier frequency to be an integer multiple of the data bitrate. That’s the reason we selected 12.207 KHz as our carrier frequency. In the same way reducing the quantization step of the carrier, we can achieve frequencies like 24.414 KHz for 128 steps and 48.828 KHz for 64 steps.

### 3.1.2 Inverter

In order to achieve the phase change by \(\pi\) rad we use an inverter block (\textit{invBlock\_0}). The aim as described from (3.1) is to invert the amplitude of the carrier with respect to the incoming bitstream since the two signals are antipodal. This implementation is more resource efficient than using a multiplier as in figure 3.8 which would need also an NRZ block to convert the incoming data from \{1, 0\} to \{1, -1\} in order to achieve the amplitude inversion.

The inverter block is implemented using \textit{NOT} gates which invert the 8-bit value received from the cosine generator. The inverted value is forwarded to an 8-bit Full Adder and it is added with 1 in order to take as a result the 2's complement of it. Finally, a two input multiplexer selects the original 8-bit value in case of logic '1' in the bitstream input or the inverted one in case of a logic '0'. Table 3.1 depicts the scanning values of the Look-Up table for the first two quadrants and the same values after the inversion block.

<table>
<thead>
<tr>
<th>Quadrant - Address (63 - 0)</th>
<th>Quadrant - Address (0 to 63)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look-Up table</td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>139 136 133 130 126 123 120 117 ...</td>
</tr>
<tr>
<td>Inversion</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>117 120 123 126 130 133 136 139 ...</td>
</tr>
</tbody>
</table>

One important thing to observe from table 3.1 is that the value in the lowest peak of the generated waveform is positive and it is 1. Since the modulated signal will be forwarded to a Digital to Analog converter, a DC offset has been added to the waveform on purpose. So, the amplitude instead of ranging between \{127, -127\}, it is in the range \{255, 1\}.

The inversion block is a pure combinational circuit. As soon as a new value arrives from the cosine generator it is inverted and the output is updated with respect to the bitstream input. However, the update of the output is not done directly and most important not for all the bits at the same time. There is a propagation delay because of the carry and as a result the MSB of the output is updated with the correct value later than the LSB. This can be seen in the figure 3.7.

Since the inversion block’s output is forwarded to the DAC’s input, it is important to ensure a stable
value on its input in every sampling instance to avoid any spikes or sampling of erroneous data. For that reason, a negative edge triggered register is used to store the stable value. It has been verified with timing analysis that the propagation delay is much less than half the period of the clock and so storing the value in the falling edge of the clock ensures the correctness of the sampled data.

### 3.1.3 DA Controller

The nanoRTU comes with two DAC7552 Digital to Analog converters on board and one of them will be used to convert the modulated signal. For detailed information about the device characteristics and the controlling sequence refer to [23]. For the needs of the project, a hardware controller has been implemented that will initialize, configure and control the DA converter on the board. The controller is mainly a state machine and the transition diagram of it is shown in figure 3.8.

![Figure 3.8: DA Controller state machine](image)

During the *idle* state the 16-bit word is loaded to the SPI register. The first four bits of the register, starting from the MSB, decide the channel that will be used to output the converted result and they are called control bits. In our implementation we chose to operate channel A, so the control bit pattern is 0100. The other twelve bits are the data to be converted. Since the modulated data are 8-bit values, they are concatenated with '0' to create a 12-bit signal in order to drive the DAC’s input. The *sync* and *sclk* signals are set to logic '1' during the *idle* state. In the next clock cycle the state machine transits to the *pulse_sync1* state where the *sync* signal is set to logic '0' and the *sclk* remains '1'. We leave the values of *sclk* and *sync* unchanged also in state *pulse_sync2* in order to ensure that the *sync* signal has been stable at '0' before we start the transfer of the SPI register data. In the next state *write1*, *sclk* toggles to '0' and the MSB of the SPI register is transmitted. It is important to note here that the data are transmitted on the falling edge of the *sclk* signal. In state *write2*, the *sclk* toggles back to '1' and a counter increments its value by 1 and the register data are shifted to the left. The transition between states *write1* and *write2* happens 15 times until all the bits are shifted to the DA converter.

The board has a 16 MHz crystal oscillator. If that frequency was used to clock the DA controller, then
the sclk frequency would be 8 MHz. Considering that 16 cycles are needed to transfer the 16-bit data word, that means the carrier generation module needs to be clocked with a frequency of 500 KHz in order to be sampled properly. In that case the carrier frequency would be 1.953 KHz taking into consideration the quantization of 256 steps. A frequency like this was too low, since there was a demand to be 4.8 KHz or higher. For that reason, it has been decided to work the DAC in its full potential. A PLL is used to provide a clock of 100 MHz to the DA Controller in order to generate an sclk signal of 50 MHz which is the maximum clock it can handle. The carrier generation module can be clocked with 3.125 MHz in order to generate a 12.207 KHz carrier.

### 3.1.4 PLL and Reset Block

In order to generate the two frequencies 100 MHz and 3.125 MHz we need to use a PLL. The ProASIC3L FPGA family, which is used with the nanoRTU board, comes with an integrated PLL structure which can be easily configured to match our requirements. As an input frequency the 16 MHz clock is used which is provided from the on board crystal oscillator of the nanoRTU. The first output of the PLL is set to 100 MHz in order to clock the DA Controller component. The second output is set to 3.125 MHz and it is used to clock the carrier generation component and everything else. Unless specified otherwise, all the blocks described in this chapter use the 3.125 MHz clock.

The use of a PLL requires a special treatment to the reset structure of the system. The reason is that the PLL needs some time in order to lock into the specified frequencies. There is a lock signal as an output from the PLL in order to solve exactly this problem. It is set up and remains to logic '1' when the output frequencies of the PLL become stable. We implemented the circuit depicted in figure 3.9 which de-asserts the reset of the system only when the reset gets to '1' and the lock signal gets to logic '1'. The powerdown signal is used to disable the PLL when the reset is asserted.

![Figure 3.9: Reset block](image)

### 3.2 Testing

In order to test this implementation on the nanoRTU board, a number of hardware blocks that provide the bitstream have been added to the system. The structure and the interconnections of these blocks is shown in figure 3.10.

#### 3.2.1 UART Controller

A user who has the board connected to a computer through UART, will send ASCII characters or bytes using a terminal. A UART hardware block will receive the serial data and store them in a register. When the reception of a full character has been completed, the data available signal will set to logic '1'. The UART component is provided from ÅAC Microtec in order to be used in this project. It has been instantiated in the top level design with the proper configuration of 9600 baud rate and no parity bit.

#### 3.2.2 Data Latch

UART is an asynchronous protocol which means that a temporary storage of the incoming data is necessary in order to read them later synchronously. Usually, a FIFO is used for this purpose but in our case a simple register would be enough since the data are sent externally from a user and their incoming
rate is very low. The data Latch is clocked with the same clock as the UART. In case the data available signal is set to '1', the output of the UART block is stored in the data register. The data remain stored until the next transition from '0' to '1' of the data available signal which means that a new character has been sent.

### 3.2.3 Data State Machine

A character that has been stored in the data register needs to be sent serially to the modulator’s input. A state machine has been implemented for this purpose and the transition diagram is depicted in figure 3.11.

![Data State Machine Diagram](image)

Figure 3.11: Data state machine

The state machine creates three signals that will be forwarded later to the modulator. The first is the bitstream which is a serial conversion of the data stored in the data register. The second is the flag signal which indicates when a new character transmission starts. The third is the enable signal which is set to '1' when the first bit is sent after the reset is de-asserted and is used for synchronization purposes with the carrier generation block.

During reset, the state machine is in the idle state. The flag and enable signals are both '0' since no character transmission has been started and there is no need for carrier generation. When, reset gets to '0', the state machine transits to state1. In that state, the state machine is updated with the contents of the data register and the MSB is sent to the bitstream output. The flag and enable signals are set to
1 since the first bit has been sent and the carrier should be generated. In state2 there is no change. In state3 the second bit is transmitted and the flag signal is set to '0'. The enable signal will remain to '1' for all the rest of the states. It will get to '0' only upon reset when the state machine transits back to the idle state. In state4 there is no change to the three signals except from that a counter is incremented. A repeated transition between states state3 and state4 ensures that the byte is shifted bit by bit to the modulator’s input. When a complete byte has been sent, the state machine transits back to state1 in order to load the updated byte from the data register, send the MSB and raise the flag to indicate the transmission of a new character.

One important thing to notice on the data state machine is that bits are sent on the states state1 and state3. That means the bitstream data rate will be half of the frequency which clocks the state machine. As a result the data state machine needs to be clocked with a frequency 12.207 KHz in order to produce a 6103 b/s bitstream. An 8-bit counter helps in that direction because frequency division of the 3.125 MHz clock with 256 can be easily achieved to get the desired clock for the state machine.

### 3.2.4 Hardware Results

The BPSK modulator has been tested on the nanoRTU hardware and the results are presented in this section. A terminal has been used to send bytes to the board. The settings of the communication are 8 bits, 1 stop bit, 9600 baud rate and no parity bit. As measurement instrument a TDS 2024C oscilloscope has been used which has the ability of storing the results as an image in JPEG format. This feature has been used extensively not only for the BPSK modulation but for all the thesis work.

Figure 3.12 shows the modulator results when the byte 10100100 has been sent from the user. The yellow waveform shows the modulated signal. The bright blue shows the bitstream and the purple one the byte transmission start point.

![Figure 3.12: BPSK hardware results](image)

There are some important information that can be extracted from these results and we summarize them below:

- The first and most important information is related to the phase of the modulated signal. It can be seen that the phase of the signal changes with respect to the bitstream. A bit of logic level '1' results to a 0 rad phase shift and a bit of logic level '0' results to a π rad phase shift.

- The carrier frequency is 12.207 KHz and the bitstream bitrate 6103 b/s. It can be observed from the results that a bit's period is twice the carrier’s.
• The purple waveform shows when the byte transmission starts. It is the flag signal that has been described in the sections above. It is logic '1' during the transmission of the first bit. Down on the right side of the scope we can see that the period of this signal is 762 Hz. This agrees fully with the value expected because the time which 8 bits need to be transmitted is

\[ \frac{1}{f_{\text{Bitstream}}} = 1.31 \text{msec} \]  \hspace{1cm} (3.2)

value which results in a frequency of 762.87 Hz.

• Finally, we can observe that the amplitude of the modulated signal is almost 200 mV. Recall from the carrier generation section where the Look-Up table has been described that an 8 bit value has been used to shape the waveform in each sampling instance. Considering that we gave it a DC offset the modulated signal ranges between the values \{255, 1\}. The DAC used has a reference voltage 3.3 V. From the datasheet we can find that the output voltage can be calculated from the formula

\[ V_{\text{out}} = 3.3 \frac{\text{DigitalValue}}{4096} \]  \hspace{1cm} (3.3)

From the formula above and considering that the maximum digital value is 255, the maximum \(V_{\text{out}}\) is 205 mV.

3.3 Synthesis Results

The BPSK modulator components have been synthesized all together and each one of them separately in order to get a clear picture of the consumed resources and the effect of each component on them. In the table below the core cells consumed by each component and their utilization percentage of the total resources is shown. The sum of them when synthesized separately is 764 core cells. When all the components are synthesized together the consumed resources are 620 core cells out of total 13824 which gives an utilization of 4%. This probably happens due to internal optimizations done by the tool during the synthesis process.

Table 3.2: BPSK synthesis results

<table>
<thead>
<tr>
<th>System component</th>
<th>Core cells</th>
<th>Total chip utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cosine generation</td>
<td>160</td>
<td>1%</td>
</tr>
<tr>
<td>DA Controller</td>
<td>78</td>
<td>1%</td>
</tr>
<tr>
<td>Inversion</td>
<td>23</td>
<td>0%</td>
</tr>
<tr>
<td>PLL</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Reset Block</td>
<td>2</td>
<td>0%</td>
</tr>
<tr>
<td>Register</td>
<td>8</td>
<td>0%</td>
</tr>
<tr>
<td>UART Receiver</td>
<td>423</td>
<td>3%</td>
</tr>
<tr>
<td>Data FSM</td>
<td>43</td>
<td>0%</td>
</tr>
<tr>
<td>Counter</td>
<td>19</td>
<td>0%</td>
</tr>
<tr>
<td>Data Register</td>
<td>8</td>
<td>0%</td>
</tr>
</tbody>
</table>
Chapter 4

SDBPSK Modulator

4.1 SDBPSK Block Diagram

The SDBPSK implementation has a lot of things in common with the BPSK implementation. Their main difference can be distinguished in the bitstream path due to the fact that the data are now encoded before modulation. The Block Diagram which is depicted in figure 4.1 shows the hardware components implemented to realize the SDBPSK modulation scheme on the FPGA.

![Figure 4.1: SDBPSK Block Diagram](image)

In the sections below only the hardware components which differ from the BPSK implementation are described.

4.1.1 Carrier Generation

The \( \text{cosine}_\text{generation} \) block in the diagram generates the sinusoidal carrier signal. In SDBPSK the phase of the carrier switches between \( (-\frac{\pi}{2}, +\frac{\pi}{2}) \) in contrast to BPSK where it switches between \( (0, \pi) \). That implies some minor changes in the initialization part of the carrier generation state machine. We choose as an initial phase the \( +\frac{\pi}{2} \) rad which means that after reset the Look-Up table should be
scanned upwards but the waveform will lie on the negative cycle. For that reason, the state machine
during reset is initialized in the *counting up* state and we set the flag to negative cycle in order to pass to
the output the 2’s complement of the Look-Up table values. The scanning address is initialized to 0. The
rest of the functionality remains identical to BPSK. From the *counting up* state, the carrier generation
state machine transits to all the states of the transition diagram shown in figure 3.6 in order to shape a
complete period of the carrier.

### 4.1.2 Differential Encoding

The main difference between SDBPSK and BPSK is in the bitstream path. In figure 4.1, the incoming
bitstream is fed to the encoder block (*encoder* 0) and encoded with the previous value. Each encoded
value is fed back to the encoder after passing through the delay block (*delay* 0).

#### 4.1.2.1 Encoder

The encoder is an *XNOR* operation between the bitstream input data and the previously encoded data.
Even though it is supposed to be a pure combinational circuit as described from formula (2.11), in figure
4.2 the encoded value is fed to the output of the block only in the falling edge of the clock.

```plaintext
Figure 4.2: Encoder
```

The reason for this can be easily understood with a look in the simulation results of the encoder
block in figure 4.3. The *previous encoded* value before the yellow cursor is '0', the *encoded* value is '0'
and the incoming *bitstream* changes from '1' to '0'. At the cursor time instance the delay block samples
the *encoded* value and updates the *previous encoded* value which remains '0' for another period T. The
new *encoded* value will change to logic level '1' due to the *XNOR* operation. If the circuit was pure
combinational the output of the encoder would be updated immediately. In practice, it still requires
some very short time due to propagation through the *XOR* and the *NOT* gates. This would have as a
result the new *encoded* value to be read directly by the delay block without achieving the required delay
of time T. Inserting a negative edge triggered register in the encoder’s output, makes the output to be
updated after the positive edge triggering of the delay register as shown in figure 4.3.

```plaintext
Figure 4.3: Encoder simulation
```
4.1.2.2 Delay

The delay block is a positive edge triggered register. The clock of the register is chosen according to the bitrate since we aim to achieve a delay of T. The bitrate is 6103 b/s so we need a clock with frequency of 6.1 KHz. We can easily generate that clock with a simple counter achieving frequency division from our main clock which is 3.125 MHz. The division factor is 512 so we need a 9-bit counting signal using the MSB as the required clock. In figure 4.3 at the cursor time instance, the cnt_val(8) signal transits from '0' to '1' occurring to the update of the previous encoded value. The reader is encouraged to focus his attention on the update of the encoded signal which happens on the falling edge of the 3.125 MHz clock.

4.1.3 Hardware Results

The testing procedure of the SDBPSK is identical to the one used for BPSK modulation. Since the symbol is 1-bit wide for both modulations the hardware for producing the bitstream has been re-used and it is the one shown in figure 3.10. For that reason, we focus mostly on the interpretation of the hardware results which is a bit more complex than BPSK.

The SDBPSK modulator has been tested on the nanoRTU with many different data patterns sent from the UART. The figure 4.4 shows the modulator results when the byte 10110001 has been sent by the user. The yellow waveform shows the modulated signal. The bright blue shows the bitstream, the green one the encoded data and the purple one the byte transmission start point.

Upon reset condition the delay register is loaded with a predefined value which is called reference value. The carrier is also given an initial phase. As can be seen from table 4.1 the initial value of the delay register is '1' and the initial phase is $\frac{\pi}{2}$ rad.

The first bit of the incoming data bitstream is encoded with the reference value (previous value) and a new encoded value is updated. Next, the second bit of the bitstream is encoded with the previously encoded value and the new encoded value is updated. This procedure occurs for all the incoming data leading to a new encoded bitstream which changes the phase of the carrier accordingly.

From the hardware results in figure 4.4 and the table 4.1 we can extract the following conclusions:

- The modulated signal doesn’t change in response to the bitstream but due to a differentially encoded pattern.
- The phase of the the modulated signal alternates between $(-\frac{\pi}{4}, \frac{\pi}{4})$. 
Table 4.1: SDBPSK testing pattern

<table>
<thead>
<tr>
<th>Modulation</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitstream</td>
<td>1 0 1 1 0 0 0 1</td>
</tr>
<tr>
<td>Encoding</td>
<td>1 1 0 0 0 1 0 1 1</td>
</tr>
<tr>
<td>Phase</td>
<td>$\frac{\pi}{2}$, $\frac{\pi}{2}$, $-\frac{\pi}{2}$, $-\frac{\pi}{2}$, $\frac{\pi}{2}$, $\frac{\pi}{2}$, $\frac{\pi}{2}$, $\frac{\pi}{2}$</td>
</tr>
</tbody>
</table>

- The purple waveform shows when the byte transmission starts. Down on the right side of the scope, we can see that the period of this signal is 762 Hz. This agrees fully with the value expected because the time which 8 bits need to be transmitted is

$$8 \cdot \frac{1}{f_{\text{Bitstream}}} = 1.31 \text{msec}$$  \hspace{1cm} (4.1)

value which results to a frequency of 762.87 Hz.

- Finally, we can observe that the amplitude of the modulated signal is almost 200 mV as expected.

### 4.1.4 Synthesis Results

The synthesis results show clearly that the consumed resources of the FPGA are slightly more than the BPSK implementation due to the encoder, delay and counter blocks. When the SDBPSK implementation is synthesised as a project the consumed resources are 627 out of 13824 with an utilization of 5%. The counter of the dataFSM has been extended from 8 bits to 9 bits in order to supply the 6.1 KHz clock to the delay block. However, it is still used to clock the dataFSM for the bitstream generation purpose as before. When each component synthesized separately, the consumed resources are shown on table 4.2 and all together sum up 769 core tiles.

<table>
<thead>
<tr>
<th>System component</th>
<th>Core cells</th>
<th>Total chip utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay</td>
<td>1</td>
<td>0%</td>
</tr>
<tr>
<td>Encoder</td>
<td>2</td>
<td>0%</td>
</tr>
<tr>
<td>Cosine generation</td>
<td>160</td>
<td>1%</td>
</tr>
<tr>
<td>DA Controller</td>
<td>78</td>
<td>1%</td>
</tr>
<tr>
<td>Inversion</td>
<td>23</td>
<td>0%</td>
</tr>
<tr>
<td>PLL</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Reset Block</td>
<td>2</td>
<td>0%</td>
</tr>
<tr>
<td>Register</td>
<td>8</td>
<td>0%</td>
</tr>
<tr>
<td>UART Receiver</td>
<td>423</td>
<td>3%</td>
</tr>
<tr>
<td>Data FSM</td>
<td>43</td>
<td>0%</td>
</tr>
<tr>
<td>Counter</td>
<td>21</td>
<td>0%</td>
</tr>
<tr>
<td>Data Register</td>
<td>8</td>
<td>0%</td>
</tr>
</tbody>
</table>
Chapter 5

QPSK & OQPSK Modulators

BPSK and SDBPSK modulation schemes achieve good bit error performance but the fact that the transmitted data are 1-bit wide makes them not so bandwidth efficient. Quadrature Phase Shift Keying (QPSK) and Offset Quadrature Phase Shift Keying (OQPSK) are phase modulation schemes which are more bandwidth efficient and they are widely used in satellite communications. Both of them have been implemented and tested on nanoRTU hardware, and their implementation is described in this chapter.

5.1 QPSK Block Diagram

QPSK modulation is a phase modulation where in contrast to BPSK, the symbol is 2-bits wide and achieves increased bandwidth efficiency. However, the biggest advantage and the main reason why it is used in satellite communications is that it manages to keep the bit error rate in the same levels with BPSK, something which doesn’t happen with other M-ary PSK modulations schemes like 8PSK or 16PSK which suffer from bit error degradation. The QPSK modulator has been implemented in hardware and the block diagram of the implementation is depicted in the figure 5.1.

An important feature that differentiates the QPSK implementation from BPSK and SDBPSK is that the incoming bitstream is splitted in two channels, the In-Phase (I channel) and the Quadrature (Q channel). The channels are modulated with two orthogonal sinusoidal signals and in the end they are added to shape the QPSK modulated signal. In the forthcoming sections the hardware components depicted in the block diagram are described in detail.

5.1.1 Carrier Generation

In the QPSK implementation in order to achieve the symbol to phase mapping, the generation of two sinusoidal signals is necessary. Important requirement for these two signals is to be orthogonal which means that their phase difference should be $\frac{\pi}{2}$ rad. For that reason, for the In-Phase channel a cosine signal with 0 phase will be used and for the Quadrature channel a cosine signal with $\frac{\pi}{2}$ phase. Two different state machines are necessary for scanning the Look-Up table and achieve the generation of these signals.

These state machines are the cosine_generation_0 and the sine_generation_0 blocks in the diagram of figure 5.1. Both of them are clocked with 3.125 MHz and considering the 256 steps quantization they generate 12.207 KHz signals. Their phase difference is easily achieved by manipulating their initialization states. For the cosine generation with 0 phase the initial state is the counting_down and the scanning address is set to 63 which contains the amplitude value for the peak of the first quadrant. The flag is set also to positive cycle since the first quadrant to be shaped is above 0. For the cosine generation with $\frac{\pi}{2}$ phase the initial state is the counting_up and the scanning address is set to 0. The flag is set to negative cycle since now the first quadrant is below 0.

Both state machines have an enable input in order to be synchronized properly with the incoming bitstream. Their simultaneous start is important in order to achieve the proper symbol to phase mapping. The enable signal is provided from the dataFSM which also generates the bitstream.
5.1.2 I / Q Channels

The fact that we have two channels in QPSK requires a proper handling of the incoming bitstream. It is very common in bibliography the incoming bitstream to be 1-bit wide and with a serial to parallel converter to be split in two channels. That means the bitrate on the channels will be the half of the incoming bitstream’s. Half bitrate on the channels means that the same information can be transmitted requiring half the bandwidth than BPSK and achieving twice bandwidth efficiency. This is an important requirement in broadband telecommunications. However, in some other applications the transmission speed can be more important than the bandwidth. In that case if the bitstream was 2-bits wide, the bitrate on the channels remains the same with the bitstream’s leading to a twice faster transmission but the same bandwidth efficiency with BPSK.

In our implementation we followed a slightly different approach that can work under both cases even though the implemented one aims for bandwidth efficiency. To achieve twice the bandwidth efficiency than BPSK we clock the I / Q block with 6.1 KHz, frequency generated from a 9-bit counter using the MSB as a clock. Inside the block, after frequency division, the incoming bitstream is forwarded to the channels with a bitrate of 3.05 kb/s at the proper time instance. The dataFSM block is also clocked with 6.1 KHz leading to a bitstream of 3.05 kb/s. The transmission of one byte in that case would take

$$4 \frac{1}{f_{Bitstream}} = 1.31 \text{msec} \quad (5.1)$$

which is the same time required for a byte transmission in BPSK. However, here the bitstream on the channels is 3.05 kb/s leading to better bandwidth efficiency.

In the case we would like to aim for transmission speed, we could simply use the MSB -1 bit of the counter to clock the I / Q block leading to a bitrate on the channels of 6.1 kb/s. The transmission of one byte in that case would be

$$4 \frac{1}{f_{Bitstream}} = 0.655 \text{msec} \quad (5.2)$$
which is half compared to BPSK. However, the bitrate on the channels would now be same with BPSK making both modulations equal in terms of bandwidth efficiency.

In figure 5.2 the internal structure of the I / Q block is shown. A register is used for frequency division producing a clock that drives the I and Q registers. The registers forward the incoming bitstream to the channels properly.

As a final remark, we could add here that this implementation works perfectly in our case since when the bitstream transmission starts a full byte has been already received from the UART and is stored on the dataLatch register. That guarantees that there is always data to transmit. That would also work in applications where a FIFO is used to temporarily store the incoming streaming data. However, in cases where there is no temporary storage of the data and the incoming bitstream is 1-bit wide, a serial to parallel converter is the only solution and as it is obvious transmission speed better than BPSK can not be achieved.

5.1.3 Inverter

Upon the bitstream generation of the I and Q channels, the next step is the data modulation. Here we could note that a QPSK implementation could be easily considered as a double BPSK implementation with the difference that now the sinusoidal signals are orthogonal. The inverter blocks used in QPSK are identical to the ones used for BPSK. One inverter is used for each channel to invert by $\pi$ rad the sinusoidal signal in case of logic 0 or leave it with no phase shift in case of logic 1. The modulated data need to be latched on the falling edge to ensure stable values on the final adder’s inputs.

5.1.4 Adder

The adder in QPSK implementation is used to add the modulated I and Q channels. Here in that point is visible the importance of orthogonality in the sinusoidal signals. The modulated signal in the I channel switches between the phases $(0, \pi)$ and the modulated signal in the Q channel switches between the phases $\left( -\frac{\pi}{2}, \frac{\pi}{2} \right)$. After addition, the modulated signal will be able to switch between four phases. The table 5.1 shows these four combinations.

For better understanding how the modulated phases are derived, a look in the constellation diagram in figure 2.8 will help. The phase of the channels decide the borders of each region. The phase of the modulated signal is the central phase of this region. In a M-ary PSK modulation with high M the borders will be closer to each other making the region much smaller and harder detectable.

The adder implemented uses the structure depicted in figure 3.7. For that reason, after addition the modulated data need to be latched again and concatenated with 0 in order to drive the DAC’s input with a stable 12-bit value.

Figure 5.2: I / Q Channels block
Table 5.1: Symbol to phase mapping

<table>
<thead>
<tr>
<th>I channel</th>
<th>Q channel</th>
<th>Modulated signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>0</td>
<td>(11) + π/4</td>
</tr>
<tr>
<td>(1)</td>
<td>0</td>
<td>(10) − π/4</td>
</tr>
<tr>
<td>(0)</td>
<td>π</td>
<td>(01) + 3π/4</td>
</tr>
<tr>
<td>(0)</td>
<td>π</td>
<td>(00) − 3π/4</td>
</tr>
</tbody>
</table>

5.1.5 Testing

In order to test the QPSK modulator the same strategy as BPSK and SDBPSK has been deployed. Data sent through the UART are stored to a register. In figure 5.3 the Block Diagram of the QPSK bitstream generation components is shown. The only difference with the previously described modulations is in the dataFSM block. The byte stored in the dataLatch register is read in groups of two bits and forwarded to the modulator’s input. The counter used to create the clock for the dataFSM is the same with the one that clocks the I / Q Channels block.

![Figure 5.3: QPSK bitstream generation](image)

5.1.6 Hardware Results

The QPSK modulator has been tested extensively on the nanoRTU board and the results are presented in this section. In figure 5.4, the yellow waveform shows the modulated signal, the bright blue one shows the I channel, the green shows the Q channel and the purple shows the byte transmission starting point. The results have been captured when the byte '10110001' was sent through UART. The following information which can be extracted from this figure are:

- We can see the serially transmitted data on the I and Q channels.
- The bitrate on the channels is 3.05 kb/s. Considering that the modulated signal is 12.207 KHz, we observe that it takes 4 cycles per each bit. Recall from the BPSK results that the bitstream which was 6.1 kb/s required 2 cycles per bit.
- Finally, we can observe that the amplitude of the modulated signal is a bit less than 400 mV. The highest amplitude of the modulated signal is achieved when the amplitude of the I channel is 217 and the amplitude of the Q channel is 219. The result is the modulated signal to have a max peak amplitude of 436. The DAC used has a reference voltage 3.3V. From the datasheet we can find that the output voltage can be calculated from the formula below

$$V_{out} = 3.3 \frac{DigitalValue}{4096}$$  \hspace{1cm} (5.3)
The result is a peak amplitude of 351 mV on the DAC's output. Exactly, in the same way can be explained why the lowest peak of the modulated signal never reaches 0. The lowest peak of the modulated signal is achieved when the amplitude of the I channel is 42 and the for the Q channel 35. That leads to an amplitude of 76. Using the formula above we find that at the lowest peak of the signal the voltage is 61 mV.

In figure 5.4 the phase shifts at the bit transitions in the channels are not so clear and for that reason a closer look to the scope in a different horizontal scale would be useful.

Figure 5.4: QPSK hardware results - I / Channels

In figure 5.4 the phase shifts at the bit transitions in the channels are not so clear and for that reason a closer look to the scope in a different horizontal scale would be useful.

Figure 5.5 helps in that direction and the following information can be extracted:

- When the information on the channels is '11', the modulated signal has a phase of $+\frac{\pi}{4}$ rad.
- Next, both channels transit simultaneously at '0', transmitting the information '00'. A phase shift of $\pi$ rad occurs leading to a modulated signal with phase $-\frac{3\pi}{4}$ rad.
Chapter 5. QPSK & OQPSK Modulators

5.2. OQPSK Modulation

- The Q channel next gets to ‘1’ when the I channel remains at ‘0’. The modulated signals shifts by \( \frac{\pi}{2} \) rad to the phase \( +\frac{3\pi}{4} \).
- The I channel goes to ‘1’ and Q to ‘0’. The modulated signal shifts again by \( \pi \) rad to the phase \( -\frac{\pi}{4} \).
- From the above transitions we can see that even we have four distinct phases, \( \pi \) rad phase shifts are still possible when both channels transit at the same time. This makes QPSK a non-constant envelope modulation scheme since we have these abrupt amplitude changes occurred because of large phase shifts. A significant improvement for this is introduced in OQPSK where only \( \frac{\pi}{2} \) phase shifts can occur leading to a more constant envelope modulation.

5.1.7 Synthesis Results

The QPSK modulator when synthesized as a project with all the hardware components together, consumed 864 core tiles out of 13824 achieving an utilization of 6%. However, all the components have been also synthesized separately in order to get a clear picture of their contribution to the total resources and the results are presented in table 5.2. The total resources when the components are synthesized separately are 995 core tiles.

<table>
<thead>
<tr>
<th>System component</th>
<th>Core cells</th>
<th>Total chip utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cosine generation</td>
<td>177</td>
<td>1%</td>
</tr>
<tr>
<td>Sine generation</td>
<td>160</td>
<td>1%</td>
</tr>
<tr>
<td>DA Controller</td>
<td>81</td>
<td>1%</td>
</tr>
<tr>
<td>Inverter (x2)</td>
<td>23</td>
<td>0%</td>
</tr>
<tr>
<td>Inverter Reg</td>
<td>16</td>
<td>0%</td>
</tr>
<tr>
<td>PLL</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Reset Block</td>
<td>2</td>
<td>0%</td>
</tr>
<tr>
<td>IQ Channels</td>
<td>4</td>
<td>0%</td>
</tr>
<tr>
<td>FA 8bit</td>
<td>16</td>
<td>0%</td>
</tr>
<tr>
<td>DAC Reg</td>
<td>9</td>
<td>0%</td>
</tr>
<tr>
<td>Counter</td>
<td>21</td>
<td>0%</td>
</tr>
<tr>
<td>UART Receiver</td>
<td>423</td>
<td>3%</td>
</tr>
<tr>
<td>Data FSM</td>
<td>32</td>
<td>0%</td>
</tr>
<tr>
<td>Data Register</td>
<td>8</td>
<td>0%</td>
</tr>
</tbody>
</table>

5.2 OQPSK Modulation

The OQPSK modulation scheme shares the same block diagram with QPSK since both modulations are based on the same concept. They both assign 2-bits per symbol and have the same bandwidth efficiency and bit error rate. Their only difference is on the I / Q channels component.

In QPSK modulation, \( \pi \) rad phase shifts can occur when both channels change simultaneously. Large phase shifts are good to be avoided because they lead to non-constant envelope signals which limit the operation of the power amplifier only in the linear region. On the other side the smaller phase shifts occur, the less abrupt changes on the amplitude which lead to more constant envelope signals.

OQPSK adds significant improvement to QPSK on that aspect since only \( \frac{\pi}{2} \) phase shifts can occur. The I / Q channels component is modified as shown in figure 5.6. The aim of the I / Q channels block in OQPSK is not only to forward the bitstream data to the channels but also also to avoid their transition happen at the same time. That means if a bit period on the channels is \( 2T \), the Q channel should be delayed by time \( T \). As a result, only one channel changes at a time and considering that in the constellation diagram the symbol to phase mapping has been done based on Gray encoding, only transitions between neighbouring regions can happen leading to \( \frac{\pi}{2} \) phase shifts.
In terms of hardware this can happen if we add one more register on the Q channel path as shown in figure 5.6. All registers are clocked with 3.05 KHz and a bit’s period on the channels is \( \frac{1}{3.05 \text{ KHz}} = 0.327 \text{ms} \). The first register on the Q path is clocked on the negative edge of the clock and the second one on the positive edge. That means the Q channel is always delayed by half a bit’s period comparing to the I channel.

### 5.2.1 Hardware Results

The bitstream generation hardware used for QPSK has been re-used in OQPSK in order to provide the bitsream to the modulator. The OQPSK modulator has been tested with many different data patterns and performed as expected. In this section the hardware results are presented. In order to compare them with QPSK they have been captured with the same pattern sent from the UART as QPSK (10110010). The yellow signal is the OQPSK modulated signal, the blue and green signals are the I and Q channels respectively, and the purple one is indication when the byte transmission starts.

From the results in figure 5.7 can be seen the delay on the Q channel and it can be verified that the
Chapter 5. QPSK & OQPSK Modulators

5.2. OQPSK Modulation

channels never change both at the same time. The transition from '11' to '00' which happened in QPSK, now happens in steps. The results of this on the phase transitions can be seen in figure 5.8 where we have more smooth amplitude changes than QPSK leading to a more constant envelope modulation.

![Figure 5.8: OQPSK hardware results - Phase shifts](image)

5.2.2 Synthesis Results

As expected the addition of one register can not create much difference on the hardware resources. The total consumed resources of the OQPSK modulator are 865 core tiles out of 13824 and an utilization of 6%. On the other side, when the each component is synthesized separately, the sum of the resources is 996 core tiles. The synthesis results are shown in table 5.3. The only difference with QPSK is observed on the I / Q Channels block.

<table>
<thead>
<tr>
<th>System component</th>
<th>Core cells</th>
<th>Total chip utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cosine generation</td>
<td>177</td>
<td>1%</td>
</tr>
<tr>
<td>Sine generation</td>
<td>160</td>
<td>1%</td>
</tr>
<tr>
<td>DA Controller</td>
<td>81</td>
<td>1%</td>
</tr>
<tr>
<td>Inverter (x2)</td>
<td>23</td>
<td>0%</td>
</tr>
<tr>
<td>Inverter Reg</td>
<td>16</td>
<td>0%</td>
</tr>
<tr>
<td>PLL</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Reset Block</td>
<td>2</td>
<td>0%</td>
</tr>
<tr>
<td>IQ Channels</td>
<td>5</td>
<td>0%</td>
</tr>
<tr>
<td>FA_8bit</td>
<td>16</td>
<td>0%</td>
</tr>
<tr>
<td>DAC Reg</td>
<td>9</td>
<td>0%</td>
</tr>
<tr>
<td>Counter</td>
<td>21</td>
<td>0%</td>
</tr>
<tr>
<td>UART Receiver</td>
<td>423</td>
<td>3%</td>
</tr>
<tr>
<td>Data FSM</td>
<td>32</td>
<td>0%</td>
</tr>
<tr>
<td>Data Register</td>
<td>8</td>
<td>0%</td>
</tr>
</tbody>
</table>

Table 5.3: OQPSK synthesis results
Chapter 6

AFSK & MSK Modulators

Two Frequency Shift Keying (FSK) modulations have been implemented on the nanoRTU hardware and in this chapter these implementations will be described in detail. The first modulation is the Audio Frequency Shift Keying (AFSK) which has been made to comply with the AFSK Bell 202 modem specifications. Even though it is an old standard, many CubeSat satellites still use it and for that reason it has been considered necessary to implement it. The second modulation is the Minimum Shift Keying (MSK) scheme which is continuous phase and more bandwidth efficient compared to AFSK and other FSK schemes.

6.1 AFSK Block Diagram

According to the Bell 202 specifications, the carrier switches between two frequencies with respect to the incoming binary data. In case of a logic '1' the carrier switches to a low frequency tone called mark and it is specified to be 1200Hz ±10Hz. In the other case of a logic '0' the carrier switches to a higher frequency tone called space which should be 2200Hz ±10Hz. This concept is relatively simple to implement efficiently in hardware considering that all needed are the frequency generation blocks that will synthesize the mark and space frequencies and a selection mechanism that will select between these two according to the data bitstream. In figure 6.1 the Block Diagram of the AFSK modulator shows the components and their interconnections which have been implemented to realize it in hardware.

![AFSK Block Diagram](image)

Figure 6.1: AFSK Block Diagram

6.1.1 Carrier Generation

An important part of the AFSK implementation is the generation of the mark and space frequencies. In the Block Diagram depicted in figure 6.1, the hardware components mark_generation_0 and space_generation_0 are responsible for this task. The method followed for wave generation is similar to
the previous chapters where a Look-Up table with pre-stored values has been used and a state machine which scans it properly to generate a full period of the sinusoidal carrier. However, in the AFSK we need two different frequencies and for that reason two separate Look-Up tables are necessary. Considering, that both state machines work under the same clock, in order to produce the two frequencies the quantization step needs to be changed for both of them. To achieve that, a new number of samples per quadrant had to be calculated to generate the required frequencies with as much accuracy as possible.

The fact that the mark and space frequencies are low, made their generation possible without the use of a PLL. The counter component counter_0 has been used to divide the 16 MHz clock provided from the crystal oscillator on the board and produce a 500 KHz clock for the state machines. For the mark tone a number of 104 samples per quadrant has been chosen. That implies 416 samples per period and considering that a sample is scanned every 2\(\mu\)s, 832\(\mu\)s are needed which makes a mark frequency of 1201.92 Hz. Similarly, for the space tone a number of 57 samples per quadrant has been chosen which makes 228 samples for a full period. That requires 456\(\mu\)s and leads to a space frequency of 2192.98 Hz.

The achieved frequencies for both mark and space comply with the ±10 Hz specification of the Bell 202 modem. As a final remark in this paragraph we can add that the design choice of one Look-Up table for both frequencies but different clock for the state machines has been rejected. The reason is that a small deviation on the state machine’s clock leads to a much bigger deviation on the generated signal’s frequency since the error is multiplied by the number of samples. In that case to achieve precise frequencies for mark and space two PLLs are required that will generate precise clocks for the state machines. With the method chosen, it has been easier to fine tune the mark and space frequencies to meet the Bell 202 specifications.

### 6.1.2 Frequency Selection

The generated frequencies are provided as inputs to a 2-input multiplexer which according to the logic level of the binary data on its selection line, will choose the proper one. The bitstream is provided to the multiplexer from the bitstream generation circuit which for AFSK is the same with the one described for BPSK (figure 3.10 since in both modulations the width of the symbol is 1-bit. Their only difference is the bitrate of the binary data which for AFSK is chosen to be 1200 bps. That requires to provide the dataFSM state machine with a 2400 Hz clock.

In that point it is useful to note some important thoughts about the bitrate choice. It has been mentioned in Chapter 2 that in FSK modulation schemes the frequency separation is critical. Recall that for signal orthogonality the frequency separation is given by

\[
 f_1 - f_2 = \frac{m}{2T} \tag{6.1}
\]

where \(\frac{1}{T}\) is the data bitrate and \(m\) is an integer. Three possibilities can be distinguished:

- \(m > 2\). In that case signal orthogonality is achieved but not phase continuity. The larger the \(m\) the less the bandwidth efficiency.

- \(m = 2\). In that case signal orthogonality and phase continuity is achieved. The frequency separation is equal to the bitrate.

- \(m = 1\). In this final case signal orthogonality is achieved but not phase continuity. However, the bandwidth efficiency is the best possible.

We have set the bitstream generation circuit to produce a 1200 bps signal to feed the modulator’s input for testing purposes. However, from (6.1) can be derived that for this bitrate the \(m\) is not an integer and takes the value 1.66 which means the lack of orthogonality and phase continuity. On the other hand, a bitrate of 1000 bps would lead to an \(m\) equal to 2 which is the second case and gives the optimum result. The bitrate has been chosen mainly for compatibility with the AFSK Bell 202 modems which mostly use 1200 bps for half-duplex communications. However, the modulator can work within all the three cases above and a variety of bitrates.
6.1.3 Hardware Results

The AFSK modulator has been tested on the nanoRTU following a similar testing procedure with the modulation schemes described in previous chapters. The user has the ability to send data through UART which are latched and then serially converted with the required bitrate. The testing results on the nanoRTU are shown in figure 6.2. The yellow waveform shows the modulated AFSK signal. The blue waveform is the binary data and the green waveform shows the start of the transmitted byte. When the results in figure 6.2 were captured, the byte '11001100' was sent.

![AFSK hardware results](image)

Figure 6.2: AFSK hardware results

The following conclusions can be extracted from the results shown in figure 6.2:

- We can see that the frequency of the carrier changes according to the bitstream as expected. A logic '1' results in low frequency (mark) and a logic '0' to a higher frequency (space).

- The result verifies that there is no phase continuity since in bit transitions abrupt phase shifts occur.

- In the right down corner of the scope’s screen we observe that the frequency of the green waveform is 150.24 Hz which actually shows the frequency which a full byte is transmitted. That means the bitrate of the binary data is 1201.92 Hz.

6.1.4 Synthesis Results

The AFSK modulator has been synthesized as a project and the total consumption of the hardware resources was 868 core tiles out of 13824 which gives a total chip utilization of 6%. The hardware blocks in the AFSK implementation have been also synthesized each one separately to get an idea about how much they contribute approximately to the total consumed resources. In table 6.1 the synthesis results are presented. It is interesting to point out that the mark generation component consumes more resources than the space generation because of the quantization issue which has been discussed. That implies larger Look-Up table which affects the synthesis results as expected.
6.2 MSK Block Diagram

The condition when the frequency separation is equal to the bitrate seems to be the optimum but it doesn’t achieve the best bandwidth efficiency. On the other side, a modulation with

\[ f_1 - f_2 = \frac{1}{2T} \]  

results in minimum separation, achieves signal orthogonality and the best possible bandwidth efficiency but lacks phase continuity. The MSK modulation scheme comes to solve this problem since with the minimum frequency separation manages to keep phase continuity.

In figure 6.3 is shown the Block Diagram of the implemented MSK modulator. Two different paths can be seen. One path is the mark and space generators, the multiplexer for frequency selection \((\text{mux})\), the inverter block \((\text{invBlock})\), the DAC register \((\text{Reg})\) and the Digital to Analog controller \((\text{DAController})\). Except from the inverter block, the rest of the components exist also in the AFSK implementation and they are performing the same task described. In MSK, the inverter block is inserted in the data path in order to change the phase of the carrier by \(\pi\) rad under specific conditions. The second path in the Block Diagram is a counter \((\text{Counter})\), the data delay block \((\text{dataDelay})\) and the phase control block \((\text{phaseControl})\). Purpose of the second path is to control the inverter block to change the phase of the carrier.

Before we proceed to the detailed description of the blocks, it is necessary to understand under which conditions the phase of the carrier should be shifted by \(\pi\) rad in order to achieve phase continuity. In Chapter 2 has been shown that the MSK modulated carrier is given by

\[ s(t) = \cos(2\pi(f_c + d_k \frac{1}{4T})t + \theta_k), \quad kT \leq t \leq (k+1)T \]  

where the phase \(\theta_k\), in order to achieve phase continuity, should be

\[ \theta_k = \begin{cases} 
\theta_{k-1} + \frac{\pi k}{2}(d_{k-1} - d_k) & d_k = d_{k-1} \\
\theta_{k-1} \pm \pi k, & d_k \neq d_{k-1} 
\end{cases} \]  

From equation (6.4) it is obvious that some kind of mnemonic behaviour is necessary for the modulator in order keep track of the previous incoming bit and phase. The phase status can be re-written based on (6.4) as following

\[ \theta_k = \begin{cases} 
\theta_{k-1}, & d_k = d_{k-1} \\
\theta_{k-1} \pm \pi k, & d_k \neq d_{k-1} 
\end{cases} \]  

From equation (6.5) is derived that when the current incoming bit \((d_k)\) is the same with the previous one \((d_{k-1})\), the phase of the carrier should not change. In the condition when two consecutive bits are different, the phase of the carrier should be shifted by \(\pi\) rad only when every odd bit is transmitted. The
delayData_0 block delays the input data by time $T$ and allows for the comparison of two consecutive bits each time instant $kT$. The Counter_0 block indicates when an odd bit is transmitted and provides this information to the phaseControl_0 block which according to the information for the values of the previous and the current data bit decides if a phase shift is needed to the carrier. Finally, the phaseControl_0 block controls the invBlock_0 block which does an amplitude inversion in order to generate a $\pi$ rad phase shift.

![Figure 6.3: MSK Block Diagram](image)

### 6.2.1 Carrier Generation

The *mark* and *space* generators follow the same method as in AFSK to generate the two different frequencies. In the MSK implementation the bitrate that has been chosen is 6200 bps in order to comply with the requirement for at least 4800 bps for Telemetry data. It has not been able to make it exactly 6103 bps as in the phase modulations and for that reason 6200 bps has been the closest bitrate for which the *mark* and *space* frequencies could be generated with good accuracy. The minimum frequency separation according to equation (6.2) results in the following conclusion for the *mark* and *space* frequencies

$$f_{space} = f_c + 1.55\, KH\,Hz$$

$$f_{mark} = f_c - 1.55\, KH\,Hz$$

(6.6)  
(6.7)

where 1.55 KHz is the frequency deviation $\Delta f$. The only thing that remains to be specified is the center frequency $f_c$ that will help to calculate the values for $f_{space}$ and $f_{mark}$. The center frequency is given by

$$f_c = \frac{f_{space} + f_{mark}}{2}$$

(6.8)

and for orthogonality must be an integer multiple of $\frac{1}{2T}$. In our implementation we selected the center frequency twice the bitrate (12.4 KHz). That means the $f_{mark}$ is 10.85 KHz and the $f_{space}$ is 13.95 KHz.
The frequency generator blocks are clocked with 3.125 MHz. For the mark frequency a Look-Up table with 72 samples per quadrant is used which makes 288 samples for a full period. That results in a sinusoidal signal with frequency 10.8507 KHz. Similarly, for space frequency a Look-Up table with 56 samples per quadrant is used which makes 224 samples per period. This results in a sinusoidal signal with frequency 13.9509 KHz. It can be verified that the separation between these two is 3.1 KHz, exactly half the bitrate.

6.2.2 Frequency Selection

Frequency selection is performed in the same way as in AFSK. A multiplexer is used to propagate the proper frequency according to the bitstream. One difference with the AFSK implementation is that here a logic '1' selects the space frequency and a logic '0' the mark. Generally, there is no strict requirement for that. However, considering MSK, it is widely used like this in contrast to the Bell 202 modem specifications were it is the opposite.

6.2.3 Phase Control

The first component on the phase control path is the Counter. The task of the Counter is to perform clock division on the 3.125 MHz clock. Result of this, is the generation of three clocks. The first one is a 12.4 KHz clock which is used for the bitstream generation circuit and more specific for the dataFSM state machine. It is not shown on the Block Diagram of figure 6.3 since it is part of the testing circuit. The second one (phClk) is a 6.2 KHz clock which is used in the dataDelay block which delays the incoming bitstream by time T. The third (n) is a 3.1 KHz clock which is used to indicate if the current bit is the odd or the even. A logic '1' means that it is the even bit and a phase change is allowed. Similarly, a logic '0' indicates that it is the odd bit and a phase shift is allowed.

Knowing if the k is an odd or even number in equation (6.5) the only thing that remains is to find the status of the previous incoming bit to see if the phase needs to be changed or not. That task is handled by the delayData block and its implementation is depicted in figure 6.4. Two registers are used to latch the incoming data but they are triggered at different edges of the clock. That ensures in every time instant kT that both the current bit $d_k$ and the previous $d_{k-1}$ will be known.

Figure 6.4: Data delay

The last block in the path, the phaseControl takes as input the current and the previous data and according to the bit indication n it produces the signal which controls the inverter. In figure 6.5 is shown the circuit that realizes the phaseControl. An XOR operation between the current and the previous data checks if two consecutive bits have the same value. If they have the same value then the selection line of the multiplexer is set to logic '0' and propagates the previously stored value which means that there is not any phase change. In case two consecutive bits are different, the XOR operation will produce a logic '1' which leaves the phase shift decision to the n signal. In case of odd number transmitted bits, the n signal is in logic '0' and the selection line of the multiplexer is set to '1'. That results in the propagation of the inverted stored value from the multiplexer which leads to a phase shift. In the other case of even number transmitted bits, the selection line is set to '0' and the previous output of the multiplexer is again propagated which means that there is no phase change.
The multiplexer’s output controls the inverter block. A logic '1' forwards the carrier without any change and a logic '0' with a phase shift of $\pi$ rad. The phase shift is implemented as in BPSK modulation where an inverter block was used to invert the carrier’s amplitude.

![Phase control](image)

**Figure 6.5: Phase control**

### 6.2.4 Hardware Results

The implemented MSK modulator has been tested in the hardware and the results which verify its correct operation have been collected and presented in figure 6.6. The yellow waveform shows the MSK modulated signal. The blue waveform shows the bitstream, the purple one shows the control signal to the inverter and the green one shows when the byte transmission starts. When the results were captured the data '11011000' was sent from the UART. The bitstream generation circuit for AFSK has been re-used in MSK with the difference that now the dataFSM is clocked with 12.4 KHz in order to produce a 6.2 KHz bitstream.

![MSK hardware results](image)

**Figure 6.6: MSK hardware results**

The following conclusions can be derived from the analysis of the results:

- We can see that the frequency of the carrier changes with the bitstream. However, in contrast with the AFSK modulation, here abrupt phase shifts don’t occur leading to phase continuity. Minimum
separation can also be observed since the difference between the two frequencies can be hardly detected.

- The blue waveform shows the control signal to the inverter. It can be seen that it never changes when two consecutive bits are the same.

- Finally, the frequency of the green waveform is 775.046 Hz as can be seen on the right down corner of the scope. It represents the frequency at which a byte is transmitted. That means the data bitrate is 6200 bps as expected.

### 6.2.5 Synthesis Results

The MSK implementation even thought it is a bit more complex than the AFSK, the hardware resources it needs are very similar. The reason for this is that in MSK we used smaller Look-Up tables because we needed higher *mark* and *space* frequencies. The complete MSK project when synthesized, 874 core tiles were occupied on the FPGA device leading to an utilization of 6%. In table 6.2 the consumed resources, when each component synthesized separately, are shown.

<table>
<thead>
<tr>
<th>System component</th>
<th>Core cells</th>
<th>Total chip utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark generation</td>
<td>208</td>
<td>2%</td>
</tr>
<tr>
<td>Space generation</td>
<td>155</td>
<td>1%</td>
</tr>
<tr>
<td>Mux</td>
<td>8</td>
<td>0%</td>
</tr>
<tr>
<td>Inverter</td>
<td>23</td>
<td>0%</td>
</tr>
<tr>
<td>DA Controller</td>
<td>81</td>
<td>1%</td>
</tr>
<tr>
<td>DAC Reg</td>
<td>8</td>
<td>0%</td>
</tr>
<tr>
<td>Counter</td>
<td>45</td>
<td>0%</td>
</tr>
<tr>
<td>Delay data</td>
<td>2</td>
<td>0%</td>
</tr>
<tr>
<td>Phase control</td>
<td>3</td>
<td>0%</td>
</tr>
<tr>
<td>Reset block</td>
<td>2</td>
<td>0%</td>
</tr>
<tr>
<td>UART Receiver</td>
<td>423</td>
<td>3%</td>
</tr>
<tr>
<td>Data FSM</td>
<td>32</td>
<td>0%</td>
</tr>
<tr>
<td>Data Register</td>
<td>8</td>
<td>0%</td>
</tr>
</tbody>
</table>
Chapter 7

BPSK Demodulator

The Telecommand RF frequency received from the spacecraft’s RF Front-End receivers, is converted to a Low-IF frequency which needs to be demodulated in order to retrieve the digital data back. This is the demodulation process which for space and avionic applications is more complex because of the high receiver speeds relative to the transmitter and the Doppler effect needs to be taken into consideration in the design process. In this thesis work a complete BPSK demodulator has been implemented. The demodulator has been tested with the BPSK modulator described in Chapter 3. The requirements for the Telecommand data were to achieve a bitrate above 1200 bps and for that reason the BPSK modulator has been modified, for testing purposes, to transmit a 7.812 KHz carrier which “carries” the digital information with bitrate 1953 bps. The implemented receiver has the ability to demodulate the incoming signal which can vary between a ±10% frequency range from the expected one. This range may be slightly changed when the RF Front-Ends are considered also in the system, but for the needs of the thesis work it is a very good estimation that will demonstrate the receiver’s capabilities. The whole modem has been made to be radiation tolerant following the triple module redundancy technique.

7.1 Carrier Recovery

In terrestrial applications where there is no motion between the receiver and the transmitter or in cases their relative speed is very low, the frequency of the transmitted signal arrives as is in the receiver. The receiver in these conditions is simple since knowing the frequency of the received signal with a matched local oscillator we can retrieve the baseband signal with down mixing. However, in space communications the very high speeds of the receiver make the received signal’s frequency vary between a range and special tracking loops are necessary in order to recover the carrier. The recovered carrier will continuously track the frequency of the incoming signal in order for their down mixing to provide the baseband signal. A demodulation which is based on the recovered carrier is called coherent. In our application the Costas Loop has been chosen to handle the task of carrier recovery. The loop generates a carrier which tracks the received signal’s frequency and performs data demodulation. An important characteristic of the Costas Loop is its sensitivity which is twice that of a conventional PLL and this makes it a suitable solution for the Doppler effect issues.

7.1.1 AD Controller

The first stage in the demodulator is the conversion of the analog Low-IF frequency to digital for further processing. The nanoRTU is equipped with a TLV2548 Analog to Digital Converter from Texas Instruments and for the needs of the thesis a hardware controller has been implemented that initializes, configures and controls it.

The TLV2548 AD Converter has advanced capabilities but also a more complex controlling sequence compared to the DA converter used in the Modulators. Detailed information about its characteristics can be found in [24]. The DA controller performs 3 main operations. It initializes the device writing the value A000h after power up, it writes the configuration data to the device which decide its functionality and
reads the digital data which have been converted from the device. In figure 7.1 is shown the transition diagram of the controller’s state machine.

The first operation after power-up is the initialization where the value A000h should be written to the device. The state machine is on the \textit{init} state where the \textit{selk} is '0' and the \textit{c\$} is '1'. The \textit{selk} is the clock signal provided to the device from the controller and \textit{c\$} is the signal which in its falling edge initiates a data transaction. When the reset is de-asserted, the state machine transits to the \textit{wait1} state where the \textit{c\$} gets to '0' and starts the process for the transaction of the initialization value. Next is the \textit{config1} state where the \textit{selk} toggles to '1' and the first bit is transmitted (the MSB of the register that contains the initialization value). In the next state, \textit{config2}, the \textit{selk} toggles to '0' and the data are shifted one position left. The state machine transits again back to \textit{config1} and in the rising edge of \textit{selk} the MSB is sent. This repetition between the states \textit{config1} and \textit{config2} occurs 15 times until all the bits of the register are shifted to the device.

The next operation is the device configuration where the controller sends a proper sequence to the device in order to configure its functionality. This sequence is a 12-bit word which is pre-determined according to the application requirements based on the information in the device’s datasheet [24]. For the demodulator the device configuration data are the following:

- **Bit 11.** It is set to '1' since on the nanoRTU layout there is no external voltage reference. Setting this bit high enables the internal voltage reference.
- **Bit 10.** The TLV2548 has internally two voltage references. One is 2V and the other 4V. Setting this bit '0' selects the 4V voltage reference and this is the option that has been chosen. The reason for this will become clear later when the Costas Loop is discussed, so we will come back to this.
- **Bit 9.** We set this bit '0' to select a short sampling period of 12 clock cycles.
- **Bits 8,7.** We set these bits to '01' in order to select the external clock \textit{selk} provided from the controller as a conversion clock.
- **Bits 6,5.** These bits are set to '00' in order to select the single shot mode. The TLV2548 has 4 modes in total. Modes like sweep mode or repeat sweep mode are not used since we need only one channel of the device’s inputs.
- **Bits 4,3.** These ones are don’t cares in single shot mode. They don’t have any effect.
- **Bit 2.** Setting this bit to '1' makes the End-of-Conversion (EOC) pin indicate when conversion has started and when finished. It is an important signal which as it will seen later it is monitored through the state machine.
- **Bis 1,0.** These ones are don’t cares in single shot mode. They don’t have any effect.

After \textit{config2}, the state machine transits to \textit{wait4} where the \textit{c\$} gets to '1' to terminate the data transaction of the initialization value A000h. In the next state, \textit{wait4b}, the \textit{c\$} gets to '0' to initiate the transaction of the configuration data. The configuration data are loaded to a 16-bit register, where the 4 MSBs contain the value '1010' and the 12 LSBs the configuration data. Next, the state machine transits to \textit{read1} where the MSB of the register which contains the configuration data is sent to the device. The \textit{selk} in this state toggles to '1'. At the same time the data from the device are read and they are stored in a register. However, in this stage the data read are invalid since the device has not been configured yet.

The state machine next transits to \textit{read2} where the \textit{selk} toggles to '0' and the contents of the register contain the configuration data are shifted one position left. The repeated transition for 15 times between the states \textit{read1} and \textit{read2} has as a result the configuration data to be sent to the device and to read an erroneous 12-bit data value from it. After the configuration data have been sent, the state machine from \textit{read1} transits initially to \textit{wait3} where the \textit{selk} toggles to '0' and then to \textit{wait4} where the \textit{c\$} get to '1' in order to the terminate the transaction of the configuration data.

Final task of the AD Controller is to read the digital values from the selected channel of the device. After state \textit{wait4} the state machine transits to \textit{wait4b}. This time in state \textit{wait4b} the 4 MSBs of the 16-bit register are loaded with the value '0000' which selects to read the analog channel A0 of the device. The logic state of the remaining 12 bits doesn’t play any role. It can not change the configuration of the
7.1. Carrier Recovery

device because the 4 MSB values need to be '1010' to write in the device's CFR register. From state wait4b the state machine again does the repeated transition between read1 and read2 in order to select the channel which needs to be converted and receives the digital data from the device. After read1 state, there is a time period that the device needs to convert the analog value to digital and this period starts from the falling edge of the EOC signal and ends with the rising edge of it. It is important before we terminate the transaction to ensure that the conversion for the selected channel has been finished. For that reason, the four states int_wait1, int_wait1b, int_wait2 and int_wait2b monitor the EOC signal with the clock transitions. When the conversion finishes, the state machine transits back to the states wait3 to set the sclk to '0' and then in wait4 to set the cs to '1' to the terminate the transaction.

![AD Controller state machine](image)

Figure 7.1: AD Controller state machine

7.1.2 Costas Loop Block Diagram

The conversion of the received signal to digital from the AD Converter makes possible its processing to be done all digitally in the FPGA. The Costas Loop that has been implemented is shown in figure 7.2 with all the interconnections between the hardware components [25, 26]. The core of the loop is a Numerically Controlled Oscillator (NCO) which generates two orthogonal sinusoidal signals in a frequency identical or very close to the one expected, in our case 7.812 KHz. These signals are multiplied with the digitized received signal. The multipliers are called Phase Detectors and their product is a signal with mainly two frequency components, a low frequency component which can even be zero when the NCO’s frequency match the incoming signal’s and a high frequency component which is the sum of their frequencies. Two FIR Filters are properly designed to reject this high frequency component and propagate only the low frequency one. In the condition when the NCO tracks the incoming signal, from the upper channel we can retrieve back the Telecommand data. However, in the other case when the NCO’s frequency is different from the incoming signal’s an error signal produced by the Error component, which controls the NCO in order bring the loop in lock condition. Once the loop locks, it will continuously track the frequency variations of the incoming signal by proper adjustments of the NCO’s frequency.

7.1.2.1 Phase Detectors

The Phase Detectors are two multipliers which have as inputs the generated reference signals from the NCO and the digitized received signal. Even though we will refer to the NCO implementation in a
later subsection, we note here that the NCO signals are 8-bits wide and their amplitude lays in the range \([-127, 127]\). In contrast to the modulators, where the amplitude of the generated waveforms was between the range \([1, 255]\), here we remove that DC offset in order to avoid multiplication of signals with DC offsets that could lead to signals with larger bit widths. Except from that, in the modulators the DC offset was necessary in order to transmit the modulated signal properly due to the fact that the DA converter was unipolar and couldn’t convert negative values. The multiplication was also done efficiently with simple amplitude inversion and the DC offset did not have any negative impact on that.

In the Costas Loop the multiplication can not be seen as amplitude inversion as in the modulators and for that reason it is better to be done between signals with zero DC offset. That of course refers also to the received signal. The ADC on the nanoRTU is also unipolar and the digital values it produces from the conversion are positive. During the configuration phase of the converter we have set the Bit 10 to ‘0’ in order to select the internal 4V voltage reference and since the ADC is powered with a 3.3V supply the internal reference voltage is also set to 3.3V. This is not generally a good practice because the reference voltage does not have the ability to stabilize the voltage but depends on the power supply. However, the advantage of this choice is that the values produced by the ADC are also lying between the range \([1, 255]\) since both the DA and AD converters are 12 bits and have the same reference voltages. That helped a lot during the hardware development of the loop. In a next step the DC offset is removed with subtraction with 128 in order to bring the amplitude range of the incoming signal to \([-127, 127]\).

It is important to point here that the Costas Loop would work the same even if we used the 2V reference voltage. In that case the received signal would be digitized with an amplitude range between \([1, 420]\). The reason is because the amplitude of the analog signal is between the voltage peaks \([0, 205 \text{ mV}]\). In that case to remove the DC offset a subtraction with 210 would be appropriate which would lead to an amplitude range \([-209, 209]\). Another reason the 4V voltage reference has been chosen is that less hardware resources are required due to the smaller amplitudes of the digitized signal.

In wireless communications the above are not much of a concern. It is not necessary to know in the receiver the conversion characteristics of the DAC in the transmitter. However, it is very important to know the analog voltage range of the receiver Front-Ends for the proper interface with the Costas Loop. Since this voltage is not guaranteed to be stable, usually an Automatic Gain Control (AGC) circuit is used to keep it stable in the input of the AD converter. That ensures the converted signal after subtraction will have zero DC offset under any condition. In this thesis work the AGC circuit has not been considered since it depends very much on the RF Front-Ends characteristics. The implemented Costas Loop has been made to demodulate signals with an amplitude of 205 mV, for easy testing with the modulator, but it can be easily adapted to any type of Front-End receiver once its characteristics are known.
Chapter 7. BPSK Demodulator 7.1. Carrier Recovery

Regarding the implementation of the Phase Detectors two multiplier cores have been used from the IP Library of the Libero Software tool. The multipliers have been configured for 8-bit input signals and an output of signed representation. The multiplier IP is an efficient way for the multiplication of signed numbers using Booth’s multiplication algorithm. The multiplication product will be in the range \([-16129, 16129]\) which means that it can be represented with a 15-bit signal and for that reason the 16th bit can be truncated. The truncation is done in the in the Reg_0 component which is a register. Except for the truncation, the register ensures a stable value in the filter’s input since the multipliers are combinational circuits and their output update depends mainly on the propagation delay of their internal components.

From the functional point of view the Phase Detectors multiply the NCO generated signals with the digitized received signal which lies in the frequency range \([7.031 \text{ KHz}, 8.593 \text{ KHz}]\) due to the Doppler effect. For the upper (In-Phase) channel the NCO signal is a cosine with frequency 7.812 KHz. For the down (Quadrature) channel the NCO generated signal has a phase of \(\frac{\pi}{2}\) rad compared to the In-Phase channel. The reason is because they need to be orthogonal. Two cases can be distinguished. The first is when the incoming received signal \((f_1)\) is in phase and has the same frequency as the NCO’s cosine signal \((f_2)\). This case can be seen in the simulation results in figure 7.3. The Phase Detector I, which shows the product of the I channel multiplication, contains a high frequency component equal to \((f_2 + f_1)\) or \(2f_1\) and a zero frequency component. This is the ideal case in the Costas Loop since with a digital filter rejecting the high frequency we can retrieve our baseband signal.

![Figure 7.3: Phase Detector simulation results](image)

However, in the other case where there is a small difference between \((f_1)\) and \((f_2)\) the low frequency component is not zero any more. The result of this can be seen in figure 7.4. The Phase Detector I now has two frequency components and the data retrieval is not possible with filtering. More specific, for testing purposes, we set the NCO to 7.031 KHz (the -10% Doppler range) and considering the 7.812 KHz signal received from the modulator the high frequency is 14.843 KHz and the low 781 Hz. We can see both frequencies in figure 7.4.

![Figure 7.4: Phase Detector simulation results - Frequency and Phase mismatch](image)

The spacecraft motion produces considerable Doppler shift which leads to the problem in figure 7.4 which makes the data retrieval not possible. The Costas Loop aims to adjust the NCO generated signals in that way to match them in frequency and phase with the received signal and recover the data as shown in figure 7.3.
7.1.2.2 FIR Filters

The FIR Filters in the Costas Loop implementation have a dual role. Their primary use is to retrieve the Telecommand data when the tracking condition of the received signal from the NCO is achieved. We will refer to this condition as a lock condition for the rest of the thesis. Their second role is to provide the error multiplier with a clean signal that will later result in the error signal which will control the NCO to bring the loop into the lock condition. For that reason it is important to specify the requirements for the filters before we proceed to their implementation details.

The Filters in the I and Q channels should be identical for optimum performance of the Loop. This is one requirement which in a digital implementation can be easily achieved. The second requirement is related to their pass and stop frequencies and their attenuations. To establish those, we need to calculate the possible frequency components after multiplication in the Phase Detectors. Considering that the NCO is initially set to 7.812 KHz with the ability to track the received signal which varies through the ±10% range, the high frequency component can be between the frequency range (14.062 KHz, 17.186 KHz) under the lock condition. In the lock condition, this high frequency component needs to be rejected from the filter and for that reason the frequency 15.7 KHz which is in the middle of this range has been chosen as a stop frequency. The filters should propagate the baseband signal which in our application has a bitrate of 1953 bps. We select for that reason a pass frequency of 1.953 KHz.

The selection of the attenuations for the pass and stop frequencies is not straightforward and it has been an iterative process because it had a significant impact on the NCO design. To select the proper attenuations and specify the Filter coefficients, the Matlab Filter Design and Analysis tool (FDA Tool) has been used which helped a lot in that direction. In figure 7.5 the Graphical User Interface of the tool is shown with the parameter settings.

![FDA Tool settings](image)

Figure 7.5: FDA Tool settings

Two of the parameters we have already referred to are the pass and stop frequencies and they can be seen in the Frequency section in the Matlab tool. The choice of the sampling frequency $F_s$ has been done primarily having in mind the NCO which initially produces a 7.812 KHz signal with the Look-Up table method and uses 64 samples per period. That means it is is clocked with 500 KHz. In the Costas Loop Block Diagram this frequency is generated from the Counter block which performs clock division by 32 on the 16 MHz clock. A higher sampling frequency for the filter is not necessary and it will increase the order of the filter leading to a larger implementation. On the left side of the tool the main Filter characteristics are shown like the Response and the Design. The response of the filters should be Lowpass since we need to pass through the baseband signal and the design has been chosen to be equiripple. The choice of the magnitude attenuation has been done iteratively in order to optimize the NCO design. The
attenuations of 4 dB and 34 dB were set for the pass and stop frequencies, respectively.

With the above parameters a 31 order FIR filter is suggested from Matlab which generates the filter coefficients. The 32 filter coefficients originally generated from the tool are real numbers and in order to be used in the design they need to be properly represented in fixed-point format. There are 3 main fixed point formats the tool supports for coefficient generation, the Q31, the Q15 and the Q7. The Q31 has been rejected since it would require a significant amount of hardware resources and the multiply-accumulate structure of the filter would suffer from bit explosion because it represents a coefficient as a 32-bit number where the MSB is the sign and the rest 31 bits are the fraction. A design decision needed to be taken between the Q15 and Q7 fixed point formats, where the Q15 was offering more accurate response and the Q7 smaller datapath reducing the hardware resources usage. Finally, the Q15 has been chosen which with the magnitude attenuation values mentioned before leads to the filter’s actual response shown in figure 7.6.

![Figure 7.6: FIR Filter response](image)

In the actual filter’s response two points of interest are shown. The first is the magnitude attenuation of the baseband signal which is 6.37 dB and the second is the attenuation of the high frequency component which is 26.8 dB. Note here that the attenuation of the high frequency component, which can be in the range (14.062 KHz, 17.186 KHz), can slightly vary according to the incoming signal. The amplitude of the high frequency component on the filter’s output is approximately 22 times smaller compared to the filter’s input which makes it negligible. On the other hand, the baseband signal is only half in the filter’s output which makes it dominant over the high frequency component. In the section about the Error multiplier we analyze how these amplitudes affect the loop.

To implement the datapath of the FIR filter and select the proper bit width for its multiply-accumulate structure two things are important to know. The first is the input data, which is a 15-bit signal varying from (-16129 to 16129) and produced from the Phase Detectors. The second is the coefficients which are 16-bit wide since we went for the Q15 fixed point representation. In figure 7.7 on the left, is shown the theoretical analysis of the datapath. The multiplier needs to be 31-bit in which the last 15 bits are the fraction. The input has no fraction and all comes from the coefficients. The multiplier product is added to the accumulator which is also 31 bits wide. For an FIR result this multiply-accumulate process needs to be repeated 31 times. Considering that the accumulator is expanding by 1-bit every $2^n$ additions where $n = 0, 1, 2, ..., ∞$ in 31 repetitions it will expand 5 times leading to a final result of (36.15). However, as we will see next a (31.15) representation, as shown in figure 7.7 left, is still sufficient since the coefficient values do not fully utilize the 16 bits of the signal.

A datapath like this would require a considerable amount of hardware resources and except from that, produces a 17-bit wide output which will be multiplied later to calculate the error. It is necessary to optimize the datapath as much as possible and reduce the bit width of the output signal for an efficient implementation.
Chapter 7. BPSK Demodulator

7.1. Carrier Recovery

From table 7.1 the first 16 coefficients can be seen. The other 16 are the same with those since the FIR filter has symmetric coefficients. All of them are positive and the largest one has the value 875 which can be represented with a 10-bit wide signal. That means the remaining 6-bits are '0'. This has as a result the implementation shown in figure 7.7 on the right. The multiplication of a 10-bit signal with the input data would require a smaller multiplier with a 25 bits output and from these 25 bits the last 15 represent the fraction. The multiplier’s output is added in the accumulator which after 31 iterations is expanded to 30 bits (30.15). Now the coefficients utilize better the 10-bit signal and for that reason the accumulator expansion is necessary. However, with hardware simulation we find out that even a 29-bit accumulator can be sufficient which probably happens because some of the coefficients are small enough to cause an overflow. The rule we referred before for bit expansion in every $2^n$ addition is actually taking into consideration the worst case scenario.

<table>
<thead>
<tr>
<th>Table 7.1: Filter coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Coefficients</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>0.026702880859375</td>
</tr>
<tr>
<td>0.008026123046875</td>
</tr>
<tr>
<td>0.00909423828125</td>
</tr>
<tr>
<td>0.010223388671875</td>
</tr>
<tr>
<td>0.011322021484375</td>
</tr>
<tr>
<td>0.01239013671875</td>
</tr>
<tr>
<td>0.013427734375</td>
</tr>
<tr>
<td>0.014404296875</td>
</tr>
<tr>
<td>0.01531982421875</td>
</tr>
<tr>
<td>0.01617431640625</td>
</tr>
<tr>
<td>0.016937255859375</td>
</tr>
<tr>
<td>0.017547607421875</td>
</tr>
<tr>
<td>0.018096923828125</td>
</tr>
<tr>
<td>0.018524169921875</td>
</tr>
<tr>
<td>0.018768310546875</td>
</tr>
<tr>
<td>0.0189208984375</td>
</tr>
</tbody>
</table>

The above datapath analysis has been tested mainly with simulations. In addition, it has been also verified with Matlab. Selecting a 15-bit input and a Q15 fixed-point coefficient representation, the tool produces the following output report:

Figure 7.7: FIR Datapath analysis
% Discrete-Time FIR Filter (real)
% -----------------------------------
% Filter Structure : Direct-Form FIR
% Filter Length : 32
% Stable : Yes
% Linear Phase : Yes (Type 2)
% Arithmetic : fixed
% Numerator : s16,15 -> [-1 1)
% Input : s15,0 -> [-16384 16384)
% Filter Internals : Full Precision
% Output : s29,15 -> [-8192 8192) (auto determined)
% Product : s25,15 -> [-512 512) (auto determined)
% Accumulator : s29,15 -> [-8192 8192) (auto determined)
% Round Mode : No rounding
% Overflow Mode : No overflow

7.1.2.3 Error Multiplier

The I and Q Lowpass FIR filters provide a clean output to the error multiplier which generates the error signal that will control the NCO to bring the loop in lock condition. The multiplier has been implemented using the multiplier IP from the Libero Software IP library. It has been configured to have two 14-bit signals as inputs and produce a 28-bit product. The 14-bit multiplier inputs are what remains from the accumulator’s output of the filters if we truncate the 15 LSBs which represent the fraction.

In order to understand the error signal and the values which it can take, we examine three cases. In all of them the received signal from the modulator is 7.812 KHz since we don’t have the ability to stretch it to simulate the Doppler effect. For that reason, in order to test the ability of the loop to lock we set the NCO to different initial frequencies. In the first case the NCO is set to 7.812 KHz. In this condition the only thing the loop needs to do is to bring in phase the NCO’s I channel with the received signal since their frequencies are the same. The Q channel will always have a $\frac{\pi}{2}$ rad phase difference. This case is depicted in figure 7.8.

![Figure 7.8: Error in lock condition when NCO is set to 7.812KHz](image)

It is important to focus on two things in the figure. The first is the phase of the NCO generated signals. When the received signal’s and the NCO’s frequency are the same, nothing guarantees that they will be also in phase. The Costas Loop with small fluctuations in the error signal attempts to bring them in phase. The second important thing is the error signal. Once the received signal and the NCO match in phase, the error signal goes to 0 in order to keep the same NCO frequency. This is what figure 7.8 depicts. As can be seen, what brings the error to 0 is the product of the Phase Detector Q which has a peak to peak range of (-7990 to 7990). After filtering, the high frequency component is removed and remains only the mean value of it which in this case is 0. On the other side, the product of the Phase Detector I has a peak to peak range of (36 to 16129) and (-36 to -16129) according to the information in the modulated
signal. That makes a mean value of 8082 and -8082, respectively. After filtering, we expect the baseband to be attenuated and appear in the filter’s output with a decreased magnitude by half. We expect then a baseband signal in the filter’s output with an amplitude switching between the peaks (-3885 to 3885). The actual results are very close to the theoretical ones. The I channel in the input of the error multiplier changes between the peaks -3585 and 3584. The small difference with the theoretical can be explained as follows. A digital bitstream with a bit period \(T_b\) can be proven mathematically that it has an infinite bandwidth. Practically that means that infinite number of frequencies are required to shape it. However, it is common practice to specify its bandwidth as \(f_b = \frac{1}{T_b}\) since the frequencies which contribute mostly to its shape and amplitude are located between (0 to \(f_b\)). In that way selecting \(f_b\) as the pass frequency in our filter results in a high attenuation of the frequencies from \(f_b\) to \(\infty\) that has an effect on the signal’s amplitude.

The second case is when the initial frequency of the NCO is set to 7.031 KHz. That means the error should have a value that will direct the NCO to change its frequency to 7.812 KHz and of course match it in phase. Here we see the value of the error under the lock condition and in the section where we refer to the NCO we describe how it operates. Figure 7.9 shows this case where the error stabilizes around a mean value of 3,359,660. Practically it is very hard and almost impossible to achieve a perfectly stable error signal with a reasonable amount of hardware resources. However, in most of the cases an error with small fluctuations around a selected mean value provides sufficient results. As can be seen in figure 7.9, the difference with the previous case is that the Phase Detector Q product has a positive mean value instead of 0. The signal’s peak to peak range is (-5776 to 10404) and (5776 to -10404) which makes a mean value of 2314 and -2314, respectively. Considering the baseband attenuation of the filters, we expected a signal with a peak to peak value of (-1112 to 1112). The actual results show that the Q channel in the filter’s output switches between the peaks (-971 to 971). The I channel on the other side switches between (-3460 to 3460). The multiplication of the two channels leads to a positive error that will direct the NCO to increase its frequency.

![Figure 7.9: Error in lock condition when NCO is set to 7.031KHz](image_url)

In the final case the NCO is set to 8.531 KHZ. The error now should direct the NCO to decrease its frequency and that will happen only if it is negative. This case is shown in figure 7.10. The error now stabilizes to -3,574,060. The reason for that again is the Q channel which is negative and switches between the peaks (-1042 to 1042).

One thing that is important to mention here is related to the phase of the NCO’s I channel and the received signal. We observe in figures 7.9 and 7.10 that the cosine signal has a phase of \(\pi\) rad with respect to the received signal. That seems to contradict in a first view with our statement that the loop tries to bring them in phase. However, as we proved in chapter 2, the loop locks when the formula

\[
\frac{1}{8} \sin 2(\phi - \theta)
\]

is equal to 0. The \(\phi\) is the phase of the received signal and \(\theta\) the phase NCO’s cosine signal. From that we derive that lock condition can be achieved only when their phases are matched or when they differ by \(\pi\) rad.
7.1.2.4 Numerically Controlled Oscillator (NCO)

The Numerically Controlled Oscillator (NCO) is used to generate two orthogonal signals. Based on the error signal produced from the error multiplier described above, it adjusts its frequency properly to match in frequency and phase the received signal. The NCO consists of three main hardware components as shown in figure 7.11. Two of them, $\text{sine\_generation\_demodulator}$ and $\text{cosine\_generation\_demodulator}$, are the Look-Up tables with their proper state machines which are used to scan them and generate the sinusoidal signals. The Look-up tables contain 16 samples per quadrant which make 64 samples for a full period. In order to generate a 7.812 KHz signal, a 500 KHz clock is required for the state machines. Even though this clock is easy to generate with a counter by frequency division, considering that our main clock is 16 MHz, the fact that the NCO frequency needs to vary in the range (7.031 KHz to 8.531 KHz) makes this solution inappropriate. The reason is that for a 7.031 KHz signal a 450 KHz clock would be required and for 8.531 KHz a 550 KHz one. It becomes obvious that for the limits of the Doppler range a maximum frequency variation of $\pm 50 \text{ KHz}$ is necessary to achieve in the clock of the state machines. That makes necessary a mechanism that will allow for fine tuning this clock to achieve each time the required NCO frequency.

The most suitable implementation to achieve precise adjustment of the clock is using a Phase Accu-
mulator. The Block Diagram of the implemented Phase Accumulator is shown in figure 7.12. The main hardware block of it is a 32-bit Accumulator which is used to generate a new period of clock each time it overflows. Its output is forwarded back to its input for addition on every rising edge of the main clock (16 MHz). For that reason, it becomes essential to establish an initial value for the accumulator and try to relate it to the error. Finally, a gain block is necessary to amplify the error signal in order to increase the speed of the lock condition. However, if the gain of the amplifier is not properly chosen the lock condition may delay a lot or lead to the instability of the loop. It is of utmost importance to calculate
and apply the right amount of gain for a Loop with good performance.

Initially, we want to set the NCO to 7.812 KHz. The 32-bit accumulator generated frequency $f_o$ is given by

$$f_o = \frac{f_{System} \cdot InitialValue + Error}{2^N}$$

(7.2)

where $f_{System}$ in our case is 16 MHz and the N is the width of the accumulator which is 32 bits. For an 7.812 KHz sinusoidal signal the NCO should generate a 500 KHz clock which is the $f_o$. The sum $InitialValue + Error$ in this case can be derived and is 134,217,728. In the case where the received signal is also 7.812 KHz we saw in the previous section that error is 0. That means only the $InitialValue$ contributes to the frequency generation. The accumulator overflows in 32 iterations, and since it is clocked with 16 MHz it needs 2$\mu$s which make a frequency of 500 KHz.

Figure 7.12: Phase accumulator

Setting the NCO to a different frequency requires only to change the InitialValue in the adder’s input as shown in figure 7.12. For a 7.031 KHz frequency where a clock of 450 KHz is required the $InitialValue$ is equal to 120,795,955 and for a 8.531 KHz frequency where a clock of 550 KHz is required the $InitialValue$ is equal to 147,639,501. However, in both of these two cases when the received signal is 7.812 KHz the NCO will try to match this frequency and in order to achieve that it has to change the sum $InitialValue + Error$. Since the $InitialValue$ is stable and it is used to set only the initial frequency of the NCO, the error signal is fully responsible for that.

In the condition where the NCO is set at 7.031 KHz and needs to match the 7.812 KHz incoming signal, the sum $InitialValue + Error$ needs to increase from 120,795,955 to 134,217,728. A positive error is required in that case with an approximate value of 13,421,773. In the error multiplier section we saw that in that case the multiplier’s product is 3,359,660 which is not sufficient to create this frequency transition. A gain which will amplify this error by 4 times would be appropriate since it will bring it to a value of 13,438,640, very close to the one required. The amplified error is added with the initial value and forwarded to the accumulator’s input. This new value will tend to keep the overflow in about 32 iterations to achieve the 500 KHz clock.

Similarly for the other case when the NCO is set to 8.531 KHz the sum $InitialValue + Error$ needs to decrease from 147,639,501 to 134,217,728. A negative error is required now with a value of 13,421,773. The error multiplier’s output which is -3,574,060, is again amplified by 4 to create a larger error in the phase accumulator’s input with value -14,296,240. The amplified error is added with the initial value and due to its sign leads to a smaller value in the accumulator’s input.

Considering the implementation of the Phase Accumulator, the only challenge was the gain block. Multiplication with such high numbers would have a negative impact on consumption of hardware resources. For that reason proper care has been taken to require a gain of $2^n$ where n an integer. These gains can be easily and efficiently achieved by shifting a number n positions left. This is the approach we followed in our design. The attenuation in the FIR filters have been chosen properly that will result in such an error in the multiplier’s output that will require a gain of 4. Due to the fact that this gain is constant, the synthesis tool doesn’t even need to infer a shift register. That leads to a very efficient implementation.

Finally, a factor that plays an important role to the performance of the loop is the sensitivity or commonly called frequency step. The smaller the frequency step the better the frequency accuracy that can be achieved from the NCO. The phase step is given by
and for our application is 3.72 mHz for the NCO clock. In the generated sinusoidal signals this is translated to 0.238 Hz. This is a very good accuracy which allows the NCO to match very precisely any incoming signal in the ±10% Doppler range. For lower frequency steps a 16-bit accumulator can be used which will also save hardware resources. However, in our implementation even though the accumulator has very small step the fact that the carrier frequency is very close to the bitrate (7.812 KHz and 1.953 kbps) and the filters compute a new output every 32 clock cycles where the clock is 500 KHz has as a result only 8 attempts for frequency correction per bit. To achieve as good performance as possible in this case the first attempt should be as close as possible and in our loop we achieve that with the proper gain. It is good practice in general the carrier to be 10 times or more compared to the bitrate since this will give more time to the NCO to stabilize and of course reduce hardware resources since now lower order filters would be required.

### 7.1.3 Hardware Results

In order to test the Costas Loop a step by step procedure has been followed. Initially, it was important to make sure that the AD controller works properly. For that reason, the digitally converted analog signal has been converted back to analog with another DAC on the nanoRTU. In that way we tested on the board that the AD conversion works as it should. The results of this are shown in figure 7.13. The purple waveform is the bitstream in the modulator’s input. The yellow waveform is the modulated signal converted from the modulator’s DAC. Finally, the bright blue waveform is the data from the AD converter which for testing purposes they have been forwarded to another DAC.

![Figure 7.13: AD Controller hardware results](image)

In the waveforms it is important to notice two things. First, that they are identical which means that the AD controller performs correctly. The second thing is that both waveforms have the same amplitude. Recall that the internal reference voltage of the AD has been configured at 4V from the controller and since it is powered with 3.3V the internal voltage reference is also 3.3V the same with DAC’s.

Next step in the testing procedure was to compare the output of the Costas Loop with the original bitstream. The results are shown in 7.14. The purple waveform is the original bitstream in the modulator’s
Chapter 7. BPSK Demodulator

7.2. Symbol Timing Recovery

Input. The green waveform is the demodulated data from the Costas Loop. It can be verified that they are identical.

![Figure 7.14: Costas Loop hardware results](image)

7.2 Symbol Timing Recovery

Costas Loop extracts the digital data back from the Low-IF modulated signal but in order to process these data a clock reference is needed. For that reason, proper circuitry that will sample the demodulated data at the middle point between bit transitions needs to be implemented. Circuits that perform this operation are called Symbol synchronizers or Symbol timing recovery circuits. There are various types of timing recovery circuits which differ in performance, complexity and acquisition time. In the Thesis work the Early-Late Gate timing recovery circuit has been implemented because it provides good performance in low bitrates like the ones we are dealing with and it requires a relatively small number of hardware resources [27]. Other choices are Gardner or Mueller algorithms which provide more accuracy and better acquisition times with the cost of higher complexity and require more hardware resources for their implementation.

7.2.1 Early-Late Gate Block Diagram

In order to recover the clock there are two things the receiver needs to know, the sampling frequency and when to take the sample between the bit transitions. The preferred sampling point is at the middle of the bit because in this way the probability of error decreases leading to accurate data retrieval. In our receiver we know that a bitstream of 1953 bps is expected which of course can vary slightly due to the Doppler shift effect. For that purpose, we generate a 1.953 KHz clock that will sample the recovered bitstream. The challenge that needs to be solved, and this is what the Early-Late Gate algorithm does, is to place a complete period of the clock between the bit transition intervals and sample the bit in the falling edge of the clock. In that way correct sampling is ensured. In wireless communications this is a very common problem since there is no form of synchronization between the transmitter and the receiver.

The Block Diagram of the implemented Early-Late Gate algorithm is shown in figure 7.15. The loop has three accumulators that sample the incoming data. The first one is the $PD_{EarlyAcc_0}$ which accumulates data for the time period $(0, \frac{T}{2})$ and the second one is the $PD_{LateAcc_0}$ which accumulates data for the period $(\frac{T}{2}, T)$. Both of them are clocked with 1 MHz generated from the $Counter_0$ component.
which performs frequency division by 16 of the 16 MHz system clock. The Early and Late accumulators are implemented efficiently with two counters which count when their data input and their start signals are '1'. The start signal is never '1' for both of them at the same time and that makes sure they sample in different time periods. If the input data is '1' for all the sampling period, then the maximum counting value is expected to be 255. In the other case which is '0' the counting value is 0 too. We can precisely say that the data input acts as an enable signal to the counter. The start signals show the location of the clock. During the positive period of the clock the start signal of the Early counter is '1' and counts according to the status of the input data. In a similar way, during the negative period of the clock, the start signal of the Late counter is '1'. The third accumulator in the design is the PD_BitDetector0 which works with the same principle as the other two but differs in the fact that it counts for a full clock period. That means its final counting value is expected to be the sum of the other two accumulators.

The Sign0 component in the design latches the counting values of all three accumulators and uses the value of the PD_BitDetector0 to do the bit estimation by simple averaging. If its counting value is below 255 then bit estimated to be '0' and in the opposite case '1'. This bit estimation appears on the signal DataOut which is also the data output of the timing recovery circuit. The bit estimation value together with the Early and Late latched values is used to produce the phase error which shows if the clock is in phase with the data. The error according to the position of the clock can be positive, negative or zero as derived from the formula

\[
\text{Phase error} = \begin{cases} 
(Early - Late) & \text{for DataOut 1} \\
(Late - Early) & \text{for DataOut 0} 
\end{cases}
\] (7.4)

When the Early and Late counter values are equal the error is 0 and the clock is between the bit intervals as shown in figure 7.16 a). When the DataOut is '1' and the Early-Late difference is positive then it means that the clock is late and next time it should start earlier. This case is shown in figure 7.16 b). Finally, the last case is when the error is negative and the DataOut is '1'. The clock arrives earlier compared to the data and it needs to be delayed. This case is shown in figure 7.16 c). The equation (7.4) is implemented in the PD_error0 component which is a multiplexer that has as a selection line the DataOut signal. Before we discuss how the error generated in the PD_error0 component brings the clock in phase with the data, it is necessary to focus on two issues that it is good to be aware of.
The first issue is related to the acquisition time. Observe that in figure 7.16 the bit is between two other bits with value '0'. In that case it is easy to derive the error. However, in the condition where we have long sequences of bits which are in the same logic level, that makes the error detection more difficult and the clock comes in phase with the data in longer number of cycles. In the Early-Late Gate algorithm the acquisition time depends on the data pattern and the more bit transitions we have the faster it is.

Another issue which arises is the data bit duration. The Costas Loop recovers the data and handles effectively the doppler shift issues but since the information data are contained in the received signal the recovered bitstream can have slightly different bitrate compared to the expected one. In that case to achieve the optimum performance possible, a Numerically Controlled Oscillator is required that will change the clock frequency and as a result the sampling frequency. In our implementation we don’t use an NCO in the timing recovery and we only handle the task to bring the clock in phase with the data. As a result the sampling accuracy is reduced in the very end edges of the doppler range.

After the error generation, it is forwarded to the _DCO.Counter_0 component which is a counter that every time is reset, it is loaded with the new error value. The counter's output is an input to the _DCO.Comparato_0r which is the component which generates the start signals to control the accumulators and the _exRst_ signal which resets them together with the _DCO.Counter_0. The Block Diagram in figure 7.15 shows these connections. The _DCO.Comparato_0r block is clocked with 1 MHz. In every cycle checks the counting value of the _DCO.Counter_0 block. When it is below 256 and above 0 , the Early accumulator is enabled and the positive period of the clock is generated. A value higher or equal to 256 and below 511 disables the Early accumulator and enables the Late one. Also the _exClk_ signal goes to '0' to indicate the negative period of the clock. At the condition of a counter value of 511, the _exRst_ is set to '1' to restart the accumulators and the counter.

The above are illustrated by the simulation results in figure 7.17. The _Recovered data_ is the output of the Costas Loop, the _Start Early_ and _Start Late_ are the start signals for the accumulators, _DataOut_ and _exClk_ are the data and the recovered symbol clock, respectively. In the second clock period when the recovered data from the Costas Loop start to stabilize, the error produced is negative with a value of -126. When the _exRst_ is asserted, the counter is loaded with that value. As a result of this the start of the Early accumulator will be delayed as shown. The next error is positive with a value 63 which means that the clock arrives late compared to the bit. As a result the counter is loaded with that value and the next Early period is shorter, leading to a smaller clock period in order to bring them in phase in the next one. Finally, we can see that from the next period the error is 0 and the clock is in phase with the data.

---

**Figure 7.16: Phase errors in symbol timing recovery**

**Figure 7.17: Early-Late Gate simulation results**
Chapter 7. BPSK Demodulator

7.2.2 Hardware Results

The hardware results shown in figure 7.18 except from the functionality of the timing recovery circuit demonstrate also the correct operation of a complete BPSK modem. The bright blue waveform is the received modulated signal. The purple waveform is the original bitstream in the modulator’s input. The yellow signal is the data produced from the PD_BitDetector based on the Costas Loop demodulated data. Finally, the green waveform is the recovered clock signal which as can be seen is in phase with the data. The falling edge is in the middle point between the bit transitions allowing for secure sampling.

Figure 7.18: Early-Late Gate hardware results

7.3 Synthesis Results

The BPSK modem implementation can be split in two parts. The BPSK modulator and the BPSK demodulator which is made up from the Costas Loop and the Early-Late gate symbol synchronizer. When all of them are synthesized in a single project the total consumed resources are 5761 core tiles out of 13824 which leads to an utilization of 42%. The BPSK modulator implementation differs with the one described in Chapter 3 only in the Look-up tables. A bitrate of 1953 bps has been achieved using a Look-up table with 16 samples per quadrant which makes 64 samples per period. A PLL was not necessary since the 7.812 KHz carrier has been achieved with a counter which produces a 500 KHz clock for the carrier generation state machine. The resources used to implement the modulator are 532 core tiles out of 13824 which gives an utilization of 4%. Since smaller Look-Up tables are used for this implementation the required resources are less compared to the implementation in Chapter 3 where they were 620.

Considering the demodulator, the total hardware requirements are 5075 core tiles and a total utilization of 37%. For a better understanding how its parts contribute to the total hardware requirements, we present in table 7.2 the results for each of them separately. The Costas Loop is the part of the demodulator which requires the most hardware resources and especially the FIR filters. Design choices such as selection of 8-bit coefficients or smaller Look-Up tables for the NCO can result to a more optimized design in terms of hardware resources with some performance compromises.

The BPSK modem has been also made radiation tolerant following the triple redundancy method. The synthesis results of the radiation tolerant implementation have shown an increase in the hardware requirements. A total utilization of 60% is achieved which corresponds to 8264 core tiles out of 13824.
In this design since the BPSK modem resources utilization were below 50%, it was safe enough to use the triple module redundancy method. In cases where the actual design consumes more than 50% of the resources it is good practice to apply the triple module redundancy method only in critical parts of the design and use for the rest the combinatorial cells method which has less hardware requirements.

Table 7.2: Demodulator synthesis results

<table>
<thead>
<tr>
<th>System component</th>
<th>Core cells</th>
<th>Total chip utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Costas Loop</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2x Phase Detector</td>
<td>636</td>
<td>4%</td>
</tr>
<tr>
<td>Error multiplier</td>
<td>863</td>
<td>6%</td>
</tr>
<tr>
<td>2x FIR</td>
<td>2256</td>
<td>16%</td>
</tr>
<tr>
<td>NCO</td>
<td>398</td>
<td>3%</td>
</tr>
<tr>
<td><strong>Symbol synchronizer</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Counter</td>
<td>9</td>
<td>0%</td>
</tr>
<tr>
<td>DCO Comparator</td>
<td>221</td>
<td>2%</td>
</tr>
<tr>
<td>DCO Counter</td>
<td>308</td>
<td>2%</td>
</tr>
<tr>
<td>PD BitDetector</td>
<td>90</td>
<td>1%</td>
</tr>
<tr>
<td>PD EarlyAcc</td>
<td>91</td>
<td>1%</td>
</tr>
<tr>
<td>PD LateAcc</td>
<td>91</td>
<td>1%</td>
</tr>
<tr>
<td>PD Error</td>
<td>189</td>
<td>1%</td>
</tr>
<tr>
<td>Sign</td>
<td>107</td>
<td>1%</td>
</tr>
</tbody>
</table>
Chapter 8

Conclusions

In this Master thesis work, study about the LEO satellites has been done with focus on the factors that affect their communication systems. Due to their close distance to Earth, they reach very high orbital speeds which make the Doppler shift effect very intense. For that reason, proper measures need to be taken in the spacecraft’s receiver to avoid loss of communication. That required an understanding of the Doppler effect phenomenon and a hardware oriented research to find ways to implement such a receiver that will have the ability to tolerate this effect.

The study about the effect of radiation in semiconductors was another very important outcome of this thesis. Deep space offers a very challenging environment for semiconductor devices since it can lead them to temporal or even permanent malfunction. Considering that this thesis focuses on a very important part of the spacecraft’s electronics such as communication systems, it was necessary to understand the causes of this radiation. The study has been split in three main areas:

- The causes of the problem. Two causes have been studied. The first one is the radiation due to high energy particles coming from cosmic rays and the second one due to trace amounts of Uranium and Thorium contained in packaging materials of integrated circuits. Both forms of radiation can affect registers or memory blocks and can change their content.

- The radiation tolerant capabilities of the two dominant FPGA technologies, SRAM and Flash, have been studied. FPGAs are the main platform for digital hardware development and the proper understanding of their behaviour in these environments is important for the suitable technology selection. Flash based FPGAs have better radiation tolerance capabilities due to the structure of their basic element which is a Flash cell. That in combination with their very low power consumption makes them suitable for aerospace applications.

- Design techniques that help achieve radiation immunity have been also studied. With the synthesis tool used three methods are available. All of these methods have in common that they aim to replace registers in the design with different structures. The first method replaces a register with combinational elements with feedback. Since, combinational elements are affected much less from radiation this method leads to an improved design with low hardware requirements. The second method that has been studied is the triple module redundancy method which replaces a register with three registers and a majority voter. The voter propagates the content that two of the three registers have. This method offers very good radiation immunity but has the disadvantage that theoretically can result in even three times more the hardware requirements. The last method is a combination of the other two and replaces a register with three other registers, made up from combinational elements with feedback, and a majority voter. The hardware requirements of this method make it applicable to only a small number of designs.

The implementation part of the thesis work aims to cover the satellite to Earth communication which is achieved through Telemetry and Telecommand data and since both of them are digital, modulation and demodulation schemes needed to be implemented for their proper transmission and reception, respectively. The result of the implementation part of the thesis is four phase modulators, two frequency modulators
and one complete modem solution. Below the outcome for each of these implementations is mentioned briefly:

- A BPSK modem has been implemented that is made up from a modulator and a demodulator. The demodulator, using a Costas Loop, handles frequency shifts in its input up to $\pm 10\%$ and achieves carrier and data recovery. An Early-Late gate symbol synchronizer is used to create and bring in phase the clock with the recovered data for further baseband processing.

- An SDBPSK modulator that allows the Telemetry data to be differentially encoded before modulation. In that way non-coherent demodulation can be achieved since now the information is between two consecutive bits.

- For improved bandwidth efficiency, the QPSK and OQPSK modulations have been implemented. These map 2 bits per symbol and require half the bandwidth compared to BPSK. OQPSK enjoys the advantage of being more constant-envelope compared to QPSK which makes it more friendly to low power amplifiers.

- The AFSK and MSK frequency modulations have been implemented. MSK modulation achieves optimum bandwidth efficiency due to minimum frequency separation and at the same time phase continuity.

All the implementations have been tested on the nanoRTU 211 using VHDL and the Libero v9.1 suite for Actel FPGAs and the Synplify synthesis tool from Synopsys.

8.1 Future Work

This project covers just the part of the communication systems related to digital modulations. That means as a future work a complete satellite transceiver is necessary to be seen as a system in overall. This will allow to specify the RF frequencies, the Front-End specifications and as a result the modulator’s and demodulator’s requirements. The future work considering the current design is the following:

- In our implementation we followed the Look-Up table method for carrier generation. It has been an easy and sufficient way of sinusoidal waveform synthesis. However, Look-Up tables require a lot of resources which in our case was not a problem due to low bitrates. Moving to higher bitrates the problem becomes visible and other methods for waveform synthesis are required. One method that is suitable for generation of sinusoidal orthogonal functions is using the COordinate Rotation DIgital Computer (CORDIC) algorithm. It is a hardware efficient algorithm that can be implemented using only basic digital structures like shift registers and adders and does not require any form of storage.

- For the FIR filters in the Costas Loop, a 16-bit coefficient representation has been adopted using the Q15 fixed point format. Even though, the data path is optimized, an 8-bit coefficient representation will lead to a smaller design. That will have as a result a different filter response which of course needs to be analysed how it affects the loop.

- Finally, an improvement to the implemented Early-Late gate algorithm is necessary, not only to bring the generated clock in-phase with the recovered data but also adjust its frequency in order to achieve correct sampling even in the edges of the Doppler range.
Bibliography


