X-band Transmitter

Baseband DSP Module

Pilot Project

Prepared By: Mo'men Ahmed Ibrahem

Supervisors: Eng. Hany Bekhit Ottafy

Eng. Aya Mohammad Ali

Date of submission:

Number of Pages: 24

Table of Contents

Table of Figures

1 PROJECT INTRODUCTION

The major challenge of any communication system used is the degrading effect of noise on the sent signals and data, while in analog communication sent data is in a propagating waveform making it affected by electromagnetic distortion and requires high quality processing which in turn demands costly hardware.

Many most of the reliable communication systems are based on Digital communication and the main challenge is to decrease the effect of noise which is very effective on large distance communication especially satellite communication which uses x-band communication for sending images and important data related to the payload of the satellite.

Noise in digital communication is mainly affecting the bits of zeros and ones sent which is always referred to as BER (bit error rate), to gain lower bit error rate signal to noise ratio must be greater which could be energy consuming operation so for a better optimization, Digital signal processing modules could be added to improve the performance of the communication system as it detects errors and correct.

Mainly the block modules for the proposed DSP system, which are made on hardware description language and tested on Artix-7, are differential encoder, scrambler and conventional encoder and to test the system pseudo random sample generator is used as an input.

1.1 X-BAND TRANSMITTER – FUNCTIONS AND IMPORTANCE

1.1.1 Electromagnetic spectrum

The X band is a segment of the microwave radio region of the electromagnetic spectrum. In some cases, such as in communication engineering, the frequency range of the X band is rather indefinitely set at approximately 7.0 to 11.2 gigahertz (GHz).

1.1.2 Applications

X band frequency sub-bands are used in civil, military, and government institutions for weather monitoring, air traffic control, maritime vessel traffic control, defense tracking, and vehicle speed detection for law enforcement.

Satellites sends images through X-band communication in a frequency range from 8 to 8.4 gigahertz (GHZ).

1.2 DSP BASEBND BLOCK – FUNCTIONS AND IMPORTANCE

1- To receive the clock and data from clock and data interfaces.
2- To apply the DSP algorithm on the transmitted data. The DSP algorithm includes differential encoding, scrambling and convolutional encoding.
3- To generate test data pattern to be used in the self-test mode of operation.

2 DSP BLOCK DESIGN

As shown in figure 1, the main block diagram takes clock (clk) from the FPGA kit, reset (rst) and the block enable which is named (clk_en) and its output is in- phase and Quadrature.

Quadrature is delayed by a half clock cycle which needs a frequency synthesizer to divide the two clocks used for the modules.

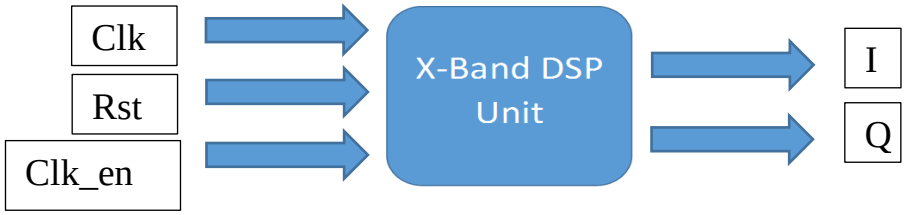


Figure 1: Block diagram

As the shown in figure 2, the detailed block diagram of the complete system which consists of PRS (pseudo random signal generator), differential encoder, scrambler, convolutional encoder and clock divider (frequency synthesizer).

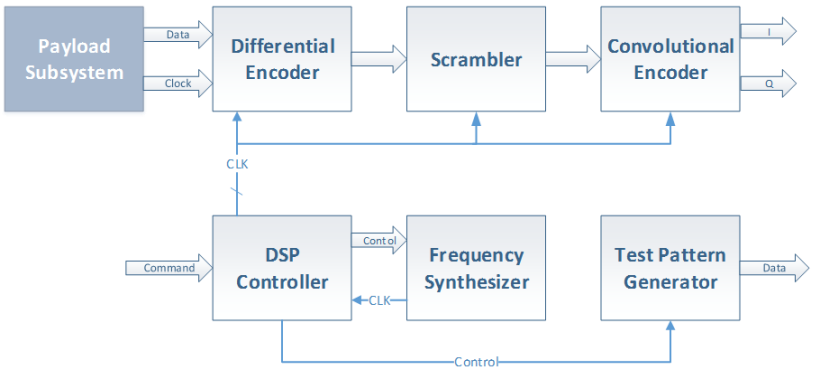


Figure 2: Detailed block diagram

2.1 PRS:

The pseudo random generator is a test generator which has random input values which repeats every period, that period is specified through a polynomial which is for 7 bits should be equal to a length of 2^7 - 1 bits meaning it should repeat every 127 bits.

This test pattern replaces the image of payload subsystem to be transmitted, so the main use of PRS is to test the function of the DSP system and its performance as well.

Its design consists of a shift register of 7-bits (D^7+D^3+1) whose last and third bit XOR together and entered again as a closed loop feedback to the last bit.

2.2 DIFFERENTIAL ENCODER

It is one of the simplest forms of error protection coding done on a baseband sequence before modulation, it consists of a modules 2 as shown in Figure 3, a feedback of XOR gate with a time delay block.

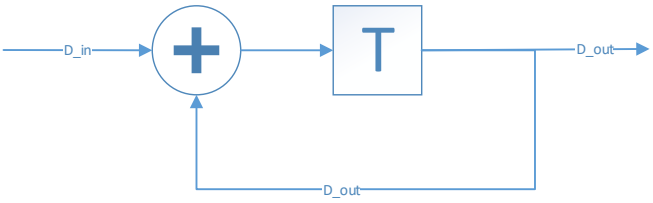


Figure 3: Differential Encoder

2.3 SCRAMBLER

It is an algorithm that converts an input bit string into a seemingly random output string of the same length, thus avoiding long sequence of bits of the same value.

Scrambling device represents shift register with feedbacks as shown in Figure 4. It performs generation of the pseudo-random code as per the given generating polynomial. According to ITU-T (formerly known as CCITT) V.35 standard, $1+x^3+x^{20}$ generating polynomial have been applied.

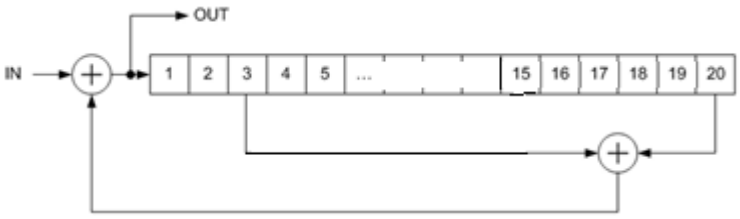


Figure 4: Scrambler

2.4 CONVOLUTIONAL ENCODER

Convolutional codes differ from block codes in that the encoder contains memory and the n encoder outputs at any time unit depend not only on the k inputs but also on m previous input blocks.

It generates an encoded bit stream based on a specified code rate. The code rate is equal to the ratio of the data word length (k) to the code word length (n).

A convolutional code is generated by passing the information sequence to be transmitted through a linear finite-state shift register as shown in Figure 5.

In general, the shift register consists of *k* (k-bit) stages and *n* linear algebraic function generators.

We use convolutional encoder with the following characteristics:

Code rate, R = 1/2

Constraint length, K = 7 bits.

Connection vectors: G_1 = 1111001 (171 octal); G_2= 1011011(133 octal).

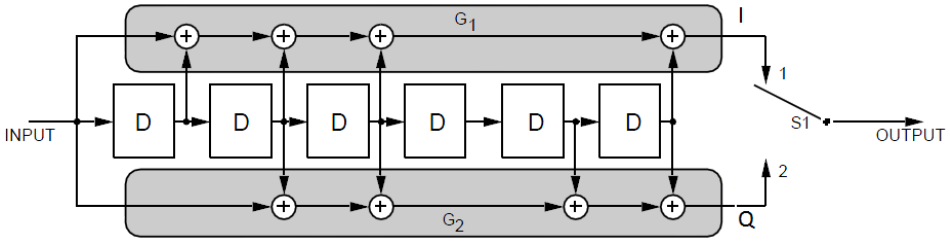


Figure 5: Convolutional Encoder

2.5 CLOCK DIVIDER

The task of this block is to provide the required clock frequencies according to the input commands of data rate selection. A set of clock frequencies based on the 100 MHz crystal oscillator frequency will be generated. The required frequencies are100 MHz, and 50 MHz.

3 DSP BLOCK CODING, SIMULATION AND IMPLEMENTATION

3.1 MATLAB SIMULATION

The block diagram in Figure 6 is used to make a simulating reference to work on and compare it with the VHDL design simulation output.

Some values of the blocks used are:

PN sequence generator which is PRS has polynomial of value [1 0 0 1 0 0 0 1] and initial states of value [1 0 0 0 0 0 0]

Differential encoder has an initial state of 0

Scrambler has a calculation base of 2 and its polynomial is 'z^20 + z^3 + 1' with zeros initial state

The Eb/No of the AWGN is changeable depending on the channel noise while the value I used is 2.

The Viterbi decoder has a trace back depth of 35 and it is the inverse of the convolutional encoder.

The error rate calculation block also should have 35 trace back and its computational delay is 2.

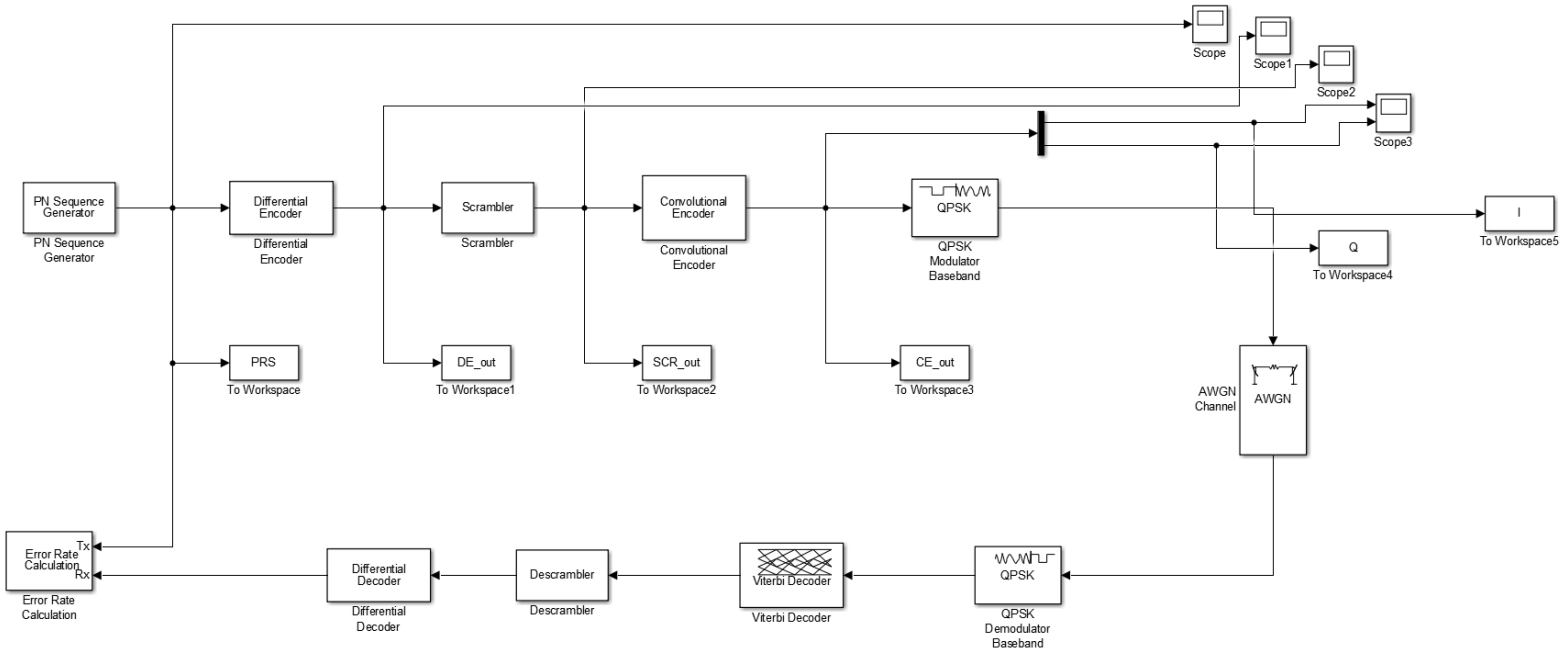
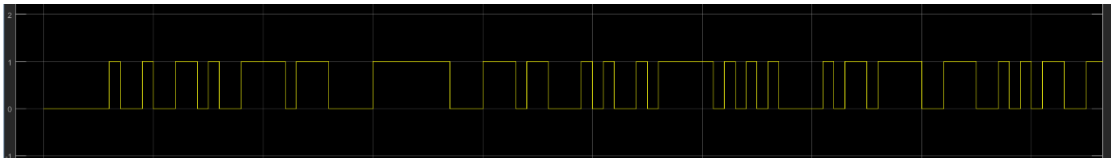


Figure 6: Matlab block diagram

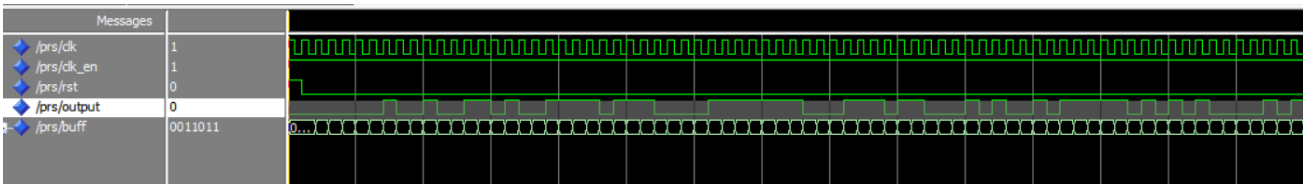
3.2 RESULTS

3.2.1 PRS (test pattern generator)

Matlab simulation result:



Modelsim Simulation:



FPGA implementation on ChipScope Pro:

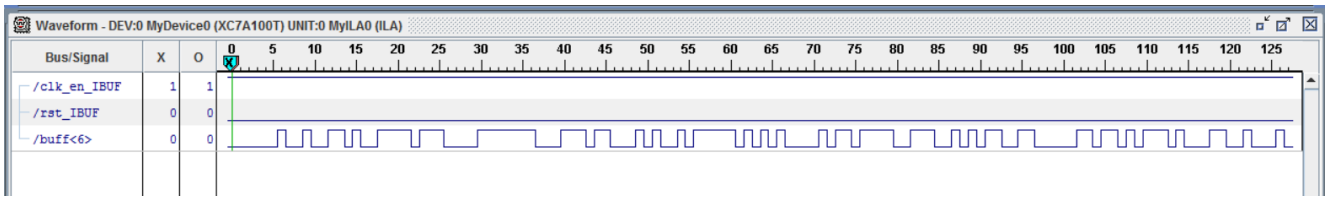
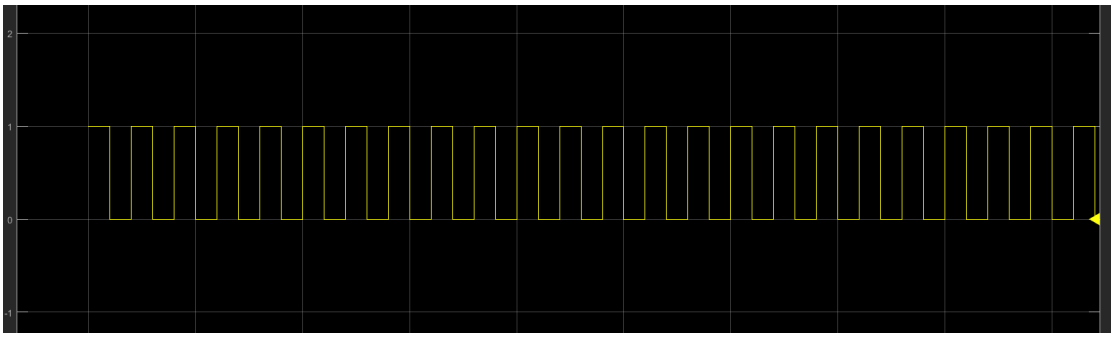


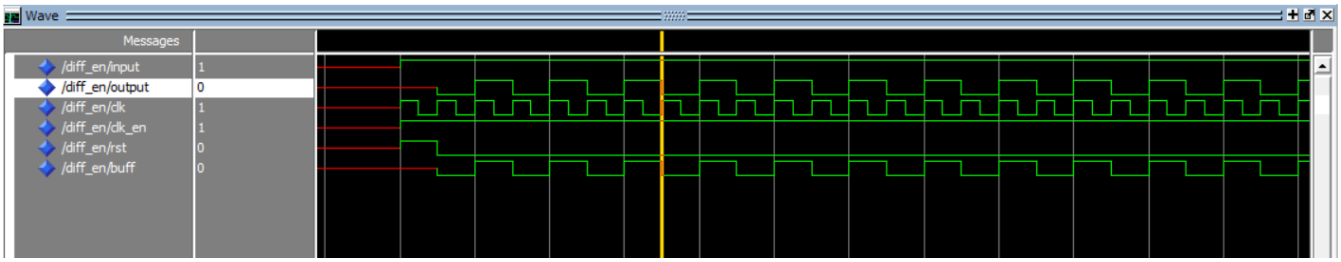
Figure 7: PRS simulation and implementation

3.2.2 Differential Encoder

Matlab simulation result for constant input of 1:



Modelsim Simulation:



FPGA implementation on ChipScope Pro:

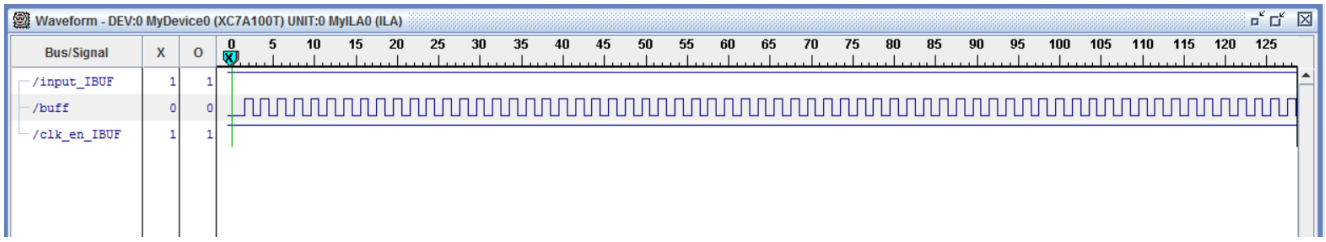
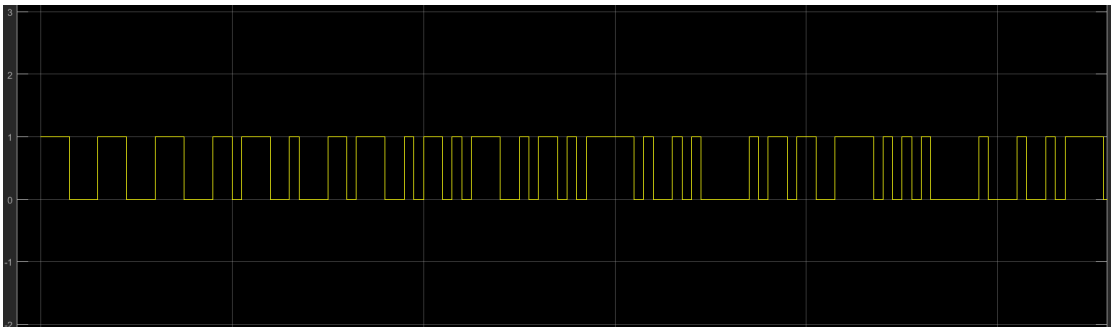


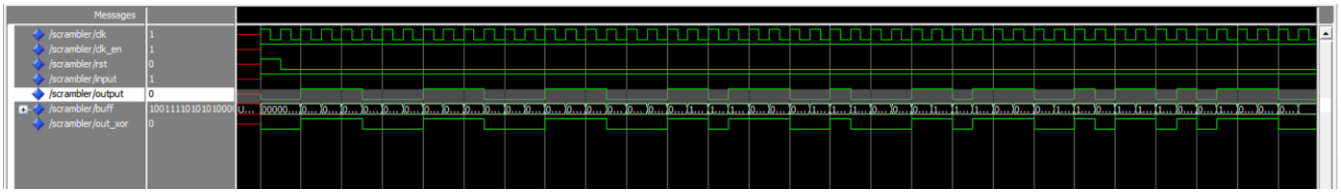
Figure 8: Differential Encoder implementation

3.2.3 Scrambler

Matlab simulation for a constant input 1:



Modelsim simulation:



FPGA implementation on ChipScope Pro:

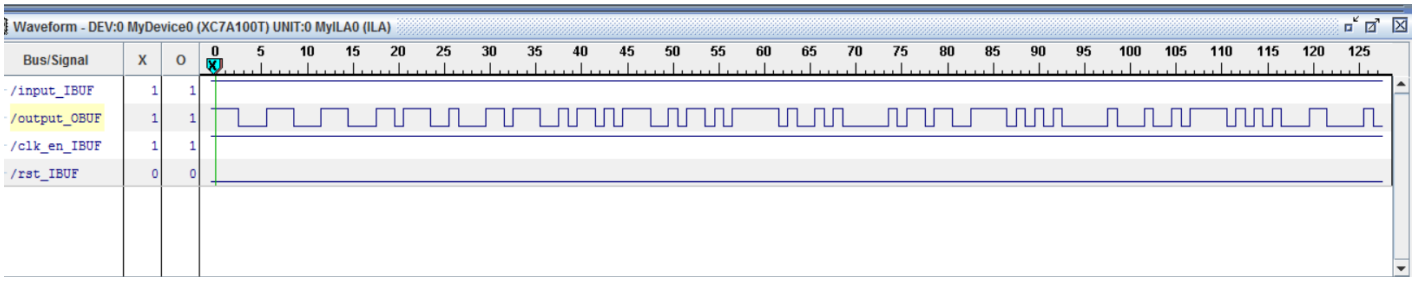
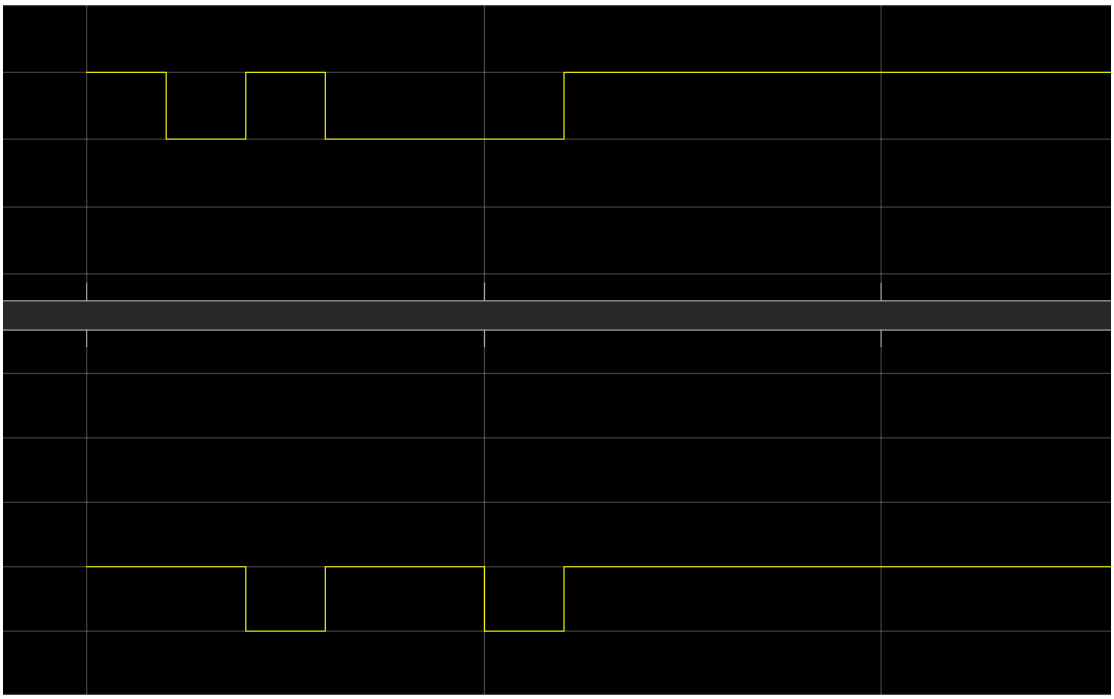


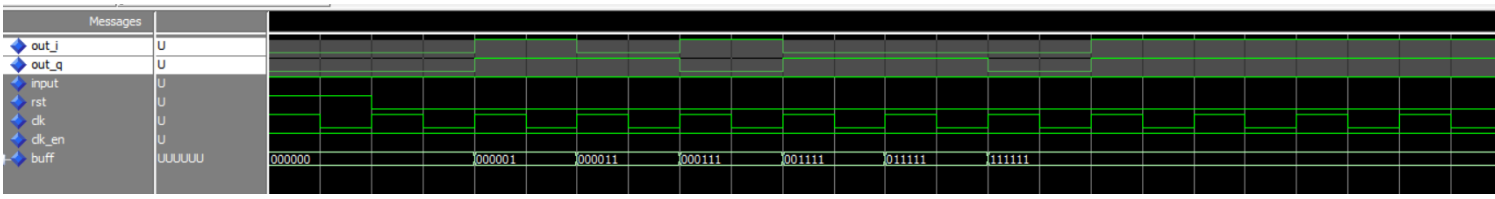
Figure 9: Scrambler implementation

3.2.4 Convolutional Encoder

Matlab simulation for constant input of 1:



Modelsim simulation:



FPGA implementation on ChipScope Pro:

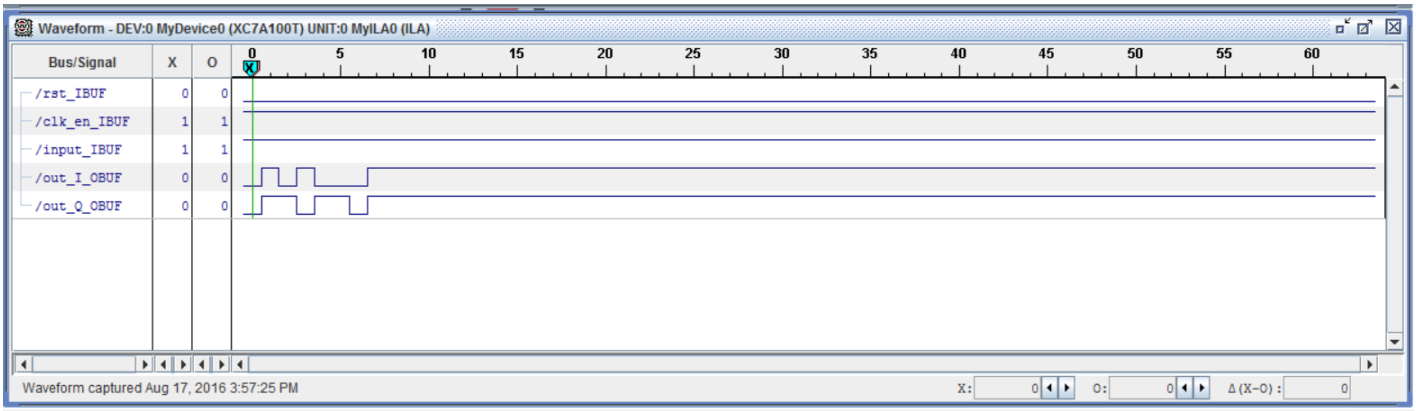


Figure 10: Convolutional simulation and implementation

3.2.5 TOP LEVEL

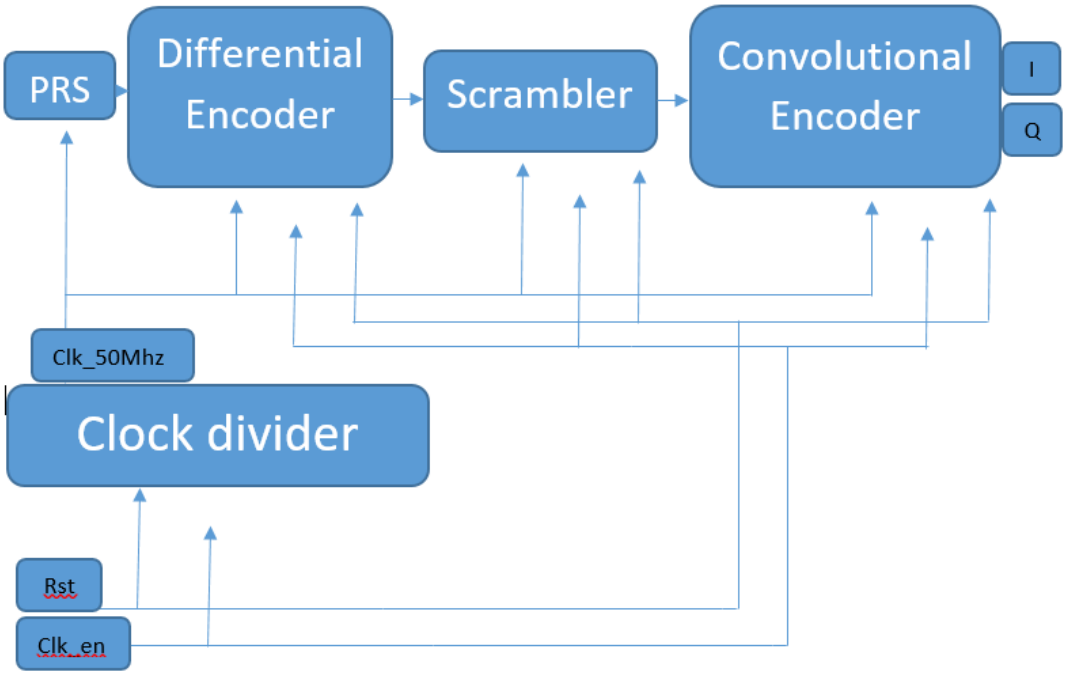
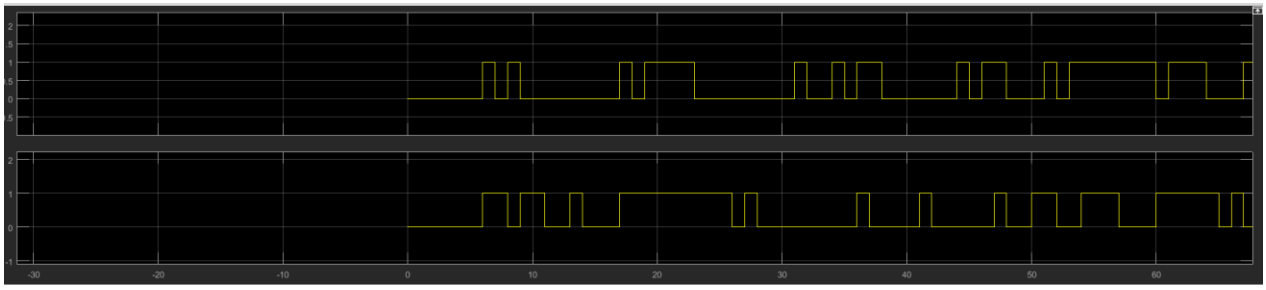


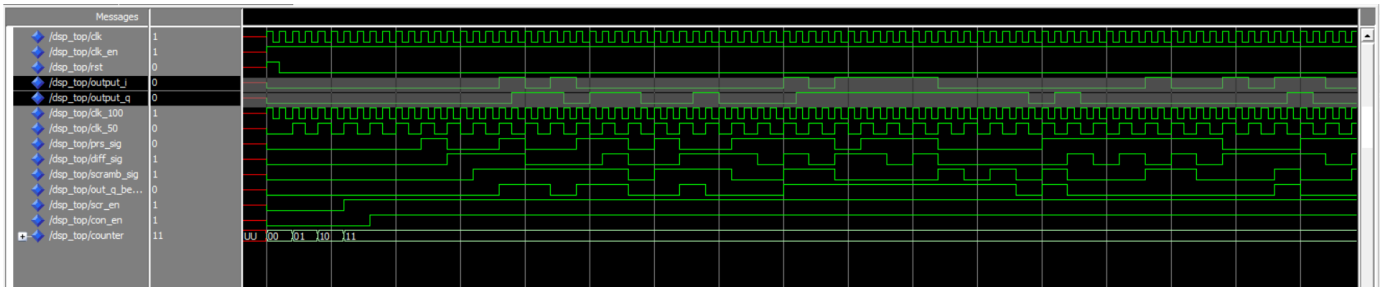
Figure 11: Top Level Block Diagram

SIMULATION

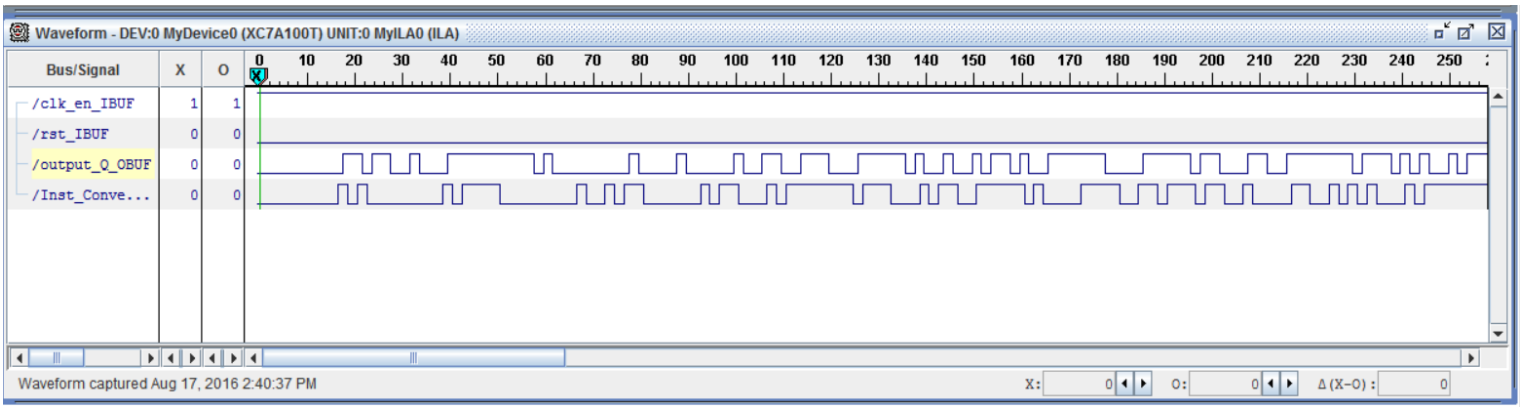
Matlab simulation showing I in the upper graph and Q in the lower one.



Modelsim simulation:



FPGA implementation on ChipScope Pro:



SYNTHESIS

Table 1 : Table of synthesis report

Module	Max. Frequency	# of Registers	Clock Buffers	I/O Buffers
Top module	634.115 MHz	43	1	4
PRS	851.798 MHz	7	1	3
Differential Enc.	944.287 MHz	1	1	4
Scrambler	851.789 MHz	21	1	4
Conv. Enc.	707.214 MHz	8	1	5
Clk. Divider	685.871 MHz	1	1	5

4 CODES

```
entity diff_en is
    Port ( input : in  STD_LOGIC;
           output : out  STD_LOGIC;
           clk : in  STD_LOGIC;
           clk_en : in  STD_LOGIC;
           rst : in  STD_LOGIC);
end diff_en;

architecture Behavioral of diff_en is

signal buff : STD_LOGIC;

begin

process(input,clk_en,rst,clk,buff)
begin

if (rst = '1') then
      output <= '0';
      buff <= '0';

elsif ( rising_edge (clk) ) then

   if ( clk_en = '1') then
      buff <= buff XOR input;

   else
      output <= '0';
      buff <= '0';

   end if;
end if;
output <= buff;
end process;



end Behavioral;
```

```
entity PRS is
    Port ( clk : in  STD_LOGIC;
           clk_en : in  STD_LOGIC;
           rst : in  STD_LOGIC;
           output : out  STD_LOGIC);
end PRS;

architecture Behavioral of PRS is
signal buff : STD_LOGIC_VECTOR (6 downto 0);
begin

process(clk,clk_en,rst,buff)
begin
if (rst = '1') then
      buff <= "0000001";

elsif ( rising_edge (clk) ) then

   if ( clk_en = '1') then
      buff (6 downto 0) <= buff (5 downto 0) & (buff(2) xor buff(6)) ;
   else
      buff <= "0000001";
   end if;
end if;

output <= buff(6);

end process;
end Behavioral;
```

```
entity Scrambler is
    Port ( clk : in  STD_LOGIC;
           clk_en : in  STD_LOGIC;
           rst : in  STD_LOGIC;
           input : in  STD_LOGIC;
           output : out  STD_LOGIC);
end Scrambler;

architecture Behavioral of Scrambler is
signal buff : STD_LOGIC_VECTOR (19 downto 0);
signal out_xor : STD_LOGIC;
begin



process(clk,clk_en,rst,buff,input,out_xor)
begin

if (rst = '1') then
      buff <= "00000000000000000000";
      out_xor <= '0';
elsif ( rising_edge (clk) ) then

  if ( clk_en = '1') then

      buff <=   buff (18 downto 0) & ((buff(2) xor buff(19)) XOR input);
      out_xor <= (buff(2) xor buff(19)) XOR input;
   else

      buff <= "00000000000000000000";
   end if;
end if;

output <= out_xor;

end process;

end Behavioral;
```

```
entity Conventional_Encoder is
    Port ( input : in  STD_LOGIC;
           out_I : out  STD_LOGIC;
           out_Q : out  STD_LOGIC;
           rst : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           clk_en : in  STD_LOGIC);
end Conventional_Encoder;

architecture Behavioral of Conventional_Encoder is
signal buff : STD_LOGIC_VECTOR (5 downto 0);

begin

process(clk,clk_en,rst,buff,input)
begin
if (rst = '1') then
      buff <= "000000";
      out_I <= '0';
      out_Q <= '0';

elsif ( rising_edge (clk) ) then

   if ( clk_en = '1') then
      buff  <=  buff (4 downto 0) & input;
      out_I <= (((input XOR buff(0))XOR buff(1))XOR buff(2)) XOR buff(5);
      out_Q <= (((input XOR buff(1))XOR buff(2))XOR buff(4)) XOR buff(5);
   else
      buff <= "000000";
      out_I <= '0';
      out_Q <= '0';

   end if;
end if;



end process;

end Behavioral;
```

```
entity clk_diff is
    Port ( clk : in  STD_LOGIC;
           clk_100 : out  STD_LOGIC;
           clk_50 : out  STD_LOGIC;
           rst : in  STD_LOGIC;
           clk_en : in  STD_LOGIC);
end clk_diff;

architecture Behavioral of clk_diff is

signal tmp : STD_LOGIC ;

begin
process(clk,clk_en,rst,tmp)
begin
if ( rst = '1'  ) then
      clk_50 <= '0';
      tmp <= '0';
elsif (rising_edge (clk)) then
   if ( clk_en = '1') then
      tmp <= NOT ( tmp) ;
   else
      clk_50 <= '0';
      tmp <= '0';
   end if;
end if;

clk_50 <= tmp ;
clk_100 <= clk;

end process;

end Behavioral;
```

```
process(clk_50,clk_en,rst)
   begin
   if (rst='1')then
      counter <= "00";
      scr_en <= '0';
      con_en <= '0';
   elsif (clk_en = '1')then
      if (rising_edge(clk_50)) then

         if ( counter = "10")then
            scr_en <= '1';
            counter <= counter + 1;
         elsif (counter ="11")then
            con_en <= '1';
         else
            counter <= counter + 1;
         end if;
      end if;
   end if;
   end process;

process (clk_100,rst)
   begin

if(rst='1') then
      output_Q <= '0';

elsif (rising_edge(clk_100)) then
   if (con_en = '1')then
      output_Q <= out_Q_before;

   end if;
end if;
   end process;

end Behavioral;
```

5 LESSON LEARNED

Training on VHDL (Hardware description language) and debugging the resulted waveforms with Mat lab's simulation output.

Understood Digital communication systems, how they work and a comparison between digital and analog communication performance, we determined that digital is much better and most of the communication systems nowadays are based on it.

While its performance is better, it also can be affected by noise so we had to add more modules to improve performance by error detection and correction, to test the system we had a stimulus that is not deterministic and its values are not completely known so we used PRS (pseudo random generator).

Learned Mat lab Simulink and build a complete transmit and receive communication system to simulate the system performance and compare it with the VHDL designs.

Implementing a half clock cycle delay needed a clock divider module to make Quadrature output delay with a half clock cycle than In-phase output.

Understanding Digital signal processing, implementing and in-chip debugging and in-chip verification for modules by using Xilinx Design Suite 14.7 for VHDL design, Modelsim 6.5 for simulation, Matlab Simulink for comparison and ChipScope Pro for FPGA implementation.

6 CONCLUSION AND FUTURE WORK

Digital signal processing is a very important and essential block in satellites in order to send data as images through X-band communication to the base station for its reliability but it can face many challenges that needs modification and performance improvements.

For future enhancements, as shown in Table 1, modules are not completely optimized so as an improvement to the design optimizing it is by decreasing the number of slice registers used and increasing the maximum frequency of the design, as the top module maximum frequency could reach up to 700 MHZ when optimized.

Also, as an enhancement is the implementing of the complete X-band module with its transmitter and receiver parts indicated in Figure 6: Matlab block diagram.

Implementing a QPSK or OQPSK modulation and demodulation module could be considered as a great improvement to the system in order to complete the X-Band communication system.