Project-Based Learning Report

on

"Design a 16 bit QPSK"

Submitted in the partial fulfillment of the requirements.

For the Project-based learning in **VLSI Design Technology**

in
Electronics & Communication Engineering

By

AASHISH KUMAR **PRN-2014111065**

EKAKSHAR JOSHI **PRN-2014111054**

Under the guidance of Course In-charge

Dr. AJAY KUMAR KUSHWAHA

Bharati Vidyapeeth
(Deemed to be University)
College of Engineering,
Pune – 411043

Academic Year: 2023-24

CERTIFICATE

This is to be Certified that the Project Based Learning report entitled, "**Design a 16 bit QPSK**" **work is done by**

AASHISH KUMAR **PRN-2014111065**

EKAKSHAR JOSHI **PRN-2014111054**

In partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in **VLSI Design Technology** Bachelor of Technology Semester V1, Electronics and Communication Engineering.

Date: 21-04-2023

Dr. Ajay kumar kushwaha **Dr. Arundhati A. Shinde**

Course In-charge **Professor & Head(ECE)**

INDEX

CHAPTER-1

Problem statement- The problem statement is to implement and design a 16 bit qpsk using a VHDL software.

Solution:

To design a 16-bit QPSK (Quadrature Phase Shift Keying) modulation scheme, we first need to understand the basics of QPSK.

QPSK is a type of digital modulation in which two bits are modulated at once, by varying the phase of the carrier signal. The modulated signal is divided into four quadrants, each representing a specific combination of two bits. The four combinations are (00), (01), (10), and (11), and they correspond to phase shifts of 0, π/2, π, and 3π/2 radians, respectively.

To design a 16-bit QPSK modulation scheme, we need to divide the 16-bit input data into 8 pairs of bits, and then modulate each pair using QPSK. This means that we will have 8 QPSK symbols, each consisting of 2 bits, for a total of 16 bits.

The QPSK modulation can be done using the following steps:

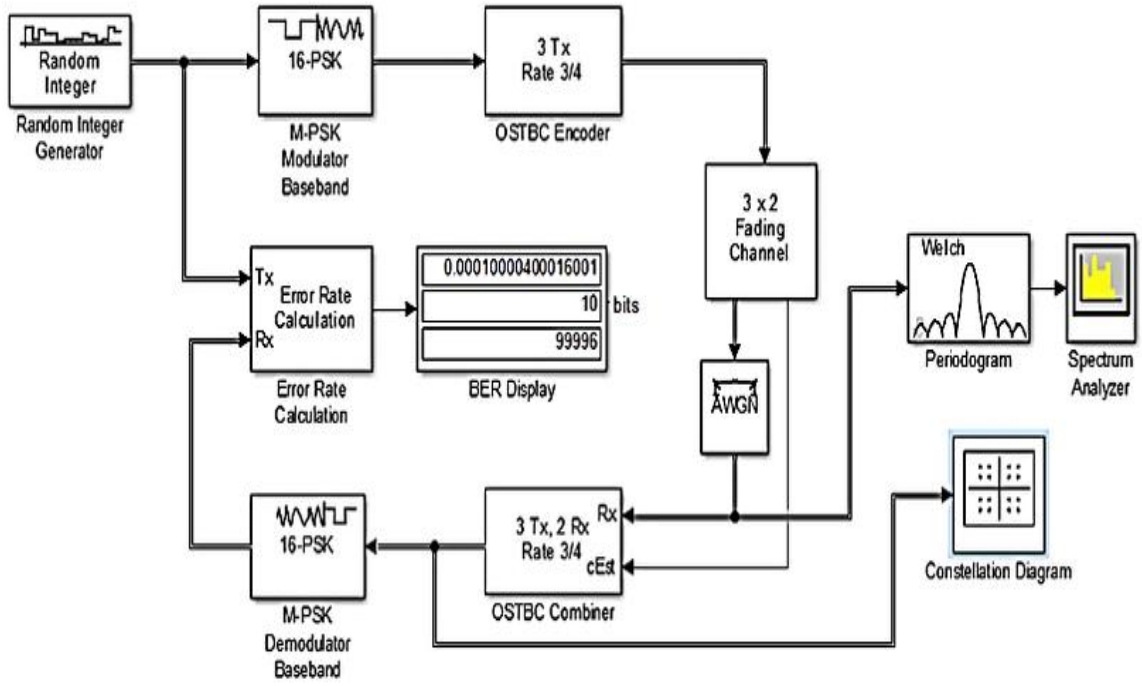
1. Divide the 16-bit input data into 8 pairs of bits.
2. Map each pair of bits to a specific phase shift using the QPSK constellation diagram. For example, the first pair of bits could be mapped to a phase shift of 0 radians for (00), π/2 radians for (01), π radians for (10), and 3π/2 radians for (11).
3. Generate a carrier signal at the desired frequency.
4. Modulate the carrier signal with each QPSK symbol by multiplying the carrier signal with the appropriate phase shift for each symbol.
5. Sum the modulated symbols to generate the final QPSK signal.

The resulting QPSK signal will have a constant amplitude and will be able to transmit 16 bits of data using only 8 QPSK symbols. This makes it an efficient modulation scheme for transmitting digital data over a communication channel.

CHAPTER-2

Design a 16 bit QPSK

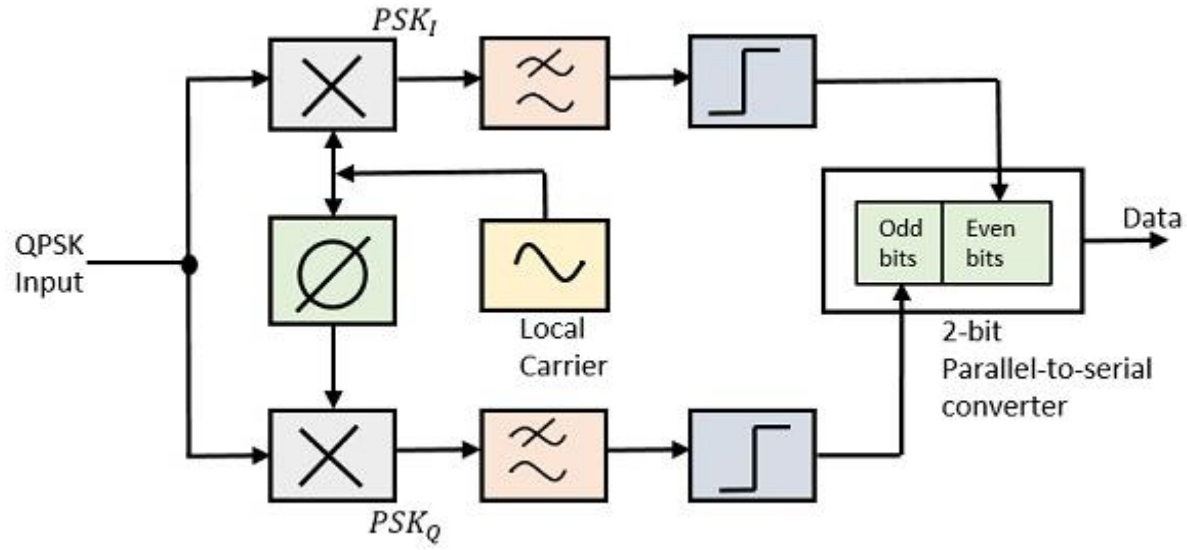
QPSK modulation and demodulation



PSK (Quadrature Phase Shift Keying) is a digital modulation scheme used in communication systems to transmit data over a carrier wave. It modulates two bits of digital data onto each carrier signal phase change. The carrier wave is split into two equal-amplitude components, with one component being shifted in phase by 90 degrees with respect to the other. The two data bits to be transmitted are then mapped to specific phase changes of the carrier wave.

QPSK is a form of PSK (Phase Shift Keying) modulation and is more spectrally efficient than simpler modulation schemes like binary phase shift keying (BPSK). This means that it can transmit more data per unit of bandwidth. QPSK is commonly used in satellite communication, wireless LANs (Local Area Networks), and digital cable television systems.

One potential drawback of QPSK is that it is more sensitive to noise and interference than simpler modulation schemes like BPSK. However, more advanced modulation schemes like 16-QAM (Quadrature Amplitude Modulation) and 64-QAM have been developed to address this issue while maintaining high spectral efficiency.



QPSK demodulation is the process of recovering the original digital data from a QPSK modulated carrier signal. This is typically done in a receiver after the modulated signal has been transmitted over a communication channel and received by an antenna.

The demodulation process involves the following steps:

Carrier recovery: The first step is to recover the carrier signal that was used to modulate the digital data. This is done using a phase-locked loop (PLL) or a similar circuit that tracks the phase of the received carrier signal.

Phase detection: The next step is to detect the phase of the received carrier signal. This is done by multiplying the received signal with two local oscillator signals that are shifted in phase by 90 degrees with respect to each other. This produces two signals that are proportional to the in-phase and quadrature components of the received signal.

Symbol recovery: Once the in-phase and quadrature components have been detected, the next step is to recover the digital data symbols that were modulated onto the carrier signal. This is done by comparing the phases of the in-phase and quadrature components to a reference constellation of QPSK symbols. Each pair of in-phase and quadrature components corresponds to a particular QPSK symbol, which can be mapped back to the original digital data.

Error correction: Finally, the recovered digital data may be subject to error correction techniques to correct any errors that may have occurred during transmission.

Overall, QPSK demodulation is a complex process that requires careful synchronization and detection of the carrier signal and the digital data symbols. However, with the right equipment and techniques, it is possible to recover the original digital data with a high degree of accuracy.

CHAPTER-3

Procedure

To implement a 16-bit QPSK in Vivado, you can follow these steps:

1. Create a new Vivado project and select the appropriate FPGA board and device.
2. Create a new VHDL module for the QPSK transmitter and receiver.
3. Define the input data format and clock frequency. For a 16-bit QPSK, the input data format will be 16-bit I/Q samples.
4. Implement the QPSK modulator and demodulator algorithms in VHDL code. For QPSK modulation, you can use a lookup table to map the input symbols to complex-valued constellations. For QPSK demodulation, you can use a complex multiplication and a slicer to recover the input symbols from the received signal.
5. Simulate the QPSK transmitter and receiver using a testbench. You can use the built-in Vivado waveform viewer to visualize the input and output waveforms and verify the correct operation of the QPSK system.

Code:

```
library ieee;

use ieee.std_logic_1164.all;

use ieee.numeric_std.all;



entity qpsk_tx is

  port (

    clk   : in  std_logic;

    rst   : in  std_logic;

    data  : in  std_logic_vector(15 downto 0);

    q_out : out std_logic;

    i_out : out std_logic

  );
```

```
end entity;


architecture rtl of qpsk_tx is
  signal counter : integer range 0 to 3 := 0;
  signal phase   : std_logic_vector(1 downto 0) := "00";
begin
  process (clk, rst)
  begin
    if rst = '1' then
      counter <= 0;
      phase   <= "00";
      q_out   <= '0';
      i_out   <= '0';
    elsif rising_edge(clk) then
      counter <= counter + 1;
      if counter = 4 then
        counter <= 0;
        case phase is
          when "00" =>
            q_out <= data(15);
            i_out <= data(14);
            phase <= "01";
          when "01" =>
```

```
            q_out <= data(13);
            i_out <= data(12);
            phase <= "10";
          when "10" =>
            q_out <= data(11);
            i_out <= data(10);
            phase <= "11";
          when "11" =>
            q_out <= data(9);
            i_out <= data(8);
            phase <= "00";
             when others =>
             q_out <= data(9);
            i_out <= data(8);
            phase <= "00";
        end case;
      end if;
    end if;
  end process;
end architecture;
```

CHAPTER-4

Vivado

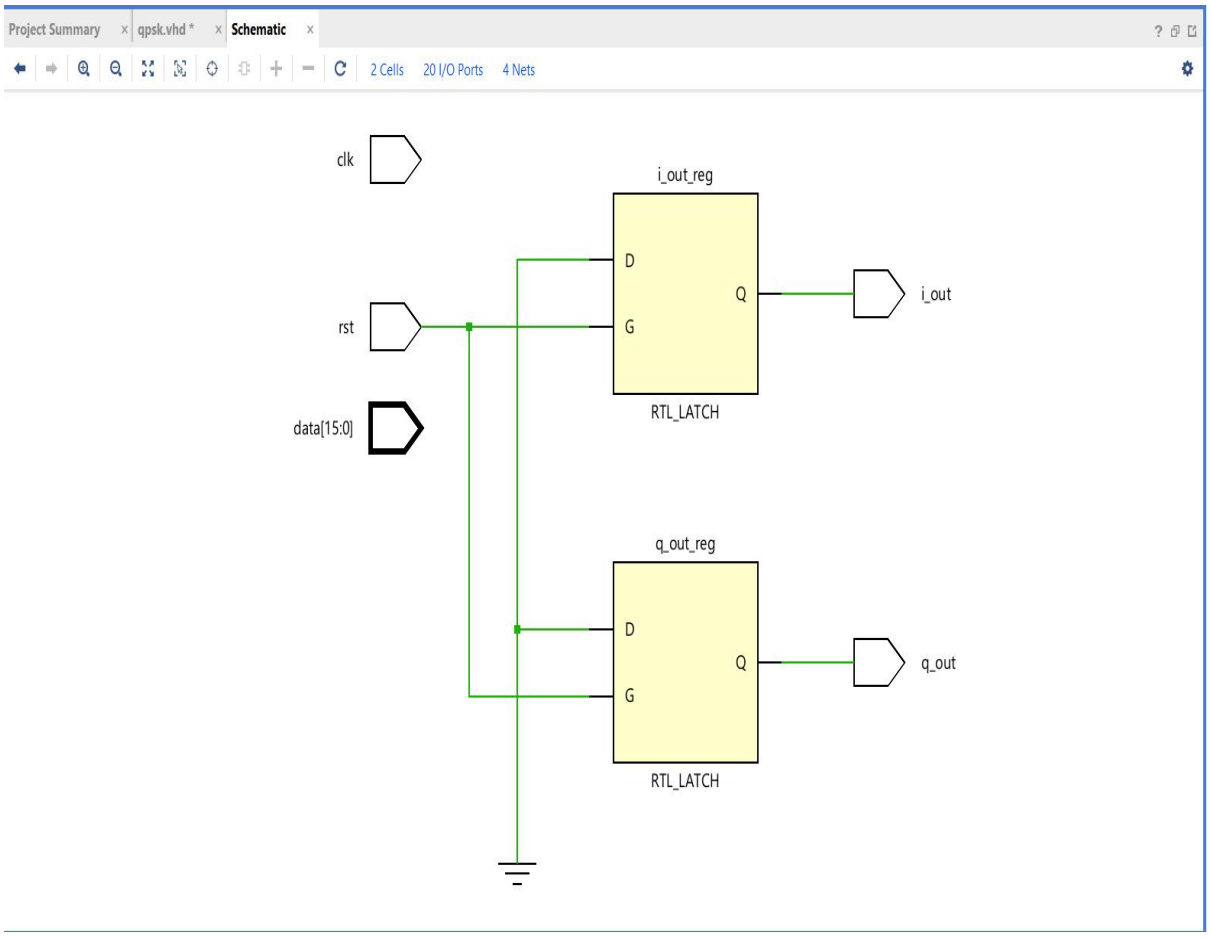
Vivado is a powerful integrated development environment (IDE) created by Xilinx for designing, testing, and implementing digital circuits on their field-programmable gate array (FPGA) and system on chip (SoC) devices. It provides a comprehensive suite of tools for digital design, simulation, synthesis, and implementation, as well as debug and verification capabilities.

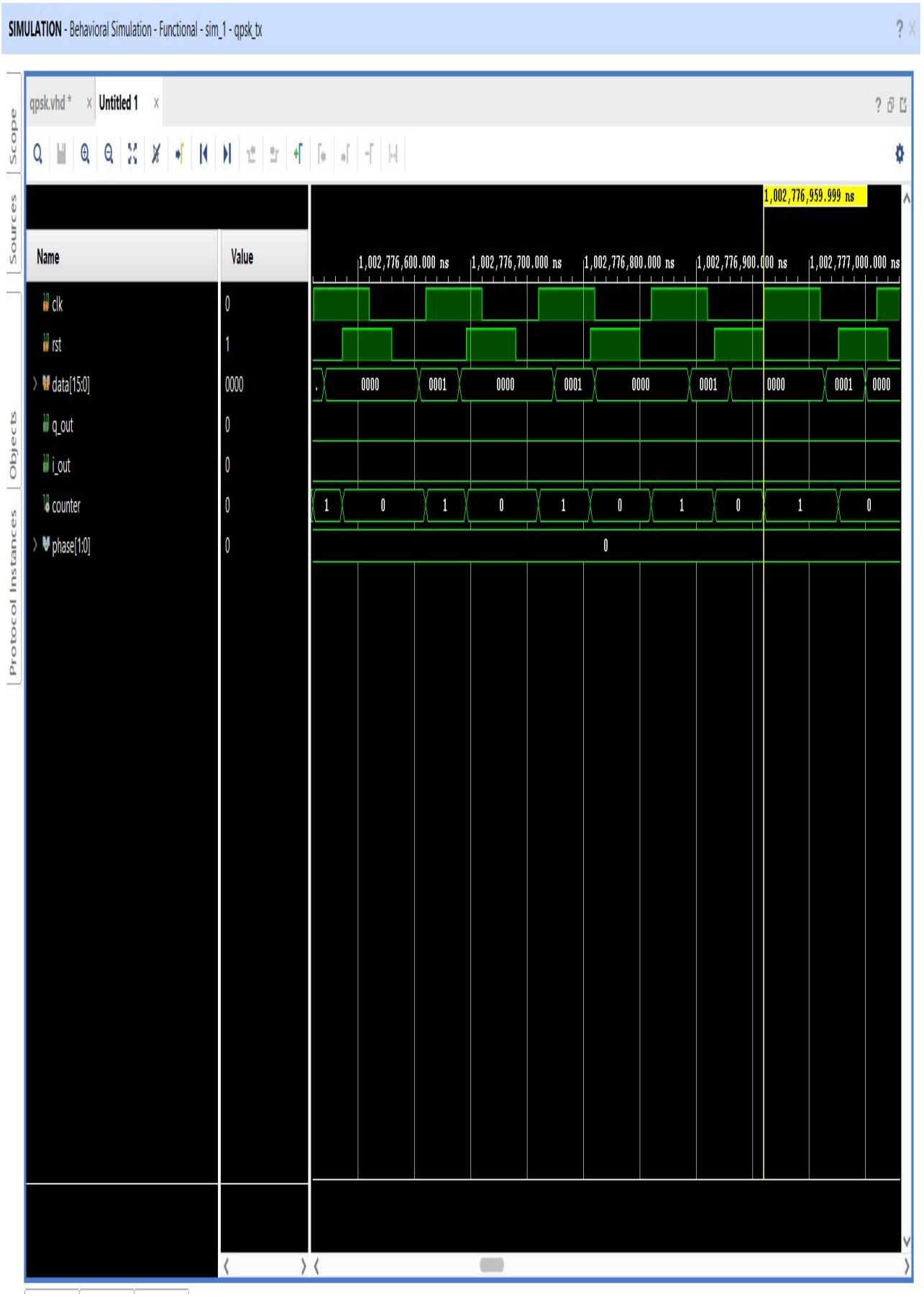
Some of the key features of Vivado include:

I. High-level synthesis (HLS): This feature allows designers to write and simulate their designs using high-level programming languages such as C, C++, and SystemC, and then automatically generate RTL (Register Transfer Level) code.
II. IP integrator: Vivado's IP integrator tool allows designers to easily integrate IP (intellectual property) blocks into their designs, including Xilinx's own IP cores, as well as third-party IP.
III. Timing analysis and optimization: Vivado includes a powerful timing analyzer that can identify and resolve timing violations in a design, as well as a set of optimization tools to help designers meet timing constraints.
IV. Debugging and verification: Vivado includes a suite of debugging and verification tools, including a logic analyzer, waveform viewer, and system-level debugger, to help designers quickly identify and resolve issues in their designs.
V. Partial reconfiguration: This feature allows designers to reconfigure parts of an FPGA while the rest of the design continues to operate, enabling flexible and dynamic system updates.
VI. Overall, Vivado is a comprehensive toolset that enables designers to efficiently develop and deploy complex digital designs on Xilinx's FPGA and SoC devices.

CHAPTER-5

SIMULATION:




SIMULATION - Behavioral Simulation - Functional - sim_1 - qpsk_tx
qpsk.vhd *
Untitled 1
Scope
Sources
Objects
Protocol Instances
Name
Value
clk
0
rst
1
data[15:0]
0000
q_out
0
i_out
0
counter
0
phase[1:0]
0
1,002,776,959.999 ns
1,002,776,600.000 ns
1,002,776,700.000 ns
1,002,776,800.000 ns
1,002,776,900.000 ns
1,002,777,000.000 ns
0000
0001
0000
0001
0000
0001
0000
0001
0000
1
0
1
0
1
0
1
0
1
0
0

Chapter-6

COURSE OUTCOME AND CONCLUSION

Project Outcome: Through this following project we conclude the outcome that a 16 bit qpsk can be easily designed by the software called as vivado

CO2 :

To introduce the VHDL Hardware Description Language (HDL) for front end design implementation

Conclusion: In conclusion, a 16-bit QPSK (Quadrature Phase Shift Keying) modulation and demodulation system can be implemented using VHDL. The QPSK modulation process involves dividing the input data into pairs of bits and mapping each pair to a specific phase shift in the QPSK constellation diagram. A carrier signal is then multiplied by the phase shift to generate the modulated signal. The modulated signals for each pair of input bits are summed to generate the final modulated signal.

On the other hand, QPSK demodulation is the reverse process, where the modulated signal is multiplied by a carrier signal, and the resulting signal is passed through a low-pass filter to remove any high-frequency noise. The resulting signal is then divided into pairs of bits, and each pair is mapped back to a specific phase shift to recover the original data.

Both the modulation and demodulation processes require the use of sine and cosine functions, which can be implemented using look-up tables or by calculating the values using the CORDIC algorithm. The implementation can be done using an FPGA or ASIC, and the design can be optimized for power consumption, speed, or area, depending on the application requirements.