PROJECT:ROM MEMORY DESIGN

Group Members:

Mohammad Akram (BU21EECE0100083)

T. Bharath (BU21EECE0100163)

C. Shanthi Priya (BU21EECE0100367)

D. Sai Bhargavi (BU21EECE0100454)

Priyanka. G (BU22EECE0100446)

Under the guidance of:

DR. ARUN KUMAR MANOHARN

DR.KARTHICK.S

INTRODUCTION:

This mini project focuses on designing and implementing a Read-Only Memory (ROM) using a Field-Programmable Gate Array (FPGA). FPGAs are highly versatile and reconfigurable devices that offer substantial performance and parallel processing capabilities, making them ideal for a wide range of digital system applications. This project aims to harness these features to create a reliable and efficient ROM memory system.

The primary objectives are to design a ROM module that can store a predefined set of data and to facilitate efficient data retrieval. Additionally, a user-friendly interface will be developed to enable easy access to the stored data. This project not only provides practical experience in FPGA programming but also covers fundamental aspects of digital electronics, including memory design, address decoding, and data retrieval mechanisms

Key tools and components for this project include an FPGA development board (such as Xilinx Vivado), the Verilog hardware description language, and user interface components like switches and LEDs for data access and visualization. The project scope encompasses designing the ROM architecture, implementing the address decoder, creating data retrieval mechanisms, and integrating and testing the complete system on the FPGA.

Overall, this project offers hands-on experience in digital system design and FPGA implementation, culminating in a fully functional ROM memory system. It serves as a valuable introduction to FPGA-based development and digital memory design, providing insights into both the theoretical and practical aspects of digital electronics.

OBJECTIVES:

Design a ROM Module : Implement a ROM that accurately stores and retrieves predefined data.

Data Initialization : Predefine and initialize the ROM with specific data sets

Create a User Interface : Develop a user-friendly interface for data access using switches, buttons, and LEDs.

Efficient FPGA Resource Utilization : Optimize the use of FPGA resources for ROM storage and retrieval functionalities.

Modular Design: Ensure the design is modular to facilitate easier testing, debugging, and potential future enhancements.

Simulation and Verification: Use simulation tools to verify the correctness of the ROM design before hardware implementation.

Synthesis and Implementation: Synthesize the design using FPGA development tools and implement it on an FPGA development board

Testing and Validation: Test the integrated ROM system on the FPGA board to ensure accuracy, reliability, and user interface responsiveness.

Performance Optimization: Analyze and optimize the ROM for speed, area, and power consumption.

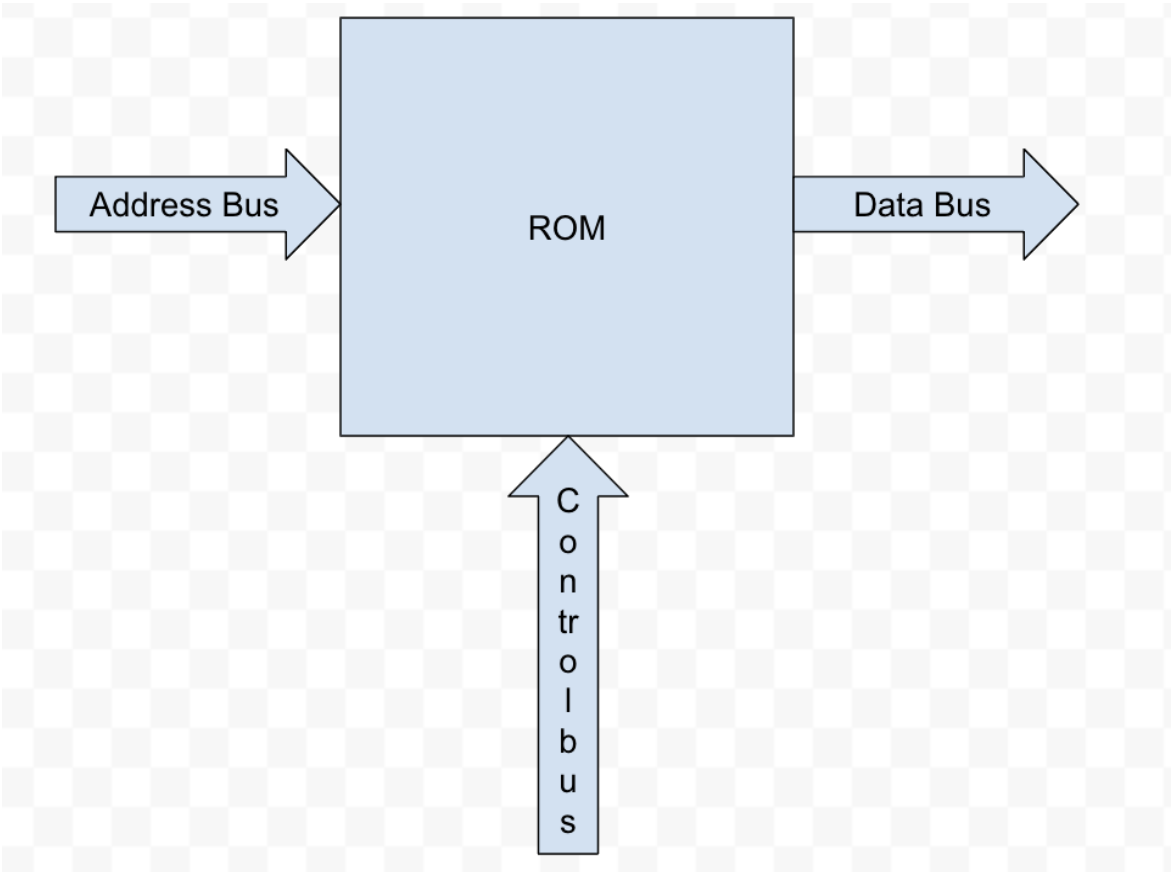
Scalability: Design the ROM to be scalable for future expansions and larger data storage needs.

METHODOLOGY/APPARATUS REQUIRED:

Hardware Used:
FPGA kit

Software Used:
Xilinx vivado

Block Diagram:



CODE FOR ROM (DUAL PORT RAM)

```
module dual_port_ram(
 input [7:0] data_a, data_b, // input data
 input [5:0] addr_a, addr_b, // Port A and Port B addresses
 input we_a, we_b, // write enable for Port A and Port B
 input clk, // clock
 output reg [7:0] q_a, q_b // output data at Port A and Port B
);

 reg [7:0] ram [63:0]; // 8-bit wide, 64-depth RAM

 always @ (posedge clk)
  begin
   if(we_a)
    ram[addr_a] <= data_a;
   else
    q_a <= ram[addr_a];
  end
 always @ (posedge clk)
  begin
   if(we_b)
    ram[addr_b] <= data_b;
   else
    q_b <= ram[addr_b];
```

```
  end

endmodule
```

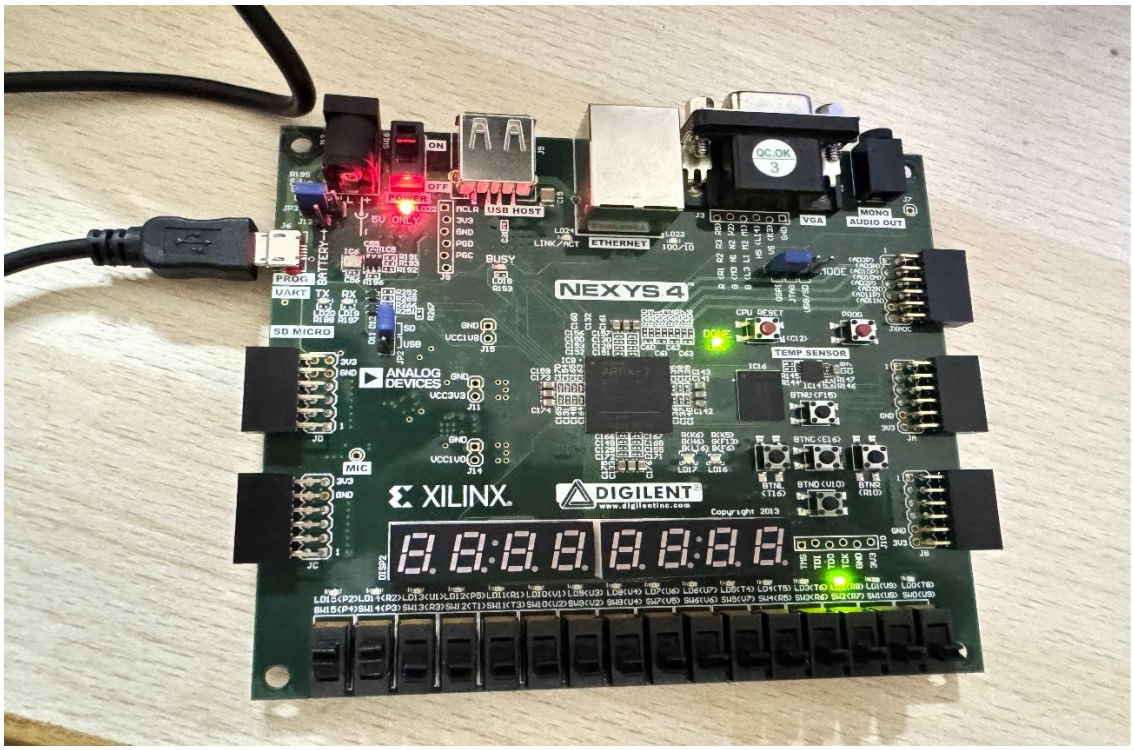
TESTBENCH CODE:

```
module dual_port_ram_tb;
 reg [7:0] data_a, data_b; // input data
 reg [5:0] addr_a, addr_b; // Port A and Port B address
 reg we_a, we_b; // write enable for Port A and Port B
 reg clk; // clock
 wire [7:0] q_a, q_b; // output data at Port A and Port B

 dual_port_ram dpr1(
  .data_a(data_a),
  .data_b(data_b),
  .addr_a(addr_a),
  .addr_b(addr_b),
  .we_a(we_a),
  .we_b(we_b),
  .clk(clk),
  .q_a(q_a),
  .q_b(q_b)
 );

 initial
  begin
   $dumpfile("dump.vcd");
```

```
    $dumpvars(1, dual_port_ram_tb);

    clk = 1'b1;
    forever #5 clk = ~clk;
  end

initial
  begin
    data_a = 8'h33;
    addr_a = 6'h01;

    data_b = 8'h44;
    addr_b = 6'h02;

    we_a = 1'b1;
    we_b = 1'b1;

    #10;

    data_a = 8'h55;
    addr_a = 6'h03;

    addr_b = 6'h01;

    we_b = 1'b0;
```

```
    #10;

    addr_a = 6'h02;

    addr_b = 6'h03;

    we_a = 1'b0;

    #10;

    addr_a = 6'h01;

    data_b = 8'h77;
    addr_b = 6'h02;

    we_b = 1'b1;

    #10;
  end

 initial
  #40 $stop;

endmodule
```

OUTPUT:


NEXYS 4
XILINX
DIGILENT
ANALOG DEVICES
ETHERNET
USB HOST
VGA
MONO AUDIO OUT
SD MICRO
TEMP SENSOR
CPU RESET
PROG
UART
MIC