CHIRALA ENGINEERING COLLEGE, CHIRALA

DEPAETMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DESIGN TOOLS LAB REPORT ON

RAM AND ROM DESIGN IN VERILOG

BY

K. BHANU PRASAD	-21E91A0421
P. NIVAS	-21E91A0436
P.V.SAI KIRAN REDDY	-21E91A0440
N. VENKATA SIVABALAJI	-22E95A0405

INTERNAL EXAMINER:

EXTERNAL EXAMINER:

HEAD OF THE DEPARTMENT:

RAM AND ROM DESIGN

ABSTRACT

This document explores the design and implementation of Random Access Memory (RAM) and Read-Only Memory (ROM) using VERILOG . RAM is a volatile memory type that supports both read and write operations, enabling quick data access essential for system performance. In contrast, ROM is a non-volatile memory designed for permanent data storage, typically containing firmware or unchangeable software.

The VERILOG design process for both memory types are crucial for simulating their behaviour and ensuring functionality in digital systems. These abstract outlines the key characteristics of RAM and ROM, their applications in modern electronics, and the significance of utilizing VERILOG for efficient memory design. By leveraging VERILOG, designers can create robust memory architectures, facilitating advancements in various electronic applications. This work underscores the importance of memory design in optimizing system performance and reliability.

TOOLS REQUIRED:

- MODELSIM – Simulation
- XILINX-ISE – Synthesis

CHAPTER-1

INTRODUCTION

In digital systems, memory plays a critical role in data storage and processing. Two primary types of memory are *Random Access Memory (RAM)* and *Read-Only Memory (ROM)*. Designing these components in VERILOG (VHSIC Hardware Description Language) allows for precise modelling and simulation of their behaviour in hardware applications.

RAM (Random Access Memory)

RAM is a type of volatile memory that enables both read and write operations. It is used for temporary storage of data that the processor needs to access quickly.

Key Characteristics:

Volatile Memory: Loses its contents when power is turned off.

Random Access: Allows for quick access to any memory location without a sequential search.

Dynamic and Static Variants: DRAM requires periodic refreshing, while SRAM retains data as long as power is supplied, offering faster access.

Design in VERILOG:

Designing RAM in VERILOG involves creating an architecture that can handle multiple read and write operations, often controlled by a clock signal. The design typically includes:

Address lines to select specific memory locations.

Data lines for input (write) and output (read) operations.

Control signals, such as write enable, to manage data flow.

ROM (Read-Only Memory)

ROM is a type of non-volatile memory used primarily for permanent data storage. It typically contains firmware or software that does not change during the device's operation.

Key Characteristics:

Non-Volatile Memory: Retains data even when power is lost.

Read-Only: Data is written during manufacturing and cannot be modified under normal conditions.

Fast Access: Allows for quick retrieval of stored data, especially during boot processes.

Design in VERILOG:

Designing ROM in VERILOG involves creating a fixed memory structure that can output data based on address inputs. Key components include:

An address input to select which data to retrieve.

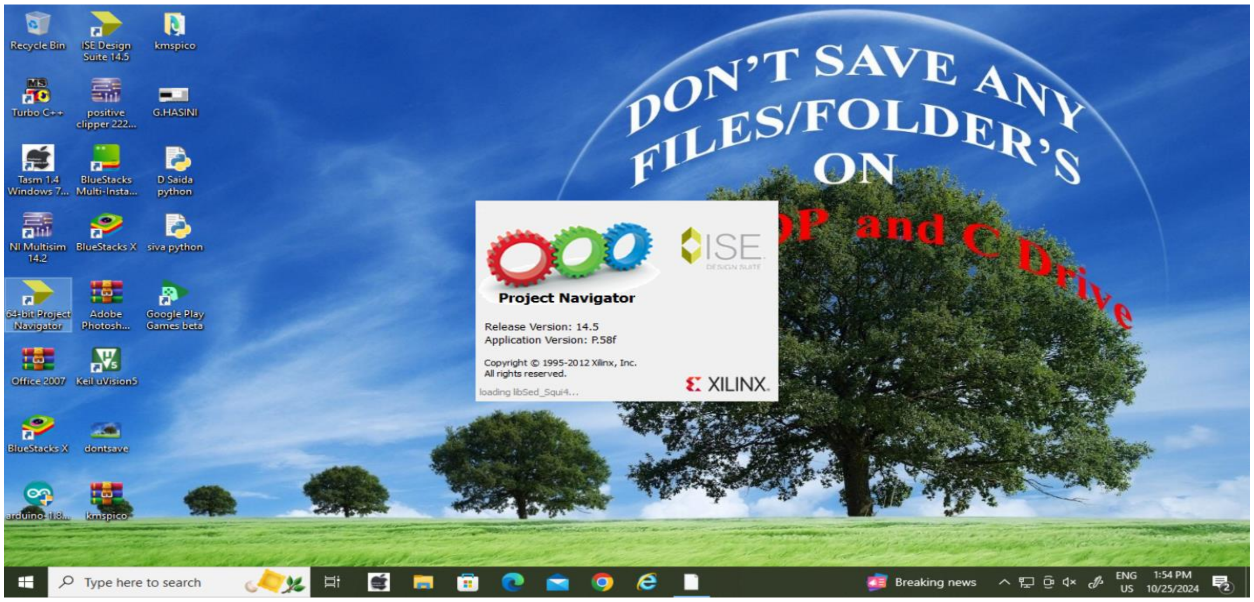
Data output lines to deliver the stored information.

An array or constant to hold the pre-defined values.

CHAPTER :2

XILINX PROCESS

Step 1: Start the Xilinx project navigator by using the desktop shortcut or by using the start.



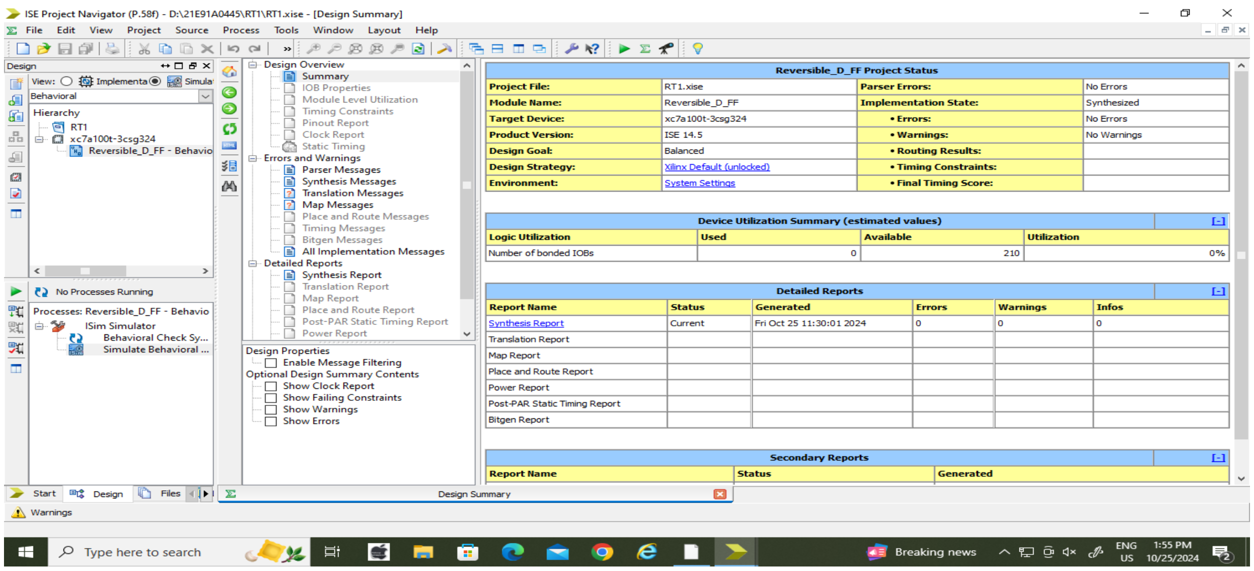
Project Navigator

Release Version: 14.5
Application Version: P.58f

Copyright © 1995-2012 Xilinx, Inc.
All rights reserved.

loading libSed_Squi4...

Step 2: In the project navigator window is open then click on ok and then click on new file



Reversible_D_FF Project Status

Project File:	RT1.xise	**Parser Errors:**	No Errors
Module Name:	Reversible_D_FF	**Implementation State:**	Synthesized
Target Device:	xc7a100t-3csg324	**• Errors:**	No Errors
Product Version:	ISE 14.5	**• Warnings:**	No Warnings
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values) [-]

Logic Utilization	Used	Available	Utilization
Number of bonded IOBs	0	210	0%

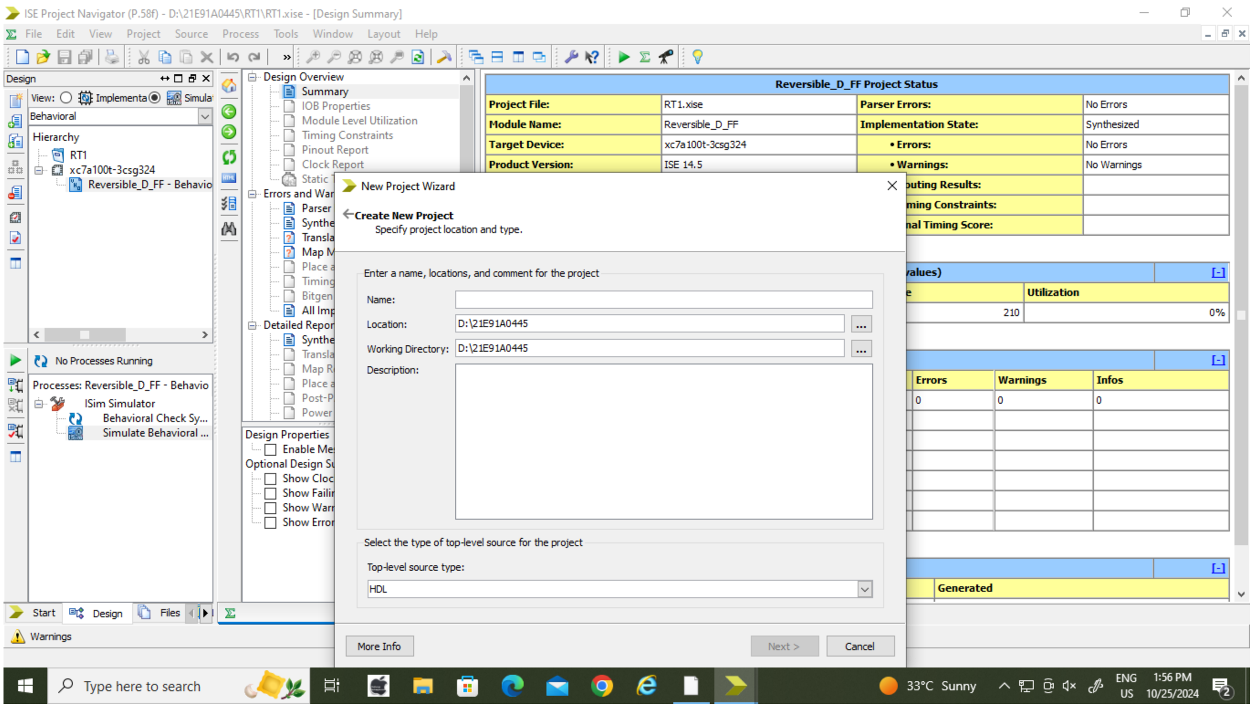
Detailed Reports [-]

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Fri Oct 25 11:30:01 2024	0	0	0
Translation Report					
Map Report					
Place and Route Report					
Power Report					
Post-PAR Static Timing Report					
Bitgen Report					

Secondary Reports [-]

Report Name	Status	Generated

Step 3: File is opened and give the file name.


ISE Project Navigator (P.58f) - D:\21E91A0445\RT1\RT1.xise - [Design Summary]

Reversible_D_FF Project Status			
Project File:	RT1.xise	Parser Errors:	No Errors
Module Name:	Reversible_D_FF	Implementation State:	Synthesized
Target Device:	xc7a100t-3csg324	• Errors:	No Errors
Product Version:	ISE 14.5	• Warnings:	No Warnings

New Project Wizard

Create New Project
Specify project location and type.

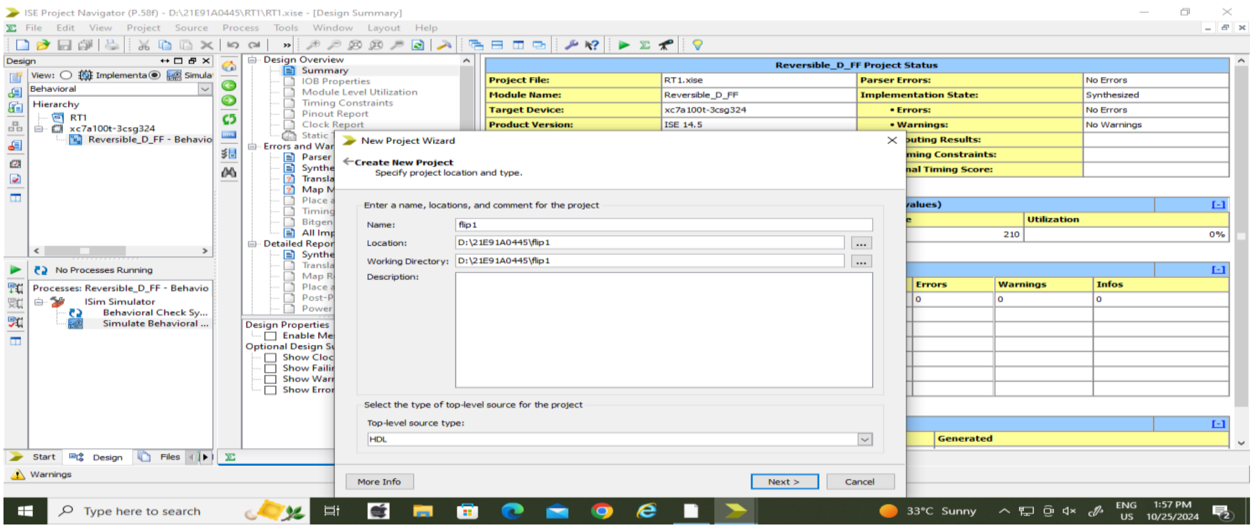
Enter a name, locations, and comment for the project

Name:
Location: D:\21E91A0445
Working Directory: D:\21E91A0445
Description:

Select the type of top-level source for the project

Top-level source type: HDL

Step 4: click on next.


New Project Wizard

Create New Project
Specify project location and type.

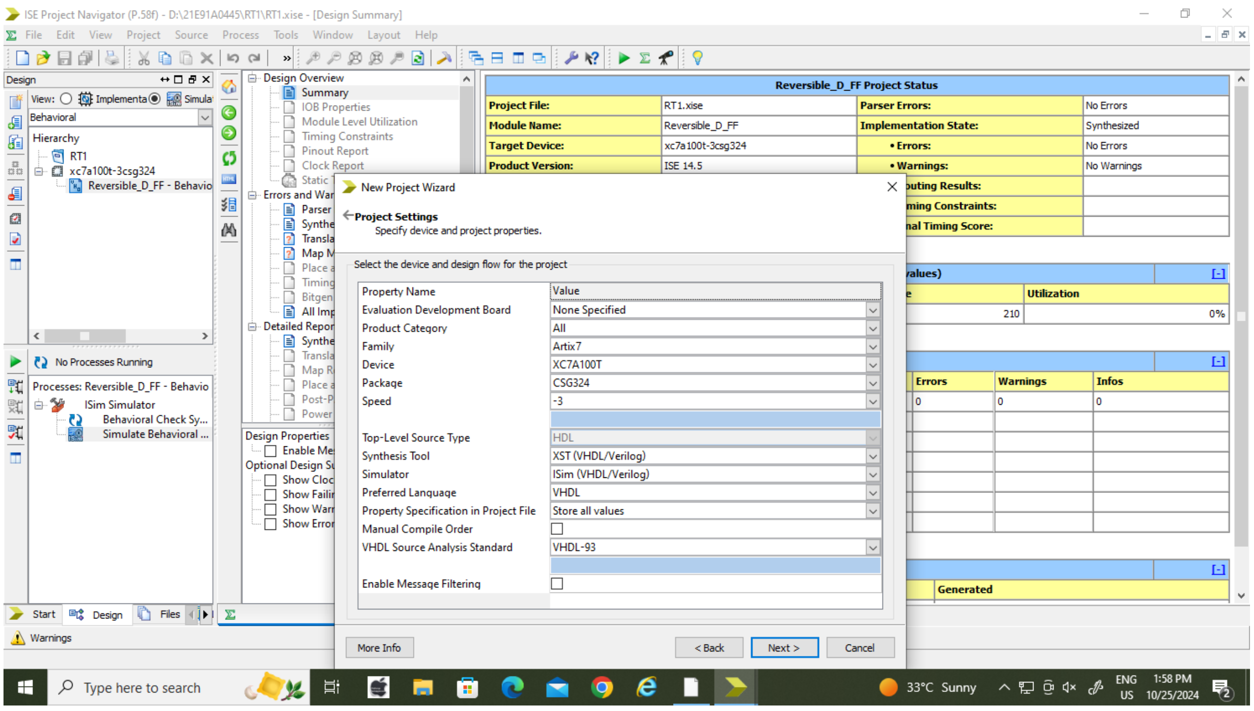
Enter a name, locations, and comment for the project

Name: flip1
Location: D:\21E91A0445\flip1
Working Directory: D:\21E91A0445\flip1
Description:

Select the type of top-level source for the project

Top-level source type: HDL

Step 5: click on next.


ISE Project Navigator (P.58f) - D:\21E91A0445\RT1\RT1.xise - [Design Summary]

Reversible_D_FF Project Status

Project File:	RT1.xise	Parser Errors:	No Errors
Module Name:	Reversible_D_FF	Implementation State:	Synthesized
Target Device:	xc7a100t-3csg324	• Errors:	No Errors
Product Version:	ISE 14.5	• Warnings:	No Warnings

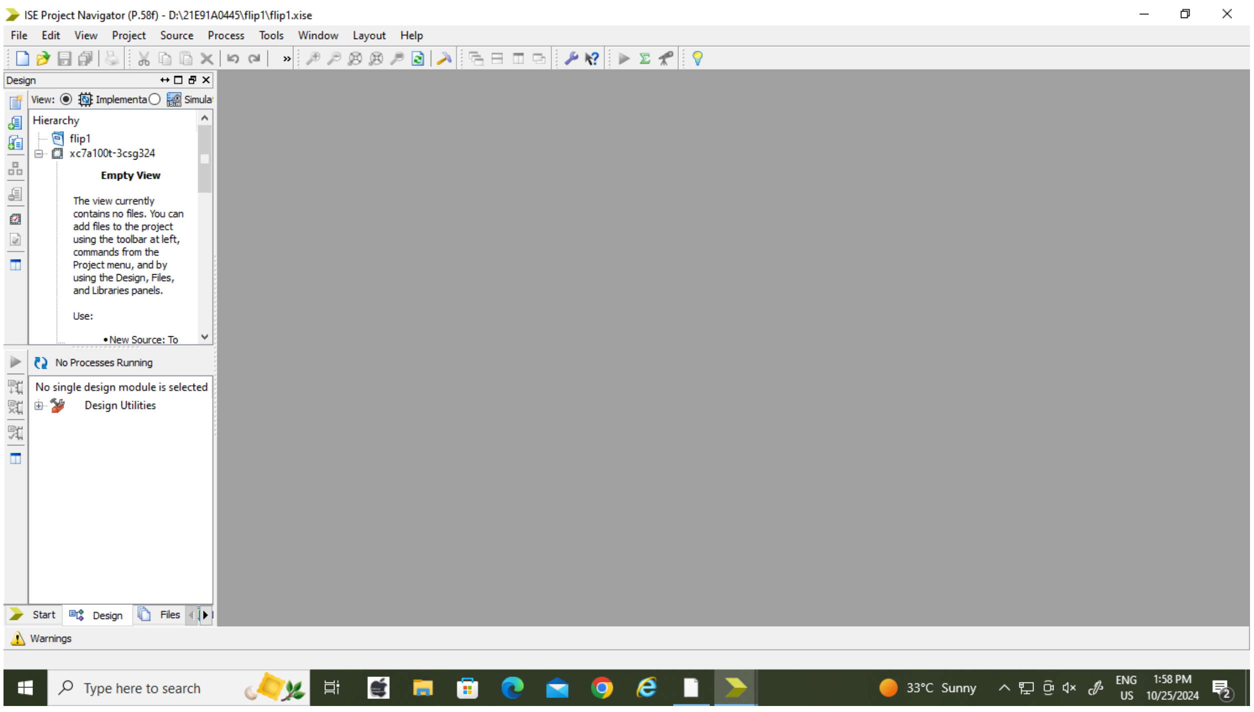
New Project Wizard

Project Settings
Specify device and project properties.

Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Artix7
Device	XC7A100T
Package	CSG324
Speed	-3
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

Step 6: Right click on file name then click on new source.


ISE Project Navigator (P.58f) - D:\21E91A0445\flip1\flip1.xise

Empty View

The view currently contains no files. You can add files to the project using the toolbar at left, commands from the Project menu, and by using the Design, Files, and Libraries panels.

Use:

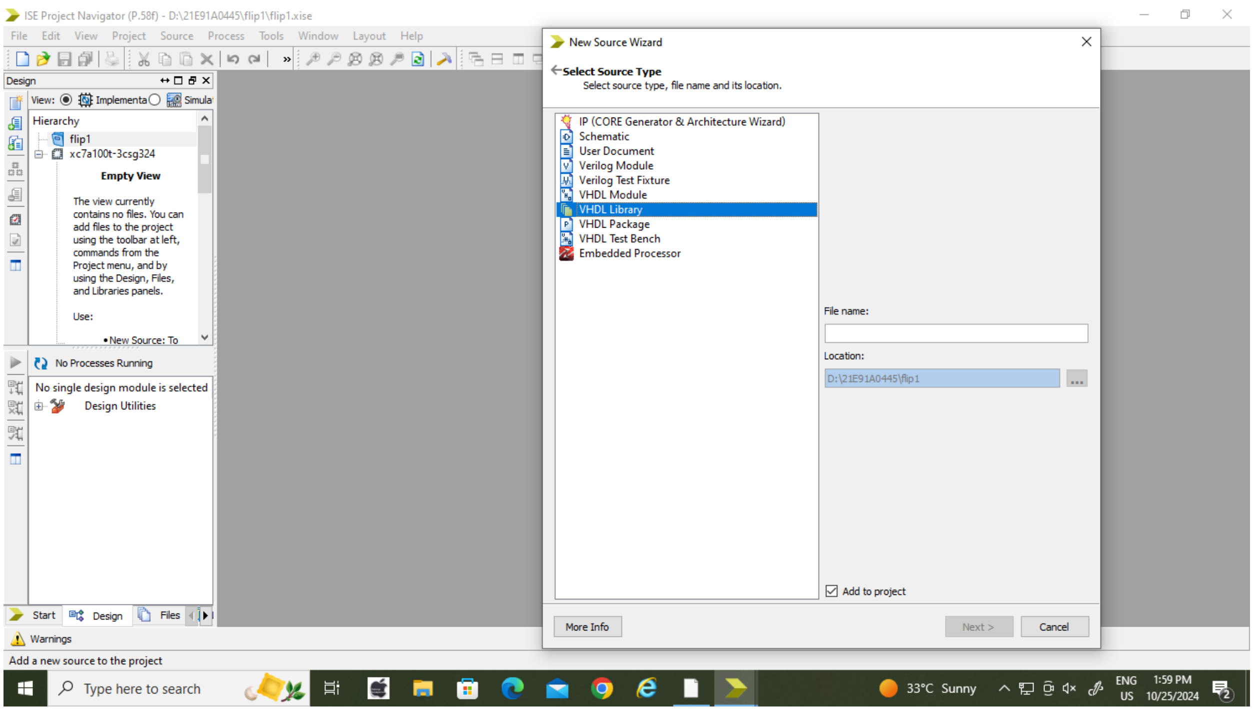
• New Source: To

No Processes Running

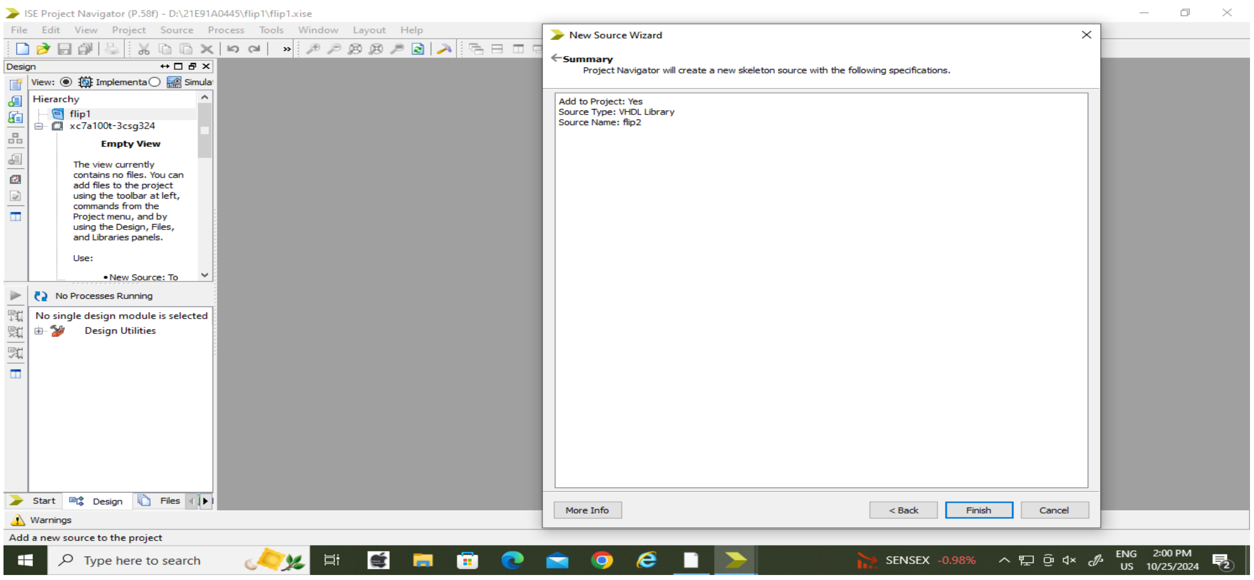
No single design module is selected

Design Utilities

Step 7: Click on schematic then give file name.



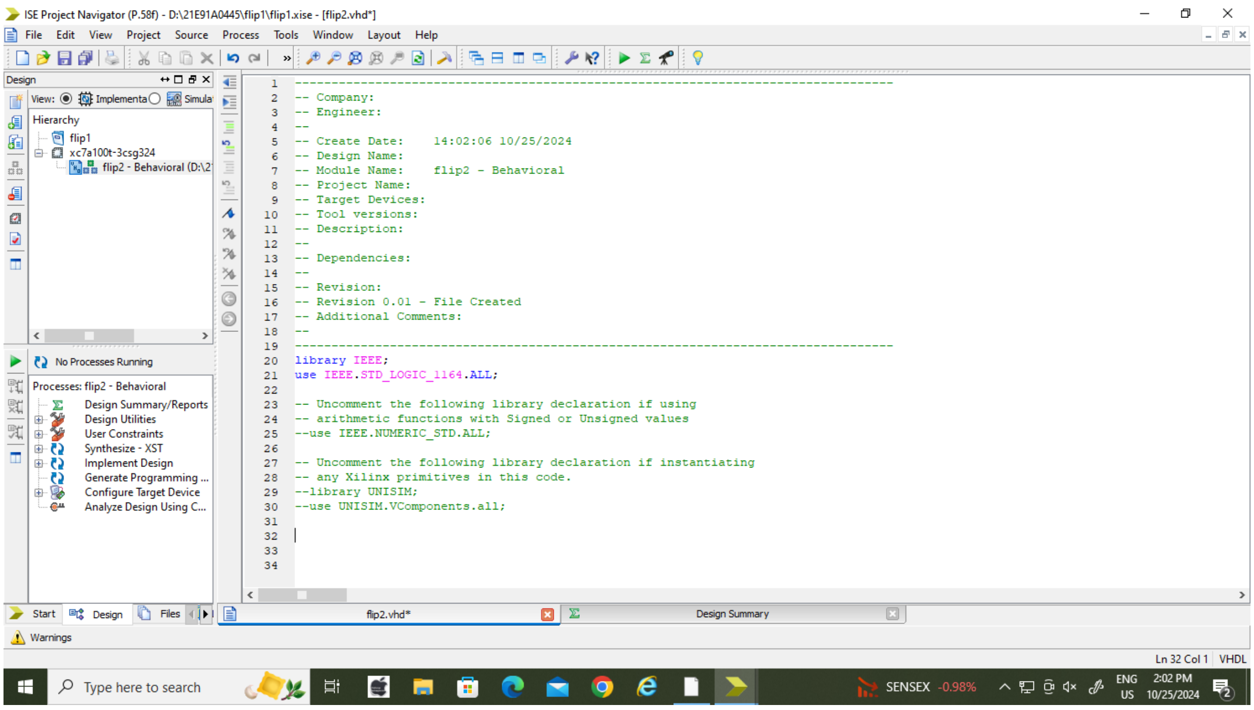
Step 8: Click on finish.


New Source Wizard
Summary
Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Type: VHDL Library
Source Name: flip2

More Info
< Back
Finish
Cancel

Step 9: Now schematic window is open.



```
--------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    14:02:06 10/25/2024
-- Design Name:
-- Module Name:    flip2 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
```

Step 10: Check syntax, and remove errors if present.

Simulate the design using ISE Simulator Highlight inverter file in the Sources in Project window.

To run the Behavioural Simulation, click on the symbol of FPGA device and then right click.

Click on new source -> Click on Verilog text fixture -> Give file name with tb -> finish.

*Generate test bench file after initial begin assign value for inputs -> Click on simulate behavioural model -> see the output.

What is a RAM (Random Access Memory)?

RAM (Random Access Memory) is a memory device that is used in the systems read/write and store the data. RAM is present ranging from small-scale systems such as embedded systems, smartphones to large-scale systems as Desktop, Laptops, etc. The RAM used in a system has a major impact on the system's performance. It's responsible for carrying out multiple tasks and storing data. The higher the version of a RAM, the higher it’s performance.

Now that we have the clear idea about what RAM is, let’s see how to implement RAM in VERILOG. Though there are higher versions of RAM that are used in the industry. In this tutorial, we are going to implement a simple 32X8 RAM. It’s a small-scale version of the memory device.

Specifications of RAM:

Usually, memory is represented by M x N, where M is the number of locations and N is the Data lines. So here 32 x 8, “32” represents 32 locations, or an array that has 32 locations, “8” represents 8 data lines. In simpler terms, a 32 x 8 RAM has 32 locations and every 32 locations can store 8-bit data. So, the total number of bits that can be stored by a 32 x 8 RAM is 256 bits. Now, you can imagine the storage capacity of 8GB, 16GB RAM that’s used in laptops/desktops.

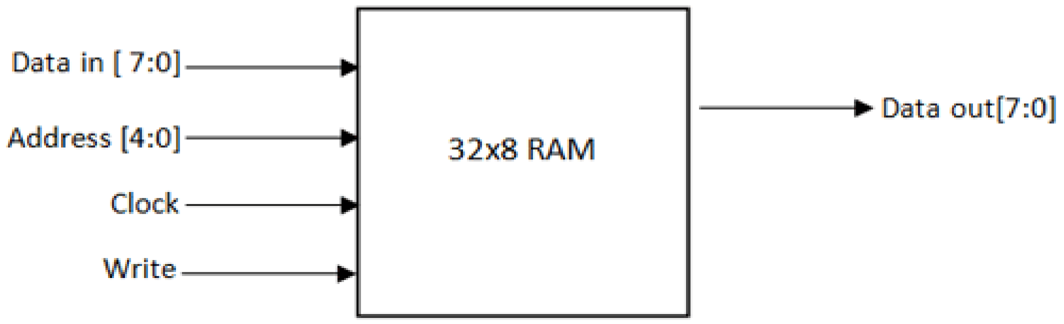


Fig-2.1 Schematic diagram of RAM:

A RAM has a data line, address line, write signal, clock signal, and data out port to read the contents from the RAM. The address line size/bits vary depending on M x N specifications. Let's understand how to calculate the size of the address line bits (A). As shown in Fig 2.1

Address Line bits/size (A): 2^A=M. So, in our case, it is 2^A=32 (M=32), where A=5. An→ Address Line

M → Number of Locations

N → Data Lines

W → Write signal

Since RAM is a sequential circuit, it has a clock signal. All sequential circuits have a clock signal. Also, the write signal takes care of burning the data into the RAM. Only when both clock, and the write signal is “1” the data is stored in the RAM.

The events that will take place concerning the write and clock signals are given below.

Clock	Write Signal	Event
0	0	No data loaded
0	1	No data loaded
1	0	No data loaded
1	1	Data will be loaded

Table 2.1 lookup table.

Types of RAMS (Random Access Memory)

1.Dynamic RAM (DRAM):

Description: Stores data in capacitors; requires periodic refreshing.

Applications: Main memory in computers and laptops.

2.Static RAM (SRAM):

Description*: Uses latching circuitry; no refresh needed.

Applications*: Cache memory in CPUs; faster but more expensive.

3.Synchronous Dynamic RAM (SDRAM):

Description: Synchronized with the system clock for faster access.

Applications: Modern desktops and laptops.

4.Double Data Rate SDRAM (DDR SDRAM):

Description: Transfers data on both clock edges, doubling the data rate.

Applications: Widely used in PCs and gaming consoles.

What is a ROM (READ ONLY MEMORY)?

A Read Only Memory (ROM) is a device that includes both the decoder and the OR gates within a single IC package. The Fig. 3.82 shows the block diagram of ROM. It consists of n input lines and m output lines. Each bit combination of the input variables is called an address. Each bit combination that comes out of the output lines is called a **word**. The number of bits per word is equal to the number of output lines, m. The address specified in binary number denotes one of the minters of n variables. The number of distinct addresses possible with an input variable is 2^n. An output word can be selected by a unique address, and since there are 2^n distinct addresses in a ROM, there are 2^n distinct words in the ROM. The word available on the output lines at any given time depends on the address value applied to the input lines.

specifications of ROM

- **Non-volatile**: ROM retains data even when the power is off.
- **Read-only**: Data in ROM can only be read and cannot be easily modified.
- **Static**: ROM does not need refreshing every time it's used.
- **Slower than RAM but faster than a hard drive**: ROM is generally slower than RAM but faster than a hard drive.

ROM is used in many devices, including:

- **Computers**: ROM contains the basic input/output system (BIOS) that instructs the computer's boot-up processes.
- **Video game consoles**: ROM allows one system to run various games
- **Optical storage**: ROM is used in compact discs (CD) such as CD-ROM and CD-RW.
- **Calculators**: ROM is frequently used in calculators.
- **Peripheral devices**: ROM is used in peripheral devices like laser printers, whose fonts are commonly stored in ROM.

schematic diagram of ROM:

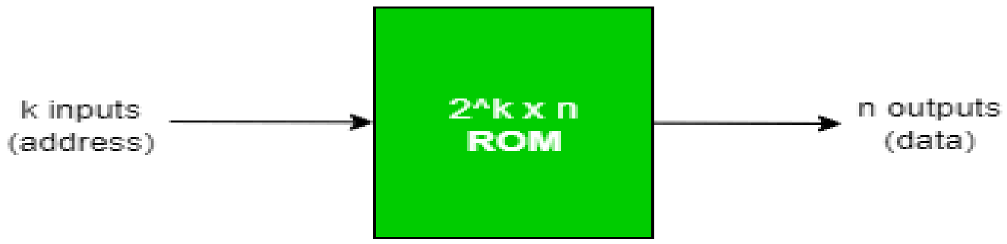


Fig-2.2 Schematic diagram of ROM:

1.Address Lines:

Function: The address lines are used to specify which memory location in the ROM is being accessed. Each address line represents a bit in a binary address.

Example: For a ROM that can address 256 locations, 8 address lines (A0 to A7) would be required (since 2^8 = 256).

2. Data Lines:

Function: The data lines carry the data output from the ROM to the external circuit. The number of data lines corresponds to the number of bits that can be read simultaneously.

Example: An 8-bit ROM will have 8 data lines (D0 to D7), allowing it to output 8 bits of data at a time.

Types of ROM (Read-Only Memory):

Read-Only Memory (ROM) comes in several types, each with unique characteristics and applications. Here are the primary types of ROM:

1.Masked ROM (MROM):

Description: Data is permanently programmed during the manufacturing process. The mask used during fabrication defines the memory content.

Characteristics:

Inflexible; cannot be modified after production.

Cost-effective for large-scale production where data is fixed.

Applications:

Used in devices where firmware does not change, such as basic consumer electronics.

2. Programmable ROM (PROM):

Description: Users can write data to the memory once after manufacturing.

Characteristics:

Data is programmed using a special device.

Once written, it cannot be erased or modified.

Applications:

Suitable for applications where the firmware needs to be loaded post-manufacturing but not changed.

3. Erasable Programmable ROM (EPROM):

Description: Data can be erased and reprogrammed using UV light.

Characteristics:

Contains a transparent window that allows UV light to erase the stored data.

Can be reused multiple times, though it requires external equipment for erasure. Applications:

Used in development and testing environments where the ability to modify firmware is necessary.

4.Electrically Erasable Programmable ROM (EEPROM):

Description: Data can be erased and reprogrammed using electrical signals.

Characteristics:

Allows for individual bytes to be rewritten without needing to erase the entire chip.

Slower than RAM and has limited write cycles compared to flash memory.

Applications:

Commonly used for storing small amounts of data that must be preserved when power is lost, such as BIOS settings in computers.

5. Flash Memory:

Description: A type of EEPROM that allows data to be erased and reprogrammed in blocks.

Characteristics:

Non-volatile and widely used due to its durability and speed.

Supports multiple write and erase cycles, making it suitable for frequent updates.

Applications:

Used in USB drives, SSDs, and memory cards.

VERILOG CODE FOR SINGLE PORT RAM:

```
module RAM1(
  input [7:0] data,
  input [5:0] addr,
  input we,
  input clk,
  output [7:0] q );
  reg [7:0] ram [63:0];
  reg [5:0] addr_reg;
  always @ (posedge clk)
  begin
  if(we)
  ram[addr]<=data;
  else
   addr_reg<=addr;
   end
   assign q = ram [addr_reg];
endmodule
```

VERILOG CODE FOR SINGLE PORT ROM:

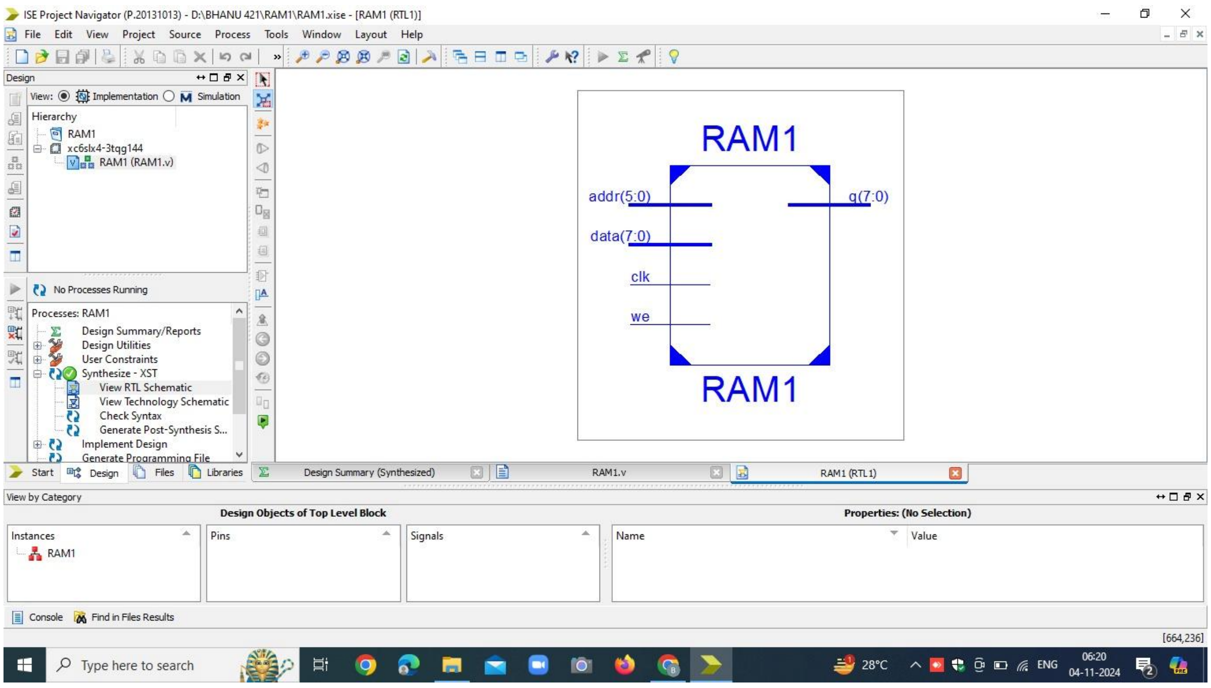
```
module ROM1(
input clk, //clk
input en, //enable
input [3:0] addr, //address
output reg [3:0] data //output data);
reg [3:0] mem [15:0]; //4 bit data
always @ (posedge clk)
  begin
   if (en)
   data <= mem[addr];
   else
   data <= 4'bxxxx;
   end
 initial
   begin
     mem[0] = 4'b0010;
     mem[1] = 4'b0010;
     mem[2] = 4'b1110;
     mem[3] = 4'b0010;
     mem[4] = 4'b0100;
     mem[5] = 4'b1010;
     mem[6] = 4'b1100;
     mem[7] = 4'b0000;
```

```
    mem[8] = 4'b1010;

    mem[9] = 4'b0010;

    mem[10] = 4'b1110;

    mem[11] = 4'b0010;

    mem[12] = 4'b0100;

    mem[13] = 4'b1010;

    mem[14] = 4'b1100;

    mem[15] = 4'b0000;

  end

end module
```

CHAPTER- 3

RESULT'S

RTL SCHEMATIC DIAGRAM for single port RAM:



ISE Project Navigator (P.20131013) - D:\BHANU 421\RAM1\RAM1.xise - [RAM1 (RTL1)]
RAM1
addr(5:0)
data(7:0)
clk
we
q(7:0)
RAM1

Fig.3.1 **RTL SCHEMATIC DIAGRAM for single port RAM**

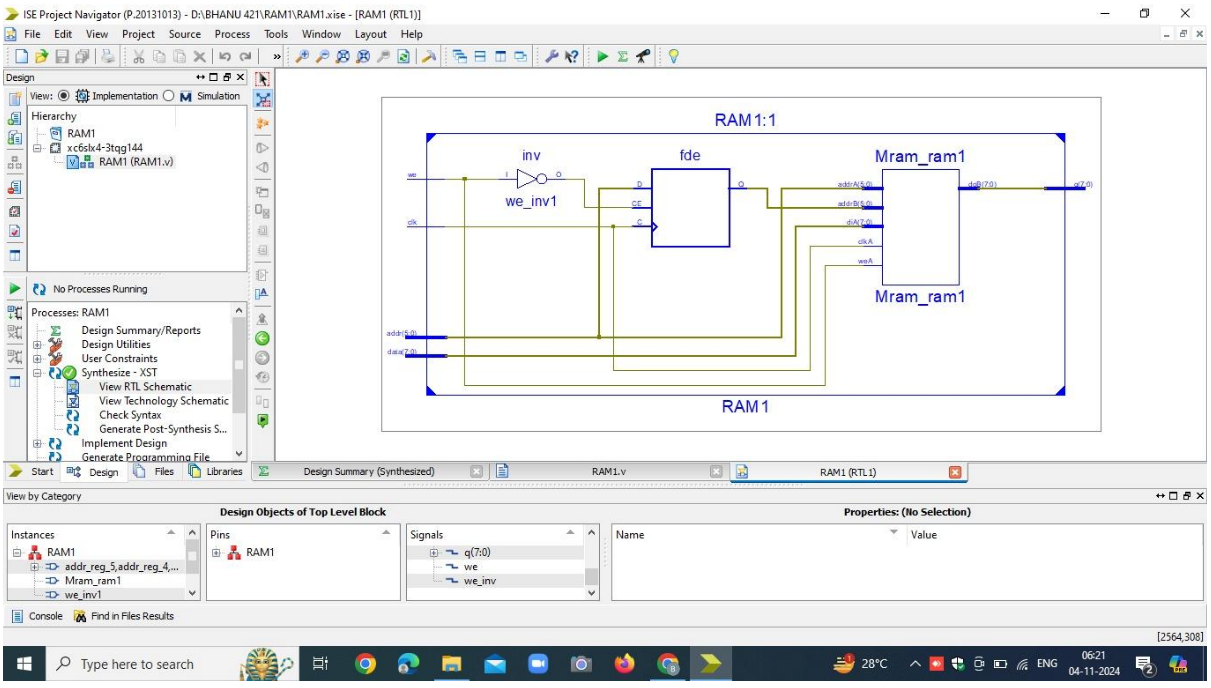


Fig.3.2 **RTL SCHEMATIC DIAGRAM-2 for single port RAM**

Schematic Diagram Explanation for Single-Port RAM:

A typical schematic diagram for a *Single-Port RAM* includes the following major components as shown in Fig3.1 and Fig3.2

1. Address Bus (A):

A set of input lines used to specify the location (address) in memory that the CPU or other devices want to read from or write to.

The number of address lines (A) determines the memory depth, i.e., how many locations the RAM can hold (e.g., 2^N locations if there are N address lines).

2. Data Bus (D):

A bidirectional bus that transfers data to and from the memory. It connects the memory to other parts of the system (like the processor or I/O devices). The width of the data bus (e.g., 8 bits, 16 bits) determines how much data can be transferred at a time.

3. Control Signals:

These signals govern the behaviour of the RAM. Some typical control signals include:

Read/Write (R/W):

This signal determines whether the RAM will perform a read or write operation.

Chip Select (CS):

This signal activates or deactivates the memory chip. When CS is active, the memory can perform operations; when inactive, the chip ignores all inputs.

Output Enable (OE):

This signal controls whether the data on the data bus is driven by the RAM chip (for read operations).

4. Memory Array:

The core of the RAM consists of an array of memory cells. Each cell stores a bit of data. The cells are organized in rows and columns, with rows corresponding to addresses and columns storing the data.

5. Read/Write Logic:

This logic determines the operation based on control signals. It allows the data at the specified address to either be written to (during a write operation) or read from (during a read operation).

How it Works:

Write Operation:

The address to which data will be written is provided via the address bus.

The data to be written is provided via the data bus.

The write signal is activated (R/W signal is high or low depending on design), and the data at the specified address is updated with the new value.

Read Operation:

The address from which data will be read is provided via the address bus.

The read signal is activated (R/W signal is flipped to the read mode).

The data from the specified address is placed on the data bus and can be read by other components.

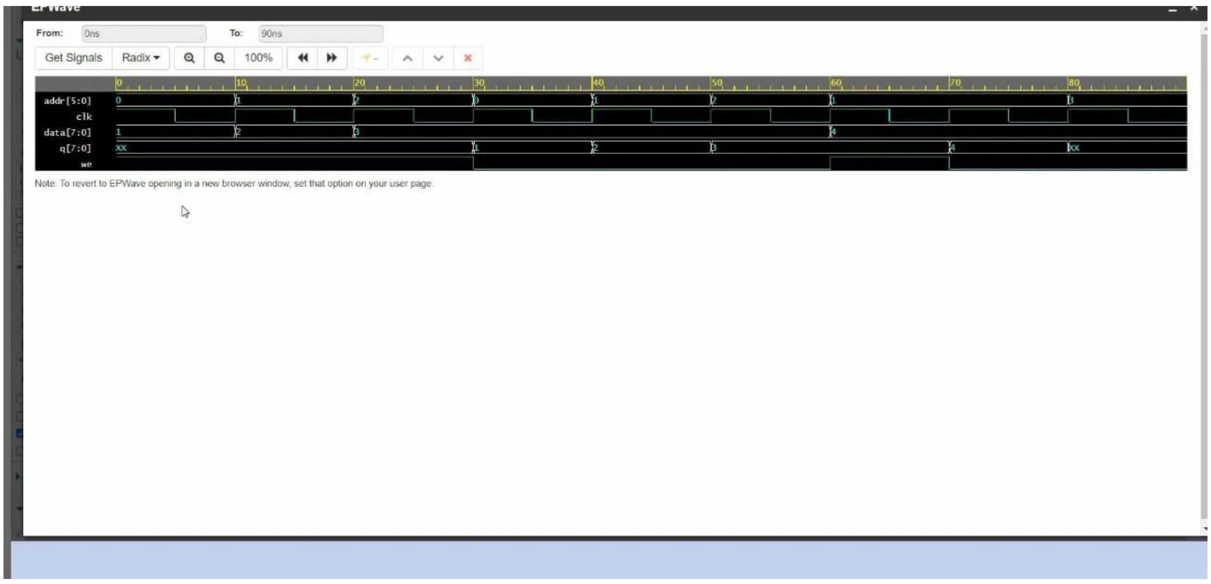
Key Characteristics of Single-Port RAM:

Single Port Access: Only one access to the memory is possible at a time, meaning a CPU or controller can either read or write data but not both simultaneously.

Simple Control: Compared to dual-port RAM, it is simpler to implement and requires fewer control lines.

Slower Access: Because reading and writing must occur one after the other, it may be slower compared to dual-port RAM in applications requiring simultaneous read and write operations.

OUTPUT WAVEFORMS:



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

Fig.3.3 output wave form for single port RAM

The output waveform of a Single-Port RAM (Random Access Memory) illustrates the behaviour of the RAM during a read or write operation. Since a Single-Port RAM can only handle one operation (read or write) at a time, the output waveform reflects how data is written to or read from the memory based on the control signals and address inputs. As shown in Fig.3.3.

Key Control Signals in the Output Waveform for Single-Port RAM:

1. Address Bus (A):

The address bus provides the address to the RAM, specifying the memory location to read from or write to.

The address is stable and valid for the duration of the read or write cycle.

2. Data Bus (D):

The data bus is the bidirectional bus that carries the data being written to or read from the RAM.

During a write operation, data is driven onto the data bus from the system (CPU or controller) to the RAM.

During a read operation, data is driven from the RAM onto the data bus to be read by the system.

3. Control Signals:

Chip Select (CS): Enables or disables the RAM chip. If CS is active (usually low), the RAM chip responds to the address and control signals. If CS is inactive (high), the RAM is disabled.

Read/Write (R/W): This signal determines whether the RAM is performing a read or write operation. If R/W = 0, the operation is a write, and if R/W = 1, the operation is a read.

Output Enable (OE): This signal controls whether the data is output from the RAM onto the data bus during at read operation. When OE is active (low), the data is driven onto the bus; when OE is inactive (high), the data bus is in a high-impedance state (i.e., no data is output).

Read and Write Operations in Single-Port RAM:

Let's explore the waveforms for both read and write operations in a Single-Port RAM.

1. Read Operation:

During a read operation, the address is provided, the RAM is enabled (via CS), the R/W signal is set to read mode (1), and the OE signal is activated.

Explanation of Read Waveform:

1. Address Bus (A):

The address is placed on the address bus by the system. This address specifies which location in the RAM the system wants to read from.

The address remains valid for the duration of the read cycle.

2. Chip Select (CS):

CS is low (active) for the duration of the read cycle, enabling the RAM chip.

If CS were high (inactive), the RAM would ignore the address and control signals, and no data would be read.

3. R/W Signal:

The R/W signal is set to high (1) to indicate that the operation is a *read* operation.

When R/W = 1, the RAM knows it is a read cycle.

4. Output Enable (OE):

The OE signal is low (active), enabling the RAM to place the data from the selected memory address onto the data bus.

If OE were high, the RAM would be in a high-impedance state, and no data would be driven to the data bus.

5. Data Bus (D):

The data corresponding to the given address is placed on the *data bus* by the RAM.

For example, if the address 0x03 corresponds to the data 0xA3, then 0xA3 would appear on the data bus at the corresponding time.

2. Write Operation:

During a write operation, the address is provided, the RAM is enabled (via CS), the R/W signal is set to write mode (0), and the OE signal is deactivated (high, indicating no output to the data bus).

Explanation of Write Waveform:

1. Address Bus (A):

The address is placed on the address bus by the system. This address specifies which memory location in the RAM the data will be written to.

The address remains valid for the duration of the write cycle.

2. Chip Select (CS):

The CS is low (active) during the write cycle, enabling the RAM chip to respond to the address and control signals.

If CS were high (inactive), the RAM would ignore the write operation.

3. R/W Signal:

The R/W signal is low (0), indicating that the operation is a *write* operation.

When R/W = 0, the RAM understands that data should be written to the specified address.

4. Output Enable (OE):

The OE signal is high (inactive) during the write cycle, meaning the RAM does not drive data onto the data bus.

In a write operation, the data to be written is placed *on the data bus* by the CPU or controller, and the RAM will accept the data from the data bus.

5. Data Bus (D):

The data bus holds the data that the system wants to write to the RAM. During the write operation, the data is transferred from the system onto the data bus and is written to the memory location specified by the address.

If the address is 0x03, and the data to be written is 0x7F, the data bus will carry 0x7F during the write cycle.

Key Differences Between Read and Write Operations:

Read Operation:

R/W is high (1) for a read operation.

OE is low (enabled) to output data from the RAM to the data bus.

Data is placed on the data bus from the RAM.

Write Operation:

R/W is low (0) for a write operation.

OE is high (disabled) during the write, meaning the RAM does not drive data to the bus.

Data is placed on the data bus by the CPU/controller, and the RAM writes it to the specified address.

RTL SCHEMATIC DIAGRAM for single port ROM:

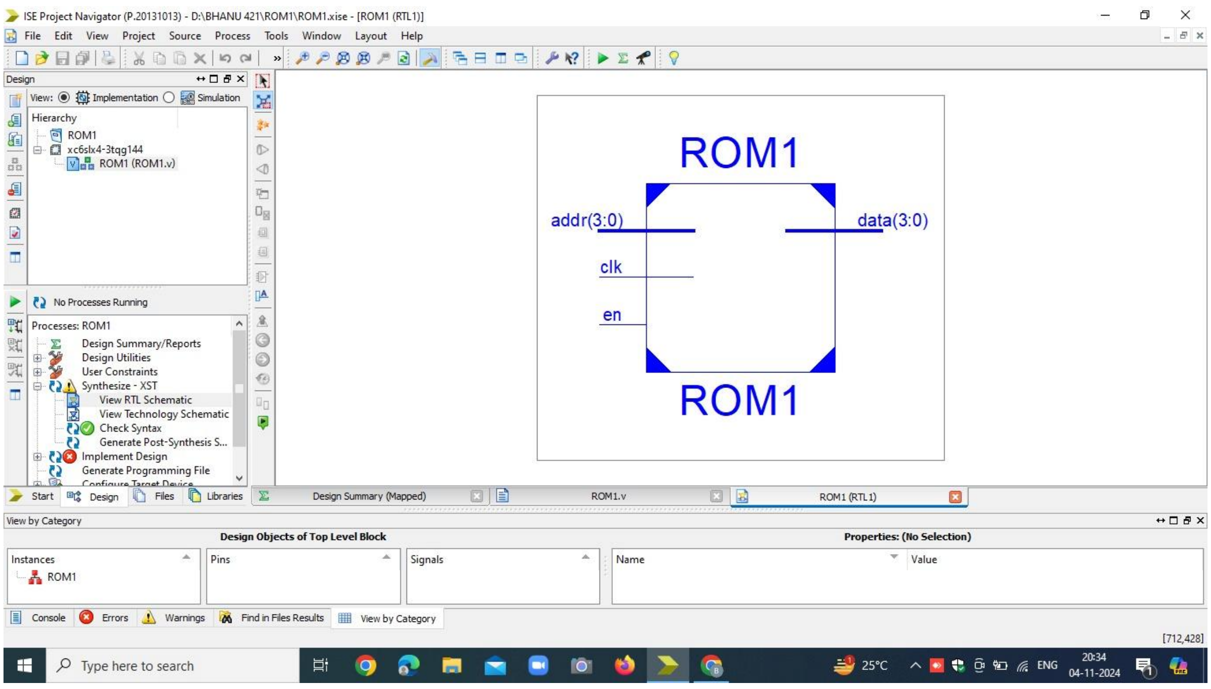
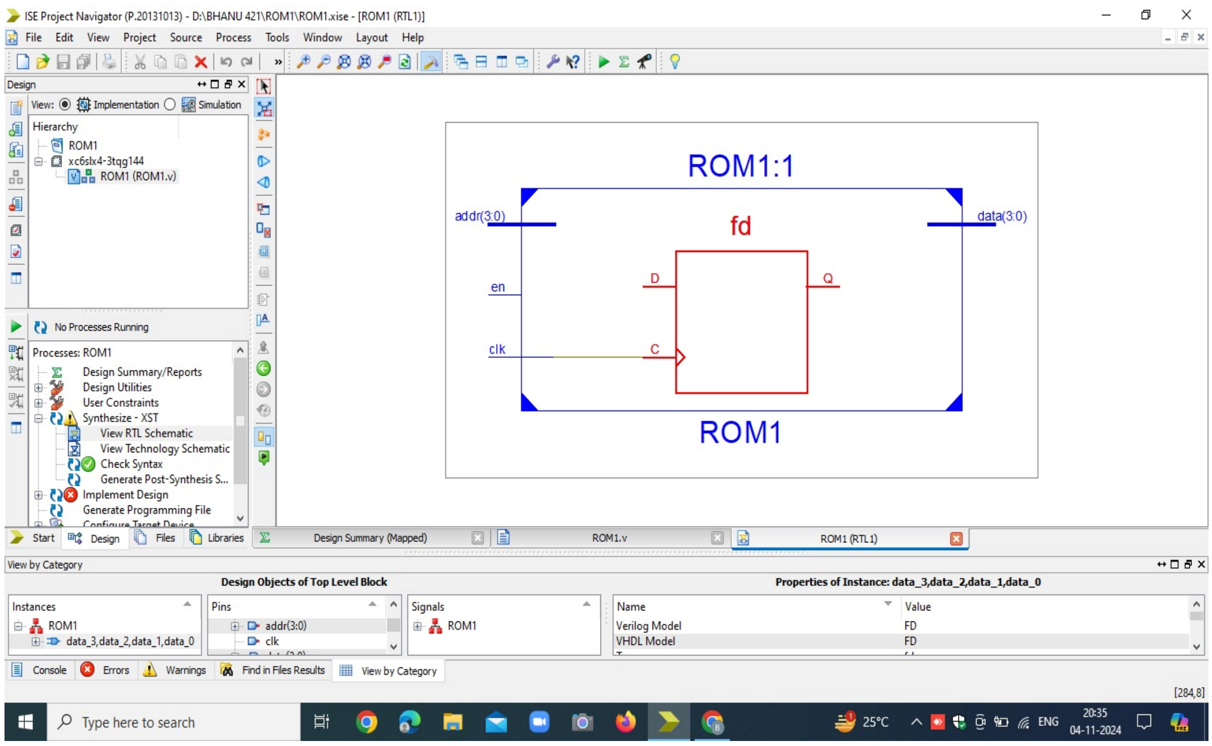


Fig.3.4 **RTL SCHEMATIC DIAGRAM-1 for single port ROM**



ISE Project Navigator (P.20131013) - D:\BHANU 421\ROM1\ROM1.xise - [ROM1 (RTL1)]

Design Objects of Top Level Block

Instances	Pins	Signals
ROM1	addr(3:0)	ROM1
data_3,data_2,data_1,data_0	clk	

Properties of Instance: data_3,data_2,data_1,data_0

Name	Value
Verilog Model	FD
VHDL Model	FD

[284,8]

Fig.3.5 **RTL SCHEMATIC DIAGRAM-2 for single port ROM**

Schematic Block Diagram Explanation for a Single-Port ROM:

A typical schematic diagram for *Single-Port ROM* includes the following major components. As shown in Fig.3.4 and 3.5.

1. Address Bus (A):

The address bus is used to specify the location in memory from which data will be read. The number of address lines determines the size of the ROM, i.e., how many locations it can store.

2. Data Bus (D):

The data bus is a unidirectional bus that carries data from the ROM to other parts of the system (like the CPU). The width of the data bus (e.g., 8 bits, 16 bits) determines how much data can be transferred at a time.

3. Control Signals:

Chip Select (CS): This signal enables or disables the ROM chip. When CS is active, the chip responds to address inputs; when inactive, the chip ignores all inputs.

Output Enable (OE): This signal controls whether the data on the data bus is actively driven by the ROM (for reading data). When OE is active, the ROM places data on the data bus.

4. Memory Array:

The core of the ROM consists of an array of memory cells. The ROM is typically designed so that data can only be read but not written to. Each memory cell stores a fixed bit of data (either 0 or 1), and the entire ROM is pre-programmed during manufacturing.

5. Decoder Logic:

The decoder logic is responsible for selecting the correct memory location based on the address input. It decodes the address bus and activates the corresponding line in the memory array, allowing the data stored at that location to be placed on the data bus.

How it Works:

Read Operation:

The address to read from is provided on the address bus.

The *Chip Select (CS)* signal is activated to enable the ROM.

The *Output Enable (OE)* signal is activated, allowing the ROM to place the data from the specified address onto the data bus.

The data is then available for use by the processor or other components.

Since ROM is read-only, there is no write operation in the diagram—data cannot be changed once it has been programmed into the ROM.

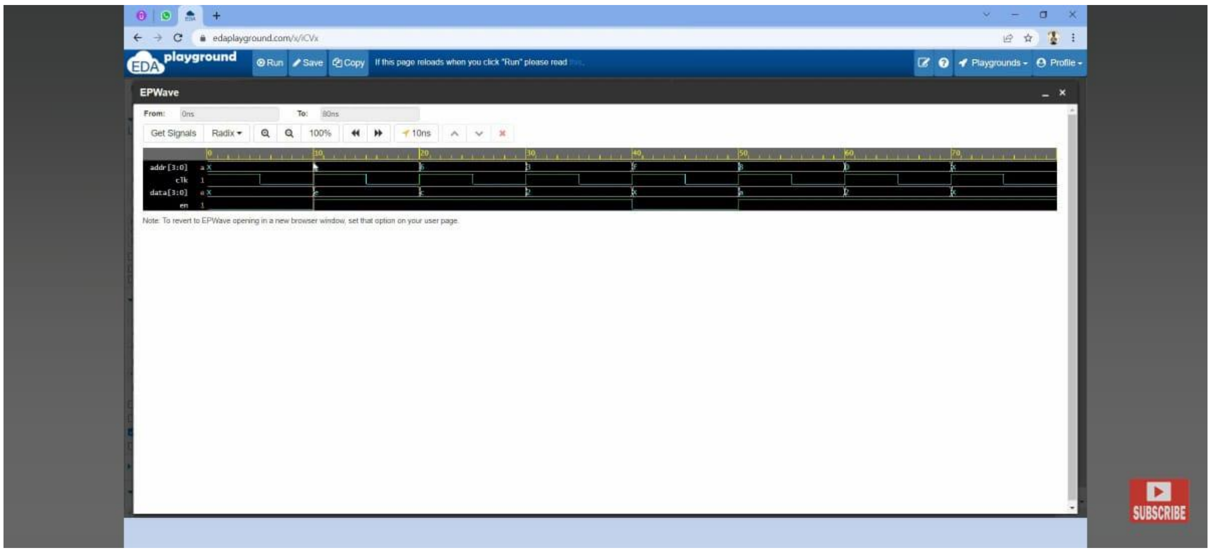
Key Characteristics of Single-Port ROM:

Single-Port Access the ROM can only be accessed through a single port (read-only), meaning it cannot handle multiple read or write operations at the same time.

Read-Only: Data is stored in ROM during manufacturing and cannot be modified during normal operation.

Fast Read Operations ROM provides fast access to fixed data, making it ideal for storing programs or configurations that do not need to change.

OUTPUT WAVEFORMS:



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

Fig.3.6 output wave form for single port RAM

The output waveform of a Single-Port ROM illustrates how the ROM responds to address inputs and control signals during the read operation. Since ROM is read-only, the output waveform primarily focuses on the read cycle when data is accessed from the ROM. As shown in Fig.3.6

Key Control Signals in the Output Waveform:

1. Address (A): The input address to the ROM that selects the memory location to be read.

2. Chip Select (CS): A control signal that enables or disables the ROM chip. If CS is low (active), the ROM will respond to the address and provide data; if CS is high (inactive), the ROM will not respond to the address.

3. Output Enable (OE): A control signal that enables or disables the ROM's data output. When OE is active (typically low), the ROM places the data from the selected address onto the data bus. If OE is inactive (typically high), the data bus is in a high-impedance state and no data is output.

4. Data Bus (D): The output lines through which data is transmitted from the ROM to other system components (e.g., CPU).

Description of the Output Waveform:

The waveform of a read operation in a Single-Port ROM would look something like this:

Waveform Components:

1. Address (A) Waveform:

This waveform represents the address being sent to the ROM. The address is valid and stable for a specific period, typically the duration of the read cycle. The address could change from one location to another as the system moves through different locations in ROM.

2. Chip Select (CS) Waveform:

The CS signal must be *low* for the ROM to be active. When CS is high, the ROM is disabled and does not respond to the address input. When CS is low, the ROM is enabled, and it can read from the memory. CS will typically stay low during the entire read cycle.

3. Output Enable (OE) Waveform:

The OE signal controls whether the data bus is driven by the ROM. During a read operation, OE must be low to enable the ROM to place data on the data bus.

When OE is low, the ROM places the data from the address location on the data bus. If *OE* is high, the data bus is in a high-impedance state, and no data is output from the ROM.

4. Data Bus (D) Waveform:

This waveform shows the data that is output from the ROM when the correct address is selected, and the ROM is enabled.

The data on the data bus is typically valid only when both *CS* and *OE* are active. The ROM places the corresponding data onto the data bus for a specific address.

Detailed Explanation of Waveforms:

1. Address Bus (A):

The address bus holds the address value for the ROM. For example, in an 8-bit ROM with 256 addresses, the address can range from 0x00 to 0xFF. Each time the address changes, the ROM will read the value stored at that address.

2. Chip Select (CS):

The chip select signal is low (active) during the entire read cycle, meaning the ROM is selected and can drive its data onto the data bus. If *CS* is high (inactive), the ROM would be deselected, and no data would be output.

3. Output Enable (OE):

The OE signal is low (active) for the duration of the read cycle. This ensures that when the ROM is selected and the address is provided, the ROM outputs the corresponding data onto the data bus. If OE is high, the ROM is effectively disconnected from the data bus, and no data will be driven.

4. Data Bus (D):

The data bus shows the data that is output from the ROM. At the time when the address is stable and both CS and OE are active, the data corresponding to that address will appear on the data bus.

For example, if the ROM's address 0x03 corresponds to the data 0x7A, then the data bus will show 0x7A during the time window when the address is 0x03, and CS and OE are active.

Key Points to Remember:

Address Changes: The address will change to access different memory locations, and each new address will fetch different data from the ROM.

Chip Select (CS): Must be low for the ROM to output data. If CS is high, the ROM is disabled and does not output data.

Output Enable (OE): Must be low to enable data output onto the data bus. If OE is high, the ROM is in a high-impedance state, and no data is output.

Data Bus (D): Reflects the data output by the ROM at the given address. Data changes as the address changes.

Example Read Cycle:

Assume you're reading data from address 0x05:

1. Set the address bus to 0x05 (binary: 00000101).

2. Activate *CS* (low) to select the ROM.

3. Activate *OE* (low) to allow the ROM to drive the data onto the data bus.

4. The data at address 0x05 (let’s say 0xA3) is placed on the data bus.

5. Once the data is available on the bus, the data can be read by other components (e.g., the CPU).

LIMITATIONS:

While designing RAM and ROM in VERILOG offers significant benefits, there are several limitations to consider:

1. Complexity in Timing and Control Logic:

Designing complex memory systems, especially for RAM, requires careful attention to timing, synchronization, and control logic (e.g., read/write enable signals). This can lead to intricate VERILOG code that might be difficult to manage and debug, especially for large designs.

2. Resource Utilization:

Memory structures like RAM and ROM in VERILOG can consume significant FPGA or ASIC resources. A poorly optimized design might lead to excessive use of logic cells, block RAM, or chip area, affecting performance and power consumption.

3. Limited by Hardware Resources:

The size of RAM and ROM in VERILOG designs is often constrained by the physical memory available on the FPGA or ASIC. While VERILOG models can simulate large memory, actual hardware may have limits that require designers to balance memory capacity with other resources.

4. Speed Limitations:

While VERILOG models can be fast in simulation, real-world implementation may face delays due to propagation times, especially in high-speed applications. Ensuring that read and write operations occur within the required timing constraints can be challenging.

5. Lack of Built-In Optimization:

VERILOG is a hardware description language and doesn't inherently include advanced memory optimization techniques like automatic caching or virtual memory, which means designers must manually address memory efficiency and optimization.

6. Limited Abstraction:

VERILOG offers a low level of abstraction for memory design, which can lead to longer development times compared to higher-level memory management tools or programming languages. It requires designers to be familiar with the hardware-level details of memory management.

7. Testing and Debugging:

Simulating large memory systems in VERILOG can be resource-intensive and time-consuming. Additionally, debugging memory issues such as race conditions or incorrect memory access requires detailed inspection and is often complex.

8. Platform Dependency:

VERILOG designs are sometimes platform-specific (e.g., targeting certain FPGA architectures), meaning that a design might need modification to work across different hardware platforms.

CHAPTER-4

CONCLUSION

Designing RAM (Random Access Memory) and ROM (Read-Only Memory) in VERILOG provides an essential foundation for implementing memory systems in digital circuits. By leveraging VERILOG, designers can specify and simulate memory behaviours with flexibility and precision, enabling efficient use of resources. The implementation of RAM typically requires the creation of read and write operations, with attention to timing and data flow, while ROM focuses on providing a fixed set of data values accessible via an address. Key challenges include ensuring correct data access patterns, managing timing constraints, and optimizing the memory's size and performance. Ultimately, the effective use of VERILOG in RAM and ROM design aids in achieving reliable, scalable, and optimized memory solutions for embedded systems and digital hardware applications.