ABSTRACT

Single Port RAM using VHDL code

In Modern World, We See a people who depended on Technology to make their job easy.

Some of the technologies are in the form of computer and mobile phones due to their speed and accuracy in result.

What makes it so fast to make a work easy?

Random Access Memory also known as RAM is the factor for the device to make the work simple and accurate. It is the hardware present inside a computer .

The purpose of RAM is to store the user data temporarily and also known as working memory. Also RAM is faster than the ROM that are present in computer, As ROM only serves a storage disk which makes it slow when compared to RAM.

RAM copies a file from the ROM that are needed at that time I.e., when you open a particular app or software , It only copies the file of that particular software and stores in it temporarily until the current or source is powering the RAM or device. RAM is also known as volatile form of storage.

Our project Is also based on one of the type of RAM I.e., Single Port RAM . The Main purpose of Our Project is about to study on working and components of Single Port RAM.

Since the modern world is improving with improving technology, There is still backlog in speed and accuracy in data retrieval in some field.

Even though Single Port RAM is older compared to present world, The speed and accuracy is still best when compared to DRAM or any other normal RAM (Since DRAM acts as Sync ,Hence its speed is quite low Compared to SRAM).

By the technique and Principle of studying , We can able to create a better version of Single Port RAM in terms of Speed and efficiency.

CHAPTER 01

INTRODUCTION

- Randomaccess memory is a one of the form of the computer memory that can be read and altered (write) in any order, mainly used to store working data and machine code.
- A randomaccess memory device allows data items to be read or written in almost the same amount of time irrespective of the physical location of data inside the memory of the system.
- This is a VHDL project presents a VHDL code for a singleport RAM (Random Access Memory). The VHDL testbench code is also provided to test the single-port RAM in Xilinx ISE software. The RAM's size is 128x8 bit.
- Single port memory : It has only one data or address port , it can either read or write at a time.
- The experiment is about construction of a basic ram that has single port that are used earlier years by VHDL codes.
- This project is concerned about construction of single port ram using VHDL code and execution of same in hardware form using IC'S or using FPGA kit.

If you do not want to create instances of RAM primitives keep your HDL code technology independent, XST offers automatic RAM recognition capability. XST can infer distributed as well as block RAM.

It covers the following features, which of these types of RAM offer:

- Synchronous write

- Write enable
- RAM enable
- Asynchronous or synchronous read
- Reset of the data output latches

- Single, dual or multiple-port read
- Single-port write

The type of inferred RAM depends on its description:

- RAM descriptions with an asynchronous read generate a distributed RAM macro

- RAM descriptions generate a Block RAM macro with a synchronous read. In some cases, a Block RAM macro actually can be implemented with Distributed RAM. The decision on the actual RAM implementation is done from the macro generator.

.

CHAPTER 02

LITERATURE SURVEY

Title of the paper	Author and year of publication	Outcome	Limitation
FPGA prototyping by verilog Examples:Xilinx Spartan-3 Version	Pong P.Chu Cleveland state university 2008 by john wiley and sons,Inc.	The book defines the single port ram with synchronous and asynchronous read and its synthesis report.	This publication deals with theoretical aspects of single port ram rather than practical circuit.
A Basic overview of types of random access memory(RAM)	Peter haugen Ian Myers	The paperwork deals with the study of various types random access memory and that have information on our RAM that we are using(SPRAM).	The literature has only the theoretical aspects of ram types and there is no practical connection with our project.
VHDL SPRAM design example v1.0 Readme file.	Altera corporation intel March 2010	This contains VHDL SPRAM design and this publication is about the single port ram design.	The paper work contains only codes but no information on single port ram.

Table 2.1: Literature Survey

CHAPTER 03

PROPOSED METHODOLOGY

- This example describes a 128 bit x 8 bit single-port RAM design in Verilog HDLwith common read and write address . Synthesis tools are used to detectdesigns in the HDL code of singleport RAM and depending on the architecture of the target device it automatically to infer either the **altsyncram** or the **altdpram** megafunctions.

- It's write operation is always synchronous.The embedded memory of a Spartan-6 device is already wrapped with a synchronous interference .

- The address, input data, and relevant control signals, such as we (i.e., write enable), are sampled on the rising edge of the clock, . A write operation is performed ,if we is asserted (i.e., the input data is stored in the memory location designated by the address signal). The read operation can be asynchronous or synchronous.

- For asynchronous read, the address signal is used directly to access the RAM array. After the address signal changes, the data becomes available after a short delay.

- For synchronous read, the address signal is sampled at the rising edge of the clock and stored in a register. The registered address is then used to access the RAM array. Because of the register, the availability of data is delayedand it is synchronized by the clock signal. An asynchronous read operation can be realized only by the distributed RAM, due to the internal structure.

CHAPTER 04

PROJECT DESCRIPTION

The Single-Port Block Memory module is generated based on the user-specified width and depth. This module for Virtex and Spartan-IIis composed of single or multiple 4 Kb blocks called SelectRAM+. The Virtex-4, Virtex-II, Virtex-II Pro, and Spartan-3 Single-Port

Block Memory modules, on the other hand, are composed of single or multiple 18 Kb blocks called Select RAM-II. Since Virtexand Spartan-II both use the 4 Kb

SelectRAM+ blocks, any particular reference to a Virtex implementation also applies to a Spartan-II, Virtex-E, Virtex-II Pro, or Spartan-IIE implementation. Similarly, because Virtex-4,Virtex-II, Virtex-II Pro, and Spartan-3 all use 18 Kb Select RAM-II blocks, any specific reference to a Virtex-II implementation also applies to a Virtex-II Pro, Virtex-4, or Spartan-3 implementation.

All memory operations occur on the active edge of the clock input (CLK), When Block Memory is enabled. The Block Memory can be configured to be active on the rising edge and the falling edge. The memory configuration and output value remain unaltered , when the block memory is disabled (enable inactive).

The data presented at the port’s data input is stored in memory at the location selected by the port’s address input, during a write operation (WE asserted). During this operation, the data output port behaves differently for the Virtex-II and Virtex architectures.

The Virtex implementation supports a single write mode option, Read-After-Write. This write mode causes the data being written to the addressed memory location to be transferred to the data output port when a write operation occurs.

Three write mode options to determine the behavior of the data output port (read port) during a write operation supported by The Virtex-II implementation:

- Read-After-Write (Write First)
- Read-Before-Write (Read First)

• No-Read-On-Write (No Change)

The memory contents at the location selected by the address will appear at the module's output, during a read operation. the module's registered outputs are synchronously reset to zero for Virtex and to a user-defined value for Virtex-II,when Synchronous Initialization (SINIT) is active,.

The contents of the memory or write operations does not effected by the Synchronous Initialization command. The initial contents of the memory (that is, the data stored in the memory immediately after device configuration) can also be specified.

The enable, write enable, and synchronous initialization control signals can also be specified as active high or active low.

4.1 General Block diagram : Single Port RAM:

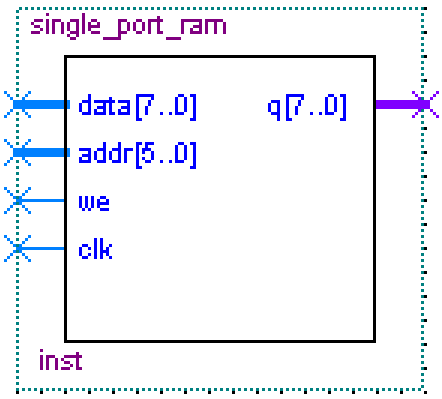

Fig 4.1: General Block Diagram of Single Port RAM

Signal	Direction	Description
DIN[n:0] (optional)	Input	Data input : data written into memory.
ADDR [m:0]	Input	Address : memory location for data written to/read from.
WE [optional]	Input	Write enable : allows data transfer into memory .
EN [optional]	Input	Enable : enables access to memory via read and write operations .

SINIT [optional]	Input	Synchronous imitialization : forces module outputs to a predefined state .
CLK	Input	Clock : all memory operations synchronous with rising or falling edge of clock input ,depending on user configuration of the clock pin polarity.
ND [optional]	Input	New data : indicates new and valid address on ADDR(active high).
DOUT[n:0]	Output	Data output : synchronous output of the memory.
RFD [optional]	Output	Ready for data : indicates that memory is ready for new address.
RDY [optional]	Output	Output ready : indicates valid data on DOUT port (active high).

Table 4.1: Specifications of in/out pins

BLOCK DIAGRAM: Output RTL Design

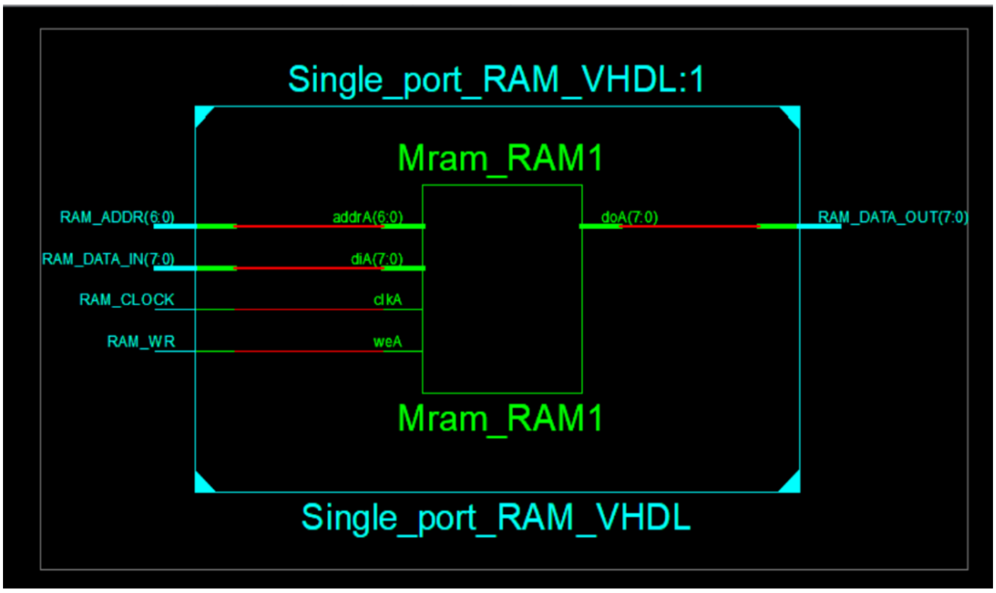

Fig 4.2: Output RTL Design of Single Port RAM

4.2 BLOCK DESCRIPTION:

Pinout

Port names for the core module is displayed in Figure and described in the above Table . Excluding these ports will alter the function of the module; the inclusion of some ports on the module is optional. The optional ports are designated in the above Table.

Clock - CLK

Block Memory is full synchronous with the input of the the clock. All input pins have setup time referenced to the port CLK pin. The DOUT port has a clocktoout time referenced to the CLK pin.

By default all memory operations are performed at the clock's rising edge . However the users , have the option to perform all operations related to memoryat the rising or the falling edge of the input clock. Performing the memory operation on the falling edge of the clock will not use any extra resources.

Enable - EN

The read, write, and SINIT performance of the port is changed by the enable pin. When the Block Memory has an inactive enable pin, the output pins are held in the previous state and writing to the memory is disabled.

By default the enable pin is active high. However,The users have the option to configure the enable pin active high or active low. Configuring the enable pin active low will not use additional resources.

Write Enable - WE

Activating the write enable pin enables writing to the memory locations. When it is at active, the contents of the DIN bus is written to memory at the address pointed to by the ADDR bus. The output latches are loaded or not loaded according to the write configuration (Write First, Read First, No Change).

When WE is at inactive, a read operation is made, and the contents of the memory addressed by the ADDR bus are driven on the DOUT bus. In the Read Only port configuration (ROM configuration), the WE pin is not present.

By means of default, the write enable pin is active high. However, The user have the option to modify the write enable pin active high or active low. Modifying ithe write enable pin active low will not use extra resources.

Synchronous Initialization - SINIT

When it is at enable, the SINIT pin forcefully push the data output latches to synchronously load the predefined SINIT value.

For the Virtex implementation, the SINIT value is zero. So only , asserting the SINIT pin make the output latches to get reset.

For the Virtex-II implementation, the SINIT value as to be defined by the user. So Similarly, asserting the SINIT pin causes the output latches to contain the user-defined SINIT value. This operation do not affect the memory locations and do not disturb the write operations. Like the read and write operation, the SINIT function is active only when the enable pin of the port is active.

By default, the SINIT pin is active high. Users, however, have the option to modify the SINIT pin active high or active low. Configuring the write enable pin active low will not use extra resources.

Address Bus - ADDR[m:0]

The memory location for read or write access is selected by the address bus.

Data-In Bus - DIN[n:0]

The DIN bus produces the data value that has to be written into the memory. Data input and output signals are always buses; that is, in a 1-bit width configuration, the data input signal is DIN[0] and the data output signal is DOUT[0]. The DIN bus is not available in the Read Only port configuration (ROM configuration).

Data-Out Bus - DOUT[n:0]

The DOUT bus is the reflection of the contents of memory locations recommended by the address bus duringa read operation.

During a write operation of a Virtex memory (Write First configuration), the DOUT bus reflects the data that is written on the DIN bus.

During a write operation of a Virtex-II or Spartan-3 memory (Write First or Read First Configuration), the data-out bus reflects either the DIN bus (Write First) or the current memory contents, previously stored value (Read First). During a write operation in No Change mode, the data-out bus is not affected.

New Data - ND

New Data ND is the Indication that there is a new and viable address on ADDR Port.

Ready for Data - RFD

Ready for Data RFD is the indication that the memory is ready to accept a new address. RFD is always true, but when EN isinactive, it is false.

Output Ready (Valid) - RDY

Output Ready RDY is the indication that the valid output on the DOUT port. RDY will lag ND by the dormancy of the block memory.

The list of VHDL templates will be listed and described below:

- Single-Port RAM with asynchronous read
- Single-Port RAM with "false" synchronous read
- Single-Port RAM with synchronous read (Read Through)
- Single-Port RAM with Enable

4.3 SinglePort RAM with Asynchronous Read

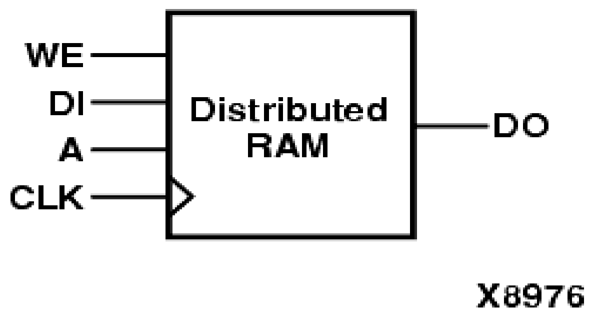

Fig 4.3: Single Port RAM with Asynchronous Read

The descriptions are directly mappable onto distributed RAM only by following.

The following below table display the pin descriptions for a single port RAM with asynchronous read.

IO Pins	Description
clk	Positive-edge clock
we	Synchronous write enable(active high)
a	Read/write address
di	Data input
do	Data output

Table 4.2: Asynchronous read

VHDL

The Following below is the VHDL code of an singleport RAM with asynchronous read.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity raminfr is
port (clk : in std_logic;
we  : in std_logic;
a   : in std_logic_vector(4 downto 0);
di  : in std_logic_vector(3 downto 0);
do  : out std_logic_vector(3 downto 0));
end raminfr;
architecture syn of raminfr is
type ram_type is array (31 downto 0)
of std_logic_vector (3 downto 0);
signal RAM : ram_type;
begin
process (clk)
begin
if (clk'event and clk = '1') then
if (we = '1') then
RAM(conv_integer(a)) <= di;
end if;
end if;
end process;
do <= RAM(conv_integer(a));
end syn;
```

4.4 SinglePort RAM with "false" Synchronous Read

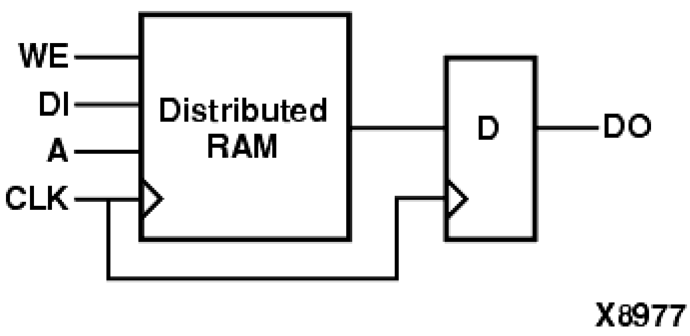

Fig 4.4: Single Port RAM with False synchronous Read

The following below descriptions do not implement true synchronous read access as defined by the virtex block RAM specification ,where the read address is registered . They are only mappable onto distributed RAM with an additional buffer on the data output ,as shown below:

The following below table display pin descriptions of an single port RAM with “false” synchronous read.

IO Pins	Description
clk	Positive-edge clock
we	Synchronous write enable(active high)
a	Read/write address
di	Data input
do	Data output

Table 4.3: False synchronous read

VHDL

The Following below is the VHDL code of an singleport RAM with "false" synchronous read.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity raminfr is
port (clk : in std_logic;
      we  : in std_logic;
      a   : in std_logic_vector(4 downto 0);
      di  : in std_logic_vector(3 downto 0);
      do  : out std_logic_vector(3 downto 0));
end raminfr;

architecture syn of raminfr is
type ram_type is array (31 downto 0)
of std_logic_vector (3 downto 0);
```

```
signal RAM : ram_type;
begin
process (clk)
begin
if (clk'event and clk = '1') then
if (we = '1') then
RAM(conv_integer(a)) <= di;
end if;
do <= RAM(conv_integer(a));
end if;
end process;
end syn;
```

The following below descriptions, featuring an additional reset of the RAM output , are also only mappable onto distributed RAM with an additional resetable buffer on the data output is as shown in the following below figure:

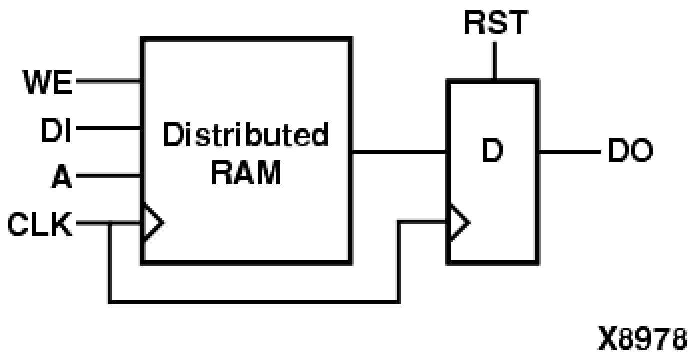

Fig 4.5: Single Port RAM with False synchronous Read and reset on output

The following below table display pin descriptions of an singleport RAM with "false" synchronous read and reset on output.

IO Pins	Description
Clk	Positive-edge clock
we	Synchronous write enable(active high)
a	Read/write address
rst	Synchronous output reset(active high)
di	Data input

do	Data output

Table 4.4: False synchronous read and reset on output

VHDL

The Following below is the VHDL code for given.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity raminfr is
port (clk : in std_logic;
      we  : in std_logic;
      rst : in std_logic;
      a   : in std_logic_vector(4 downto 0);
      di  : in std_logic_vector(3 downto 0);
      do  : out std_logic_vector(3 downto 0));
end raminfr;

architecture syn of raminfr is
type ram_type is array (31 downto 0)
of std_logic_vector (3 downto 0);
signal RAM : ram_type;
begin
process (clk)
begin
if (clk'event and clk = '1') then
if (we = '1') then
RAM(conv_integer(a)) <= di;
end if;
if (rst = '1') then
do <= (others => '0');
else
do <= RAM(conv_integer(a));
end if;
end if;
end process;
end syn;
```

4.5 SinglePort RAM with Synchronous Read (Read Through)

The following below description implements a true synchronous read. A true synchronous read is the synchronization mechanism available in Virtex block RAMs, where the read address is registered on the RAM clock edge. Such descriptions are directly mapable into BLOCK RAM as shown below in fig.

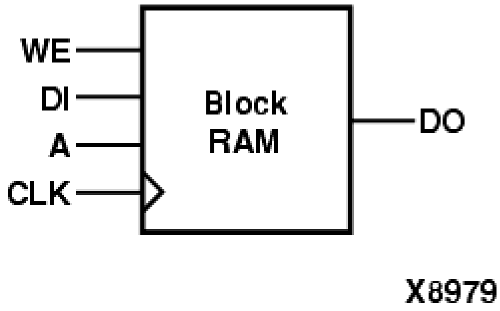

Fig 4.6: Single Port RAM with Synchronous Read (Read Through)

The following below table display pin descriptions of an singleport RAM with synchronous read

(read through)

IO Pins	Description
clk	Positive-edge clock
we	Synchronous write enable(active high)
a	Read/write address
di	Data input
do	Data output

Table 4.5: Synchronous read (read through)

VHDL

The Following below is the VHDL code of an singleport RAM with synchronous read (read through).

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity raminfr is
port (clk : in std_logic;
we  : in std_logic;
a   : in std_logic_vector(4 downto 0);
di  : in std_logic_vector(3 downto 0);
do  : out std_logic_vector(3 downto 0));
end raminfr;
architecture syn of raminfr is
type ram_type is array (31 downto 0)
of std_logic_vector (3 downto 0);
signal RAM : ram_type;
signal read_a : std_logic_vector(4 downto 0);
begin
process (clk)
begin
if (clk'event and clk = '1') then
if (we = '1') then
RAM(conv_integer(a)) <= di;
end if;
read_a <= a;
end if;
end process;
do <= RAM(conv_integer(read_a));
end syn;
```

4.6 Single-Port RAM with Enable

The following below is the description of implementing a singleport RAM with a global enable.

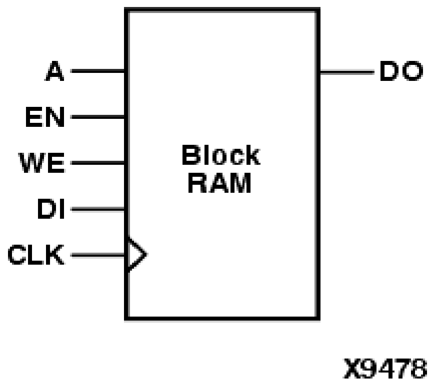

Fig 4.7: Single Port RAM with Enable

The following below table display the pin descriptions of an singleport RAM with enable.

IO Pins	Description
clk	Positive-edge clock
we	Synchronous write enable(active high)
a	Read/write address
di	Data input
do	Data output
en	Global enable

Table 4.6: single port ram with enable

VHDL

The Following below is the VHDL code of an singleport block RAM with enable.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity raminfr is
port (clk : in std_logic;
      en    : in std_logic;
      we  : in std_logic;
      a   : in std_logic_vector(4 downto 0);
```

```
        di  : in std_logic_vector(3 downto 0);
        do  : out std_logic_vector(3 downto 0));
end raminfr;

architecture syn of raminfr is
type ram_type is array (31 downto 0)
of std_logic_vector (3 downto 0);
signal RAM : ram_type;
signal read_a : std_logic_vector(4 downto 0);
begin
process (clk)
begin
if (clk'event and clk = '1') then
if (en = '1') then
if (we = '1') then
RAM(conv_integer(a)) <= di;
end if;
read_a <= a;
end if;
end if;
end process;
do <= RAM(conv_integer(read_a));
end syn;
```

Flow Chart of Project:

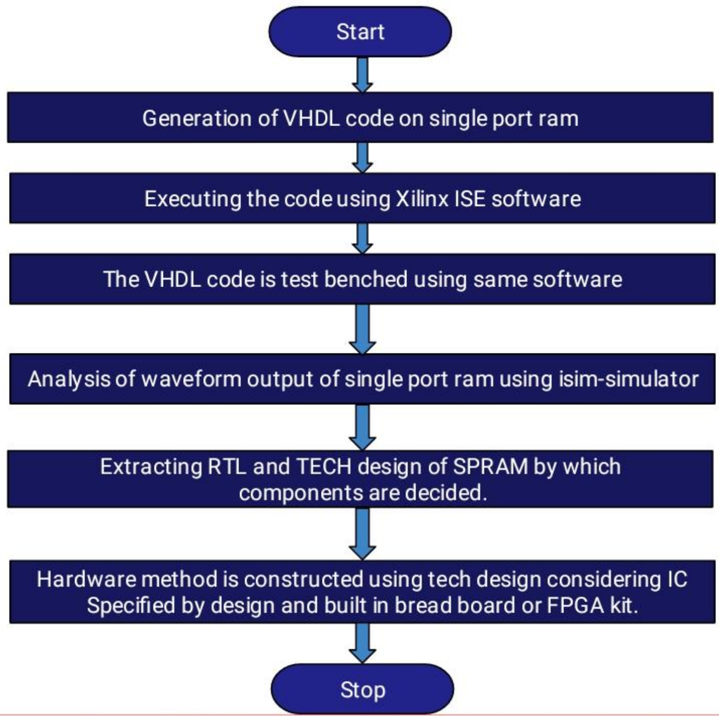

Fig 4.8 : Flow chart

CHAPTER 05

SOFTWARE SPECIFICATION

Software Tool used: Xilinx ISE Design Suite 14.7

The software tool used in this project is Xilinx ISE Design suite version 14.7;

ISE stands for (Integrated Synthesis Environment).

This is the software made by Xilinx to analyze HDL designs and its synthesis , and even to compile the design ,examine RTL Design.

Developer: Xilinx

Release: October 23 2013

Advantages of Xilinx ISE Design Suite:

- The software make user work easy.
- Xilinx software used to reduce components that indulge in design.
- Make the usage of hardware components less and make it efficient.
- Reduce the physical error such as connection and interconnection between parts.
- Reduce the electronic waste by rectifying the software output I.e., encouraging pollution free.

Disadvantages of Xilinx ISE Design Suite:

- Unexpected crash while Synthesis of code.
- The iSim Simulator takes more time to display the output waveform.
- Usage of Vivado Suite is superior and better when compared to Xilinx ISE in terms of performance

CHAPTER 06

RESULT AND DISCUSSION

RESULT:

- The RTL and TECH design of single port RAM is implemented using xilinx ISE software.
- Thus the function and working is analyzed by the output waveform and design.
- The project defines the functionality and construction of Single Port RAM.
- Thus we have learn about how the Single port RAM works and with RTL and Tech

design by HDL programming using xilinx software; We learn in depth working

Of Single Port RAM.

OUTPUT WAVEFORM :

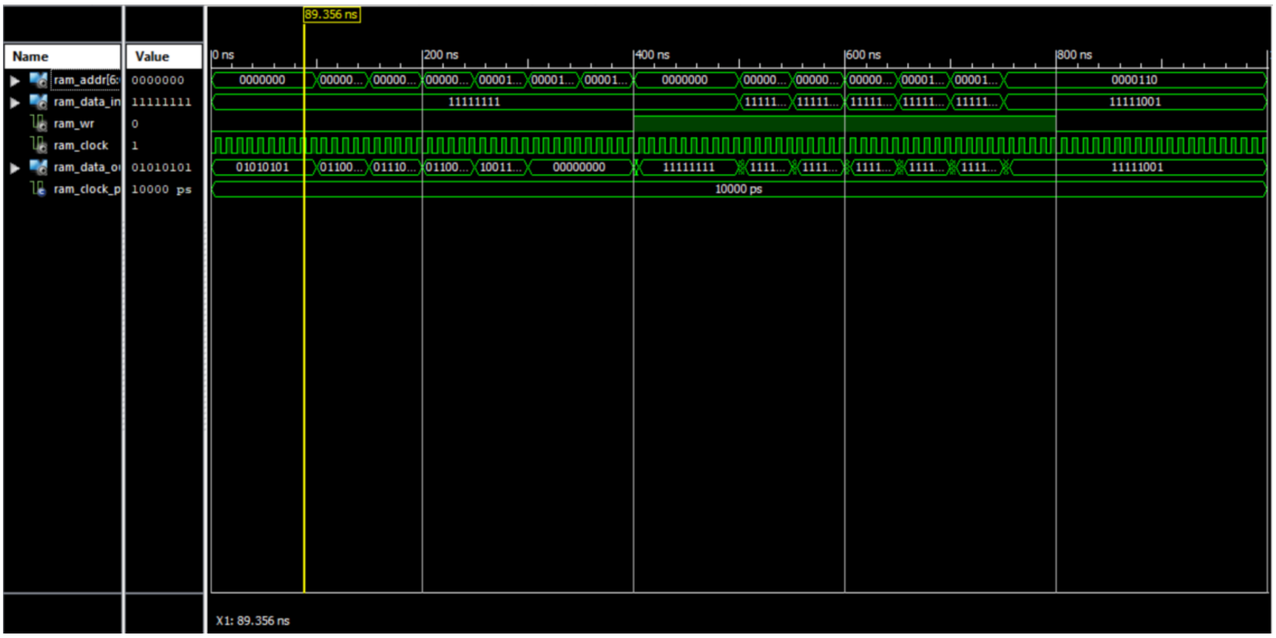

Fig 6.1: Output Waveform

OUTPUT TECH DESIGN :

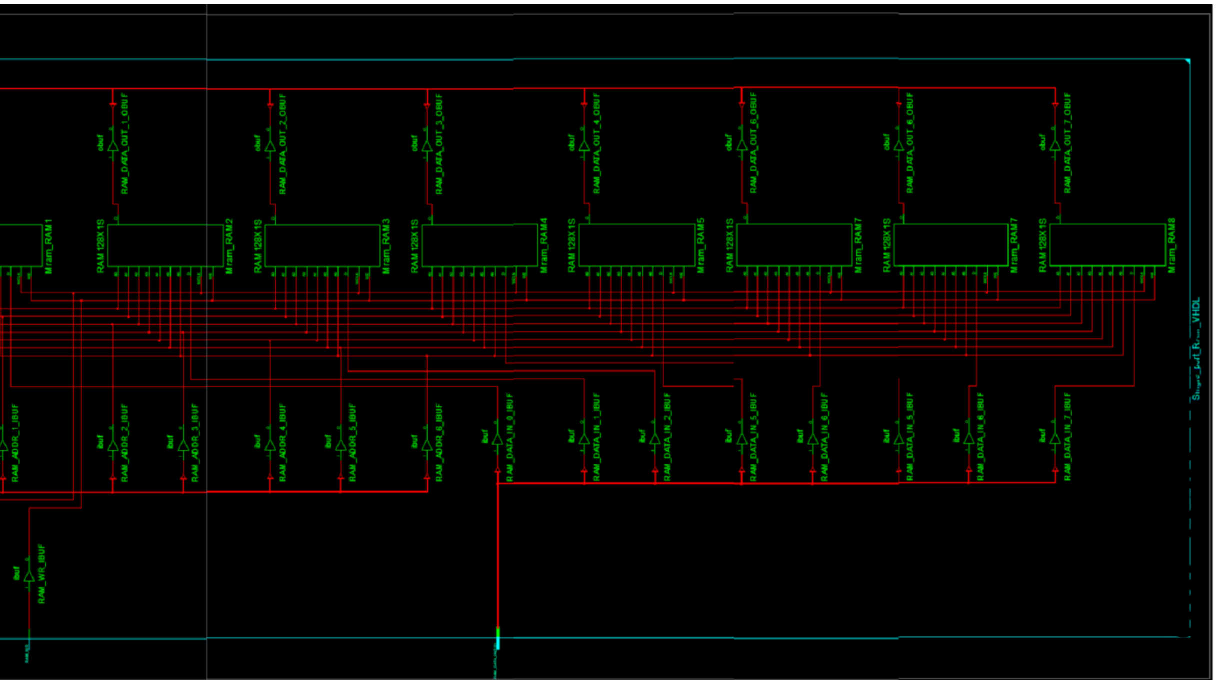

Fig 6.2: Output Tech Design

DISCUSSION :

- Even though single port ram is old but the read and write speed is much speed when

Compared than sync RAM i.e., dual port ram or any DRAM.

- Since our project only concerned about the working of single port ram , So understanding the RAM can make the future RAM still more speed and better.

- So the better version of single port ram can be constructed by using the technique used and more functions can be added to make it more efficient.

CHAPTER 07

ADVANTAGES AND APPLICATION

Advantages of the project:

- It can access the memory faster than ever.
- The size is small and easy to place .
- Operations are made fast and simple.
- Although a dual-port memory is ideal, it is not always available. Using regular single-port memory, such as the S3 board's external SRAM, for the video memory requires careful coordination between the write and read operations to avoid interruption in data retrieval.
- The main benefit of this feature is performance, since single port RAM allows only one memory cell to be read/written during each clock cycle, whereas dual-port memory allows for two memory cell accesses per cycle.
- Since SPRAM acts only as RAM hence the delay to output is very less, Whereas dual port is used as sync that delay output.

Application of the Project:

- It is used to store the data file or operating system.
- Since the speed of single port RAM is fast , It can be used to make RAM more effective in data transmission without any restrictions.
- Used for micro controller and microprocessors.

CHAPTER 08

CONCLUSION AND FUTURE SCOPE

CONCLUSION :

- Single port ram design had been constructed using Xilinx ISE design suite software.
- The software part of our project is understood by RTL and TECH design.
- The Single Port RAM is yet to be constructed in bread board as rather than using PCB making it different.
- But our project can be well developed using FPGA kit spartan 6
  Hence yet to complete on kit.
- Working of single port ram is thoroughly understood and demonstrated.
- The fundamental components of Single port ram understood and how the FPGA kit works withVHDL programming.

FUTURE SCOPE :

- In future, this may help us develop a bigger size of ram or DDR ram in our future mini project with more features.
- To create speed RAM with less data interpretation in data retrieval.
- The main purpose of our project is to understand about working of single port

  RAM and to study the speed, Hence this could lead to make better version.
- Since single port RAM is know for it's speed an bigger version can be created similar to it.

REFERENCES

[1] Pong P. Chu ," **FPGA Prototyping By Verilog Examples**", A John Wiley & Sons, INC., Publication, n9781118210611,2011.

[2] Pong p. Chu ," **FPGA Prototyping By VHDL Examples**",A John Wiley & Sons, INC., Publication, 9781119282754, 2017.

[3] William J. Dally, R. Curtis Harting" **Digital Design using VHDL",** Cambridge University Press , 9781107098862, 2016.

APPENDIX

PROJECT CODE: SINGLE PORT RAM USING VHDL CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.ALL;

entity Single_port_RAM_VHDL is-- A 128x8 single-port RAM in VHDL
port(
RAM_ADDR: in std_logic_vector(6 downto 0);-- Address to write/read RAM
RAM_DATA_IN: in std_logic_vector(7 downto 0);-- Data to write into RAM
RAM_WR: in std_logic; -- Write enable
RAM_CLOCK: in std_logic;-- clock input for RAM
RAM_DATA_OUT: out std_logic_vector(7 downto 0)-- Data output of RAM
);
end Single_port_RAM_VHDL;

architecture Behavioral of Single_port_RAM_VHDL is
-- define the new type for the 128x8 RAM
type RAM_ARRAY is array (0 to 127 ) of std_logic_vector (7 downto 0);
-- initial values in the RAM
signal RAM: RAM_ARRAY :=(
  x"55",x"66",x"77",x"67",-- 0x00:
  x"99",x"00",x"00",x"11",-- 0x04:
  x"00",x"00",x"00",x"00",-- 0x08:
  x"00",x"00",x"00",x"00",-- 0x0C:
  x"00",x"00",x"00",x"00",-- 0x10:
  x"00",x"00",x"00",x"00",-- 0x14:
  x"00",x"00",x"00",x"00",-- 0x18:
  x"00",x"00",x"00",x"00",-- 0x1C:
  x"00",x"00",x"00",x"00",-- 0x20:
  x"00",x"00",x"00",x"00",-- 0x24:
  x"00",x"00",x"00",x"00",-- 0x28:
  x"00",x"00",x"00",x"00",-- 0x2C:
  x"00",x"00",x"00",x"00",-- 0x30:
  x"00",x"00",x"00",x"00",-- 0x34:
  x"00",x"00",x"00",x"00",-- 0x38:
  x"00",x"00",x"00",x"00",-- 0x3C:
  x"00",x"00",x"00",x"00",-- 0x40:
  x"00",x"00",x"00",x"00",-- 0x44:
```

```
  x"00",x"00",x"00",x"00",-- 0x48:
  x"00",x"00",x"00",x"00",-- 0x4C:
  x"00",x"00",x"00",x"00",-- 0x50:
  x"00",x"00",x"00",x"00",-- 0x54:
  x"00",x"00",x"00",x"00",-- 0x58:
  x"00",x"00",x"00",x"00",-- 0x5C:
  x"00",x"00",x"00",x"00",
  x"00",x"00",x"00",x"00",
  x"00",x"00",x"00",x"00",
  x"00",x"00",x"00",x"00",
  x"00",x"00",x"00",x"00",
  x"00",x"00",x"00",x"00",
  x"00",x"00",x"00",x"00",
  x"00",x"00",x"00",x"00"
  );
begin
process(RAM_CLOCK)
begin
if(rising_edge(RAM_CLOCK)) then
if(RAM_WR='1') then-- when write enable = 1,
-- write input data into RAM at the provided address
RAM(to_integer(unsigned(RAM_ADDR))) <= RAM_DATA_IN;
-- The index of the RAM array type needs to be integer so
-- converts RAM_ADDR from std_logic_vector -> Unsigned -> Interger using numeric_std
library
end if;
end if;
end process;
-- Data to be read out
RAM_DATA_OUT <= RAM(to_integer(unsigned(RAM_ADDR)));
end Behavioral;
```

TEST BENCH CODE:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;
-- VHDL testbench code for the single-port RAM
ENTITY tb_RAM_VHDL IS
END tb_RAM_VHDL;

ARCHITECTURE behavior OF tb_RAM_VHDL IS
```

```
-- Component Declaration for the single-port RAM in VHDL

COMPONENT Single_port_RAM_VHDL
   PORT(
       RAM_ADDR : IN  std_logic_vector(6 downto 0);
       RAM_DATA_IN : IN  std_logic_vector(7 downto 0);
       RAM_WR : IN  std_logic;
       RAM_CLOCK : IN  std_logic;
       RAM_DATA_OUT : OUT  std_logic_vector(7 downto 0)
      );
   END COMPONENT;

  --Inputs
  signal RAM_ADDR : std_logic_vector(6 downto 0) := (others => '0');
  signal RAM_DATA_IN : std_logic_vector(7 downto 0) := (others => '0');
  signal RAM_WR : std_logic := '0';
  signal RAM_CLOCK : std_logic := '0';

 --Outputs
  signal RAM_DATA_OUT : std_logic_vector(7 downto 0);

  -- Clock period definitions
  constant RAM_CLOCK_period : time := 10 ns;

BEGIN
-- Instantiate the single-port RAM in VHDL
  uut: Single_port_RAM_VHDL PORT MAP (
        RAM_ADDR => RAM_ADDR,
        RAM_DATA_IN => RAM_DATA_IN,
        RAM_WR => RAM_WR,
        RAM_CLOCK => RAM_CLOCK,
        RAM_DATA_OUT => RAM_DATA_OUT
      );

  -- Clock process definitions
  RAM_CLOCK_process :process
  begin
 RAM_CLOCK <= '0';
 wait for RAM_CLOCK_period/2;
 RAM_CLOCK <= '1';
 wait for RAM_CLOCK_period/2;
  end process;

  stim_proc: process
  begin
 RAM_WR <= '0';
 RAM_ADDR <= "0000000";
```

```
 RAM_DATA_IN <= x"FF";
  wait for 100 ns;
 -- start reading data from RAM
 for i in 0 to 5 loop
 RAM_ADDR <= RAM_ADDR + "0000001";
 wait for RAM_CLOCK_period*5;
 end loop;
 RAM_ADDR <= "0000000";
 RAM_WR <= '1';
 -- start writing to RAM
 wait for 100 ns;
 for i in 0 to 5 loop
 RAM_ADDR <= RAM_ADDR + "0000001";
 RAM_DATA_IN <= RAM_DATA_IN-x"01";
 wait for RAM_CLOCK_period*5;
 end loop;
 RAM_WR <= '0';
 wait;
end process;
END;
```

Spartan 6 FPGA Data Sheet

Spartan 6 LXT and LX FPGAs are accessible with different speed grades, with - 3 having the best in performance. The DC and AC electrical parameters of the Automotive XA Spartan6 FPGAs and Defensegrade Spartan6Q FPGAs gadgets are proportional and equal to the commercial specs with the exception of where noted. The timing characteristics of the commercial (XC) - 2 speed grade modern gadget are equivalent to for a - 2 speed grade commercial gadget. The - 2Q and - 3Q speed grades are only for the extended (Q) temperature range. The planning qualities are proportionate to those appeared for the - 2 and - 3 speed grades for the Automotive and Defensegrade gadgets.

Spartan 6 FPGA DC and AC qualities are determined for commercial (C), industry (I), and extended (Q) temperature ranges. Just chose speed grades as well as gadgets may be accessible in the industry or extended temperature ranges for automatic and Defensegrade gadgets.

References to gadget names allude to every single accessible variety of that part number (for model, LX75 could signify XC6SLX75, XA6SLX75, or XQ6SLX75). The Spartan6 FPGA - 3N speed grade assigns gadgets that don't support MCB function.

All supplied voltage and temperature at junctionspecs are illustrative of most pessimistic scenario conditions. The parameters included are normal to popular designs and run of the classical applications.

More information on Spartan 6 FPGA Data Sheet

https://www.xilinx.com › data...PDF
Web results
Spartan-6 FPGA Data Sheet: DC and Switching Characteristics (DS162)