International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com
Volume 6, Issue 1 (February 2013), PP. 15-20

Design and Implementation of Memory Interface Controller with SRAM Memory

Dr. R. D. Kanphade[1], S. B. Patil[2], Rohan Musale[3]
[1]Principal, Nutan Maharastra Institute of Engg.& Technology, Vishnupuri, Talegaon Dabhade - 410507 Tal. Maval, Dist. Pune (Maharastra)
[2,3]Department of Electronics & Telecommunication Engineering, S.S.G.M.College of Engineering, Shegaon, Dist-Buldana, Shegaon-444203

Abstract:- Any digital system requires memory to store the data and a controller to operate it. Generally ardware is used for computation with the memory. But these computations are permanently frozen by manufacturing process. So this kind of system can't provide flexibility to change the design. The proposed model is an integrated memory controller with its SRAM; which can be used for any real time application. A Memory controller and SRAM of size (256×8) is designed with VHDL coding. It is a prototype model using Xilinx's Spartan FPGA .FPGA gives nearly all benefits of software flexibility and development model. This FPGA- based system can be reprogrammed many times or even new task can be performed.

Keywords:- Memory controller, SRAM, FPGA, Xilinx's Spartan.

I. INTRODUCTION

A digital processing system invariably requires a facility for storing digital information. The information usually consists of instructions (processing steps) coded in binary form, data to be processed, intermediate and final results, etc. The subsystem of a digital processing system, which provides the storage facility, is referred to as the memory. With the development in semiconductor technology, it has become possible to make semiconductor memories of various types and sizes. Therefore it is necessary for a designer of digital processors to know thoroughly the principles of operation and limitations of various semiconductor memory devices.

In the computer and electronics world, we are using two different ways of performing computation: hardware and software. Computer hardware, such as application-specific integrated circuits (ASICs), provides highly optimized resources for quickly performing critical tasks, but it is permanently configured to only one application via a multimillion-dollar design and fabrication effort. Computer software provides the flexibility to change applications and perform a huge number of different tasks, but is orders of magnitude worse than ASIC implementations in terms of performance, silicon area efficiency, and power usage. Field-programmable gate arrays (FPGAs) are truly revolutionary devices that blend the benefits of both hardware and software. So I decided to design An integrated SRAM with its memory controller which is used for any real-time applications.

II. OVERVIEW OF SRAM

An SRAM (Static Random Access Memory) is designed to fill two needs: to provide a direct interface with the CPU at speeds not attainable by DRAMs and to replace DRAMs in systems that require very low power consumption. The SRAM cell consists of a bi-stable flip-flop connected to the internal circuitry by two access transistors shown in Figure 1. When the cell is not addressed, the two access transistors are closed and the data is kept to a stable state, latched within the flip-flop.

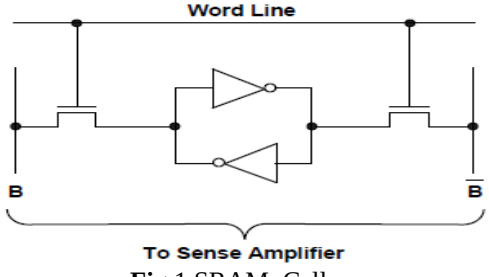

Fig 1 SRAM Cell.

However, the data does not "leak away" like in a DRAM, so the SRAM does not require a refresh cycle. One of the main reasons causing the leakage power increase is the increase of sub-threshold leakage power. Advanced technology requires scaling of supply voltage and threshold voltage. Sub-threshold leakage power increases exponentially as threshold voltage decreases. This project focuses on reducing sub-threshold leakage power consumption. Different types of SRAM cells are based on the type of load used in the elementary inverter of the flip-flop cell. There are currently three types of SRAM memory cells:

- The 4T cell (four NMOS transistors plus two poly load resistors)
- The 6T cell (six transistors—four NMOS transistors plus two PMOS transistors)

The TFT cell (four NMOS transistors plus two loads called TFTs.

Basic Architecture of SRAM

The basic architecture of a static RAM includes one or more rectangular arrays of memory cells with support circuitry to decode addresses, and implement the required read and write operations. Additional support circuitry used to implement special features, such as burst operation, may also be present on the chip. Figure 2 shows a basic block diagram of a synchronous SRAM. As you read, you may wish to refer to the diagram to help you visualize how the SRAM works.

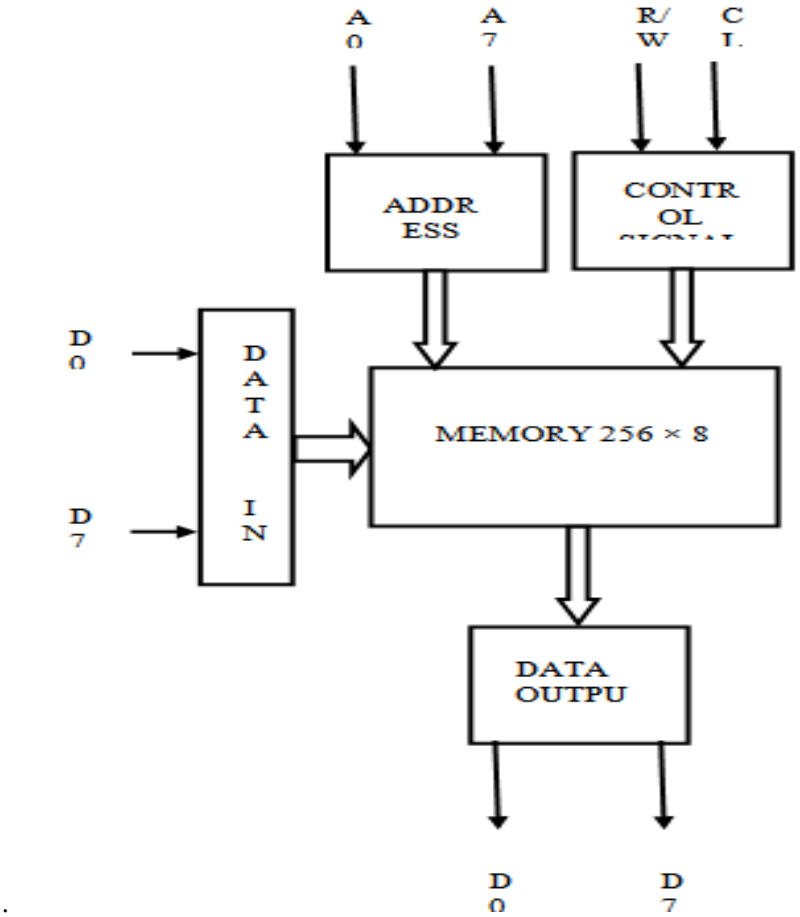

Fig 2 Basic structure of SRAM

Figure 2 shows the block diagram of memory (256×8). This has 8 bit address bus A0 to A7. The address inputs are used to select a memory location on the chip. In actuality, when you select an address, you choose a memory location. For performance reasons, there are row and column address pins so that both may be selected at once. The number of address input pins depends on the size of the memory and how it is organized. For example, a 256 by 8 SRAM has 8 address input pins and 8 bit data bus from D0 to D7. This memory has got control signals R/W and Clock from controller.

System Block diagram System

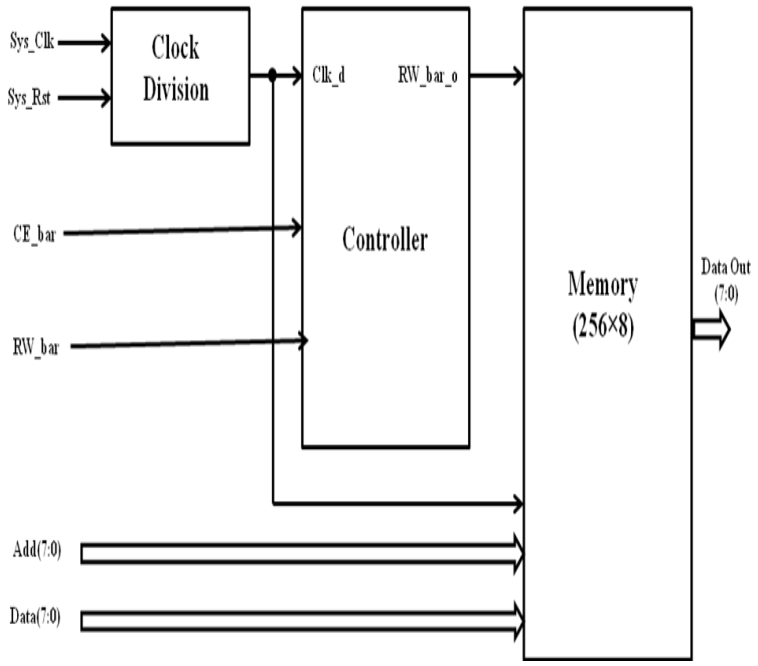

Fig 3 System block diagram

Above figure shows the block diagram of whole system. It consist of three blocks namely clock division unit, controller unit and memory unit. First block is clock division unit. It has two inputs i.e. system reset and system clock. Purpose of this block is to divide the system clock. This divided clock is input clock for controller unit. Controller generates the controlling action like read or writes for the memory. And the last block is memory which has 8 bit address as well as data bus. The detail description of these blocks is given below.

III. IMPLEMENTATION OF CONTROLLER WITH SRAM

For implementation of the system master clock is divided because it very difficult to observe the output on LED's connected to the board at very high frequency. Following are important steps for implementation of whole system

1. Division of clock

Initially system clock is divided by clock divider unit. Clk_d is output of clock divider unit. If system reset is logic 0 then clk_d signal is 0.

$$Clk\_d = \frac{Systemclock}{Count}$$

$$= \frac{12MHz}{12\times10^{6}}$$

$$=1\ Hz$$

2. Controller

When Clk_d' event and Clk_d='1' and CE_bar ='0' then RW_bar is assigned to RW_bar_o.

3. Memory

When Clk_d' event and Clk_d='1' and RW_bar ='0' then controller will perform write operation. Otherwise if RW_bar ='1' then controller will perform read operation.

It has System clock and Reset as input and output is divided clock. Clock divider circuit is used to divide system clock frequency which is very high.

3. Flow chart of system

The Figure 4 shows the flowchart for system operation. First off all system clock and system reset are applied to clock divider unit. If system reset is logic '0' then memory gets reset. When system reset is logic '0', system clock is divided by the count variable.

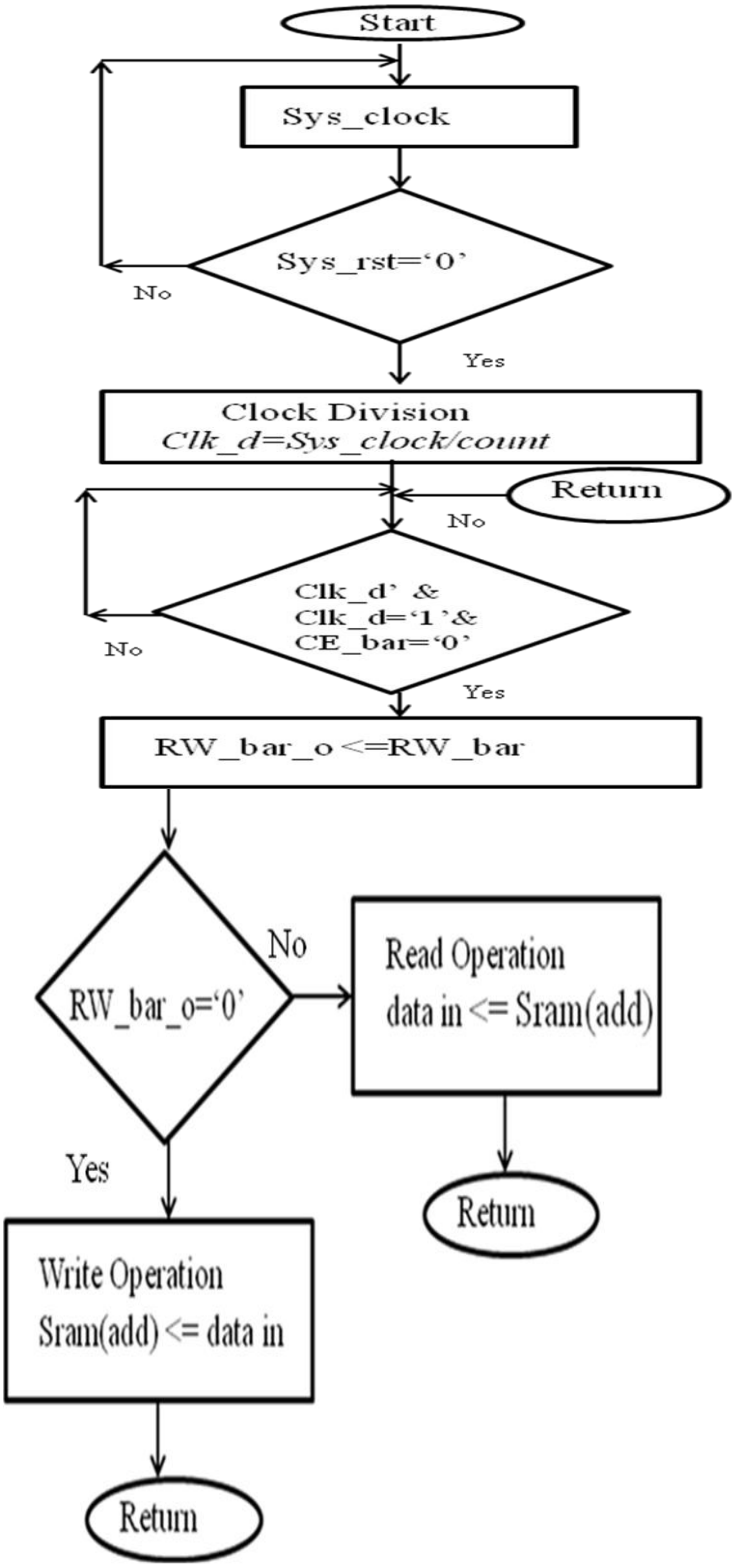

Fig 4. Flow chart of system

This divided system clock i.e. clk_d is applied as clock signal to the controller; which is used to interface with the memory. Controller gives efficient access to memory and generates controller signal RW_bar_o. When divided clock i.e. clk_d' event and clk_d is equal to '1' and CE_bar is logic '0' input signal RW_bar is assigned to RW_bar_o. this output signal from controller decides whether write or read operation is to be done. But if CE_bar is not logic '0' RW_bar will not be activated.

Write operation : When divided clock i.e. clk_d' event and clk_d is equal to '1' and CE_bar is logic '0' input signal RW_bar is logic '0' controller asserts RW_bar_o signal which initiates write operation.
Here data present on data bus is written on the memory location available at that time.
Read operation: Write operation : When divided clock i.e. clk_d' event and clk_d is equal to '1' and CE_bar is logic '0' input signal RW_bar is logic '1' controller asserts RW_bar_o signal which initiates read operation

Integrated module

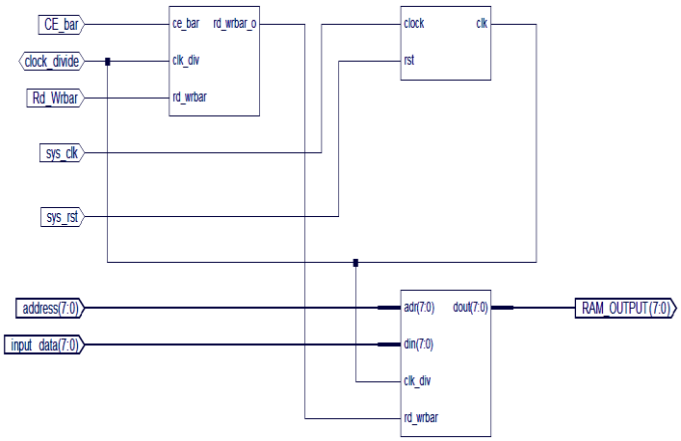

Fig 5 Integrated model

5. Result

Controller output

The following Figur 6 shows the simulation result of controller unit. When divided clock i.e. clk_d' event and clk_d is equal to '1' and CE_bar is logic '0' input signal RW_bar is assigned to RW_bar_o.

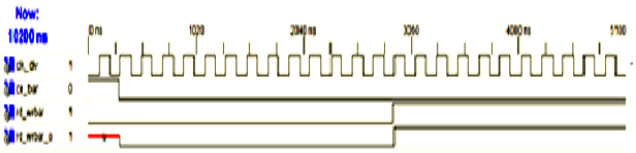

Fig 6 Controller output

Place and Route Report

Device Utilization Summary:

Number of BUFGMUXs	1 out of 8	12%
Number of External IOBs	29 out of 141	20%
Number of LOCed IOBs	0 out of 29	0%
Number of RAMB16s	1 out of 16	6%
Number of Slices	53 out of 3584	1%
Number of S LICEMs	0 out of 1792	0%

Fig 7 Device utilization summery

IV. CONCLUSION

Integrated approach SRAM and its controller is a general purpose system could be used in any real-time application. FPGA is the best platform to design and test digital system before go for ASIC. We can design and redesign (in case of dissatisfaction) until we reach to the optimum solution, all it costs a time and nothing else.

Future Work

In this design single SRAM is implemented with the help of controller. Further SRAM bank can be implemented with controller to select one of SRAM chip from the bank. For this one additional signal 'cs_bar' can be used. Thus number of SRAM bank can be controlled with controller.

REFERENCES

[1]. Mahendra Kumar, Kailash Chandra,"Low Power High Performance SRAM Design Using VHDL '' *Global journal of researches in Engineering. Volume 11 Issue 1 Version 1.0 February 2011.*

[2]. Baosheng Wang, Yuejian Wu and André Ivanov, "A Fast Diagnosis Scheme for Distributed Small Embedded SRAMs'', *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05)1530-1591/05 $ 20.00 IEEE*

[3]. Naagesh. S. Bhat , " Design and modelling of different sram's Based on cntfet 32nm technology", *International Journal of VLSI design & Communication Systems (VLSICS) Vol.3, No.1, February 2012*.

[4]. Mr. Sunil Jadav, Mr. Vikrant and Dr. Munish Vashisath, "Design and performance analysis of ultra low power 6t sram using adiabatic technique", *International Journal of VLSI design & Communication Systems (VLSICS) Vol.3, No.3, June 2012*

[5]. Joonseok Park and Pedro C. Diniz, "Synthesis and Estimation of Memory Interfaces for FPGA-based Reconfigurable Computing Engines", *FCCM'03, April 08-11, 2003, Napa, California, USA*.

[6]. http://www.engr.uconn.edu/~omer.khan/courses/ece4401_f12/sramop.pdf

[7]. http://smithsonianchips.si.edu/ice/cd/MEM96/SEC08.pdf

[8]. Reconfigurable computing thetheoryandpractice of FPGA- Based computation, Scott Hauck and Andr´e DeHon

[9]. http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/dsm_c_generating_reports.htm-.Journal of Hybrid Information TechnologyVol.3, No.1, January, 2010