Volume 7, Issue 2, February – 2022 International Journal of Innovative Science and Research Technology

ISSN No:-2456-2165

Implementation of Self Addressing RAM

Kiran P V, Naveen Kumar Kanavi, Gita Reshmi, Veena A
Department of Electronics and Communication Engineering,
Proudhadevaraya Institute of Technology,
Hosapete, India

Abstract:- This paper aims to implement a RAM which can address itself to load the data into its memory. Using Xilinx 14.1 ISE for simulation and synthesis the RAM of size 16 x 32 is implemented with consecutive addresses generated automatically by the additional circuit in the design.

Keywords:- *RAM, BRAM, Single Port BRAM, Dual Port BRAM.*

I. INTRODUCTION

The basic memory structures are divided into RAM and ROM. The RAM, also known as Read/Write memory is capable of both Read and Write operations, whereas ROM is read only and can be useful in program memory of a computing system. The basic operations in a RAM requires an enable signal for both read and write operations, clock for synchronization and data to be written into memory locations. The address decoders are needed to reach the memory locations in the memory for both operations which significantly increases the complexity of the design.

II. BRAMS IN FPGAS

Xilinx FPGAs provide SelectRAM memory blocks that can be utilized for implementing RAM, ROM, FIFO, CAMs, large look up tables, buffers, shift registers etc. The Spartan family of devices supports two configurations of memories such as Dual Port RAMs and Single Port RAMs. The BRAMs when used in dual port configuration have two independent access ports which support read and write data operations. Each port has its own clock and enable signals. If used in single port configuration, it has only one set of clock and enable signals as shown in the fig1 and Fig 2.

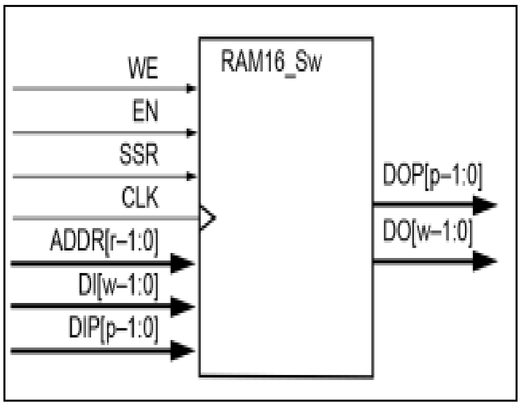
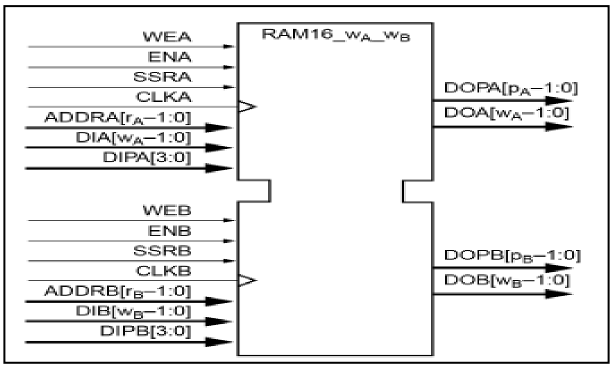


Fig. 1: Single Port **BRAM**

Fig. 2: Dual Port BRAM



Utilizing these configurations we can implement variety of applications that involves memories, such as FIFO, CAM, ROM, etc.

III. ALGORITHM

In this paper we discuss the algorithm that implements a RAM that does not require address generation by address decoders that are commonly found in any memory implementations. Instead it relies on self addressing scheme where each location of the memory is addressed by data itself which is part of any location. Therefore data that we refer here is partly actual data that we store in a location and additional bits that act as an address for next location. The overall algorithm is shown in Fig 3.

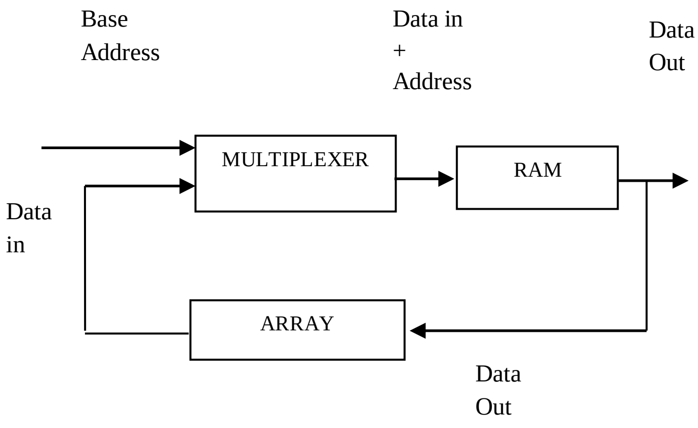


Fig. 3: Block Diagram of ᵕ n

The Multiplexer block selects the Base address which is the address from where we need to write the data in to the memory or next address where data is to be written into the memory. Array block stores the set of data values to be written into RAM at different locations. And RAM block involves implementation of a single port RAM instantiated through the VHDL code.

The data from the array is combined with Address by the Multiplexer and this data is written into location specified by the output from the multiplexer.

IV. IMPLEMENTATION

The implementation is done using Xilinx 14.1 ISE and VHDL programming language. The entity is shown in Fig 4. It has three input ports for Base Address, Clock and Write Enable. One output port for reading the data out from the memory. Fig 5 shows the internal architecture of the algorithm which is implemented using structural modeling in VHDL where it has four components. The component **mux** implements the functionality of a multiplexer, **Arry** impalements the data array, the component **incmenter** implements auto increment for address and **dualpram** for implementing single port RAM using VHDL component primitive.

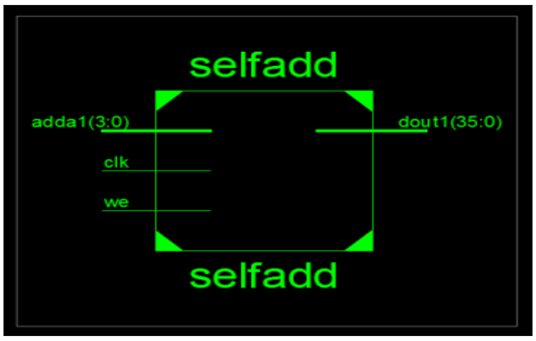


Fig. 4: Design Entity

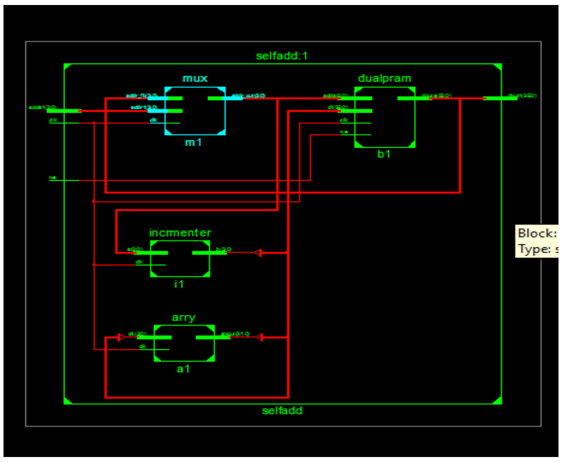


Fig. 5: Architecture of Entity

V. OUTPUT WAVEFORMS

Fig.6 shows output waveforms produced by simulation tool ISE simulator. The simulation results consists of Clock, Enable and Data out signals.

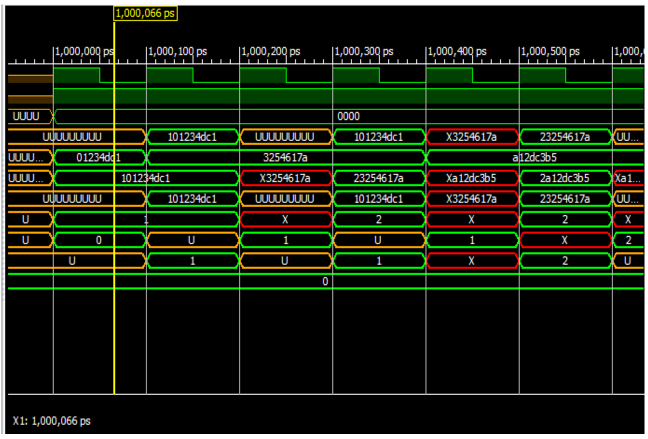


Fig. 6: Simulation Waveforms

VI. DEVICE UTILIZATION

The device utilization is found by Xilinx Synthesis tool.

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	40	126800	0%
Number of Slice LUTs	40	63400	0%
Number of fully used LUT-FF pairs	39	41	95%
Number of bonded IOBs	42	210	20%
Number of Block RAM/FIFO	1	135	0%
Number of BUFG/BUFGCTRLs	1	32	3%

Table 1: Device utilization

VII. CONCLUSIONS

In this paper we implemented an algorithm that loads a whole set of data values from the array into various memory locations of RAM without involving address decoders. The algorithm can write data into memory locations with data itself as an address which reduces the dependency on decoders. The algorithm was successfully verified using Xilinx simulation tools.

REFERENCES

[1.] Kylie Locke, "Parameterizable Content Addressable Memory", Xilinx Application Note, March 2011.
[2.] "Using Block RAM in Spartan-3 Generation FPGAs", Xilinx Application Note, March 1, 2005.
[3.] Dave Astels, "Memories", November 2021.