iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VII **Month of publication:** July 2022

DOI: https://doi.org/10.22214/ijraset.2022.45763

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VII July 2022- Available at www.ijraset.com

Design and Verification of LC3 Microcontroller

Kirana U N
ECE Department (VLSI Design and Embedded systems), PES College of Engineering Mandya, Karnataka (India)

Abstract: ***The main focus of the work is on utilising System Verilog to verify and debug the LC-3 Microcontroller, a 16-bit RISC processor. The utilised LC-3 Design Under Test (DUT) contains numerous flaws in the Fetch, Decode, Execute, and Controller sub-design units as well as flaws throughout. Numerous intricate SV features, including OOPS, Randomization, Functional Coverage, Assertions, and UVM, are used to identify the problems. For testing and determining the origin of design problems, the System Verilog Hardware Verification Language is combined with the Mentor Graphics Questa Simulation Environment. As long as it adheres to the design standards, the LC-3 microcontroller used for verification is presumed to function flawlessly. Otherwise, any behaviour that is not in line with the design specifications is considered a bug. The paper offers an effective approach for Verification Engineers in the Embedded Systems Industry to use System Verilog, the newest trend in the EDA Industry today, as their preferred language for debugging.***

Keywords: OOPS, Randomization, Assertions, UVM, debugging.

I. INTRODUCTION

The data and control paths of a PIPELINED LC-3 microcontroller with a rich instruction set are verified in this project. This article will walk you through the microcontroller's implementation and specifications. The goal is to present you with the basic first steps in interacting with the system that you will need as you continue through this programme[1].

The foundation of any modern microprocessor's or microcontroller's smooth operation is effective synchronisation and timing, which System Verilog supports. The availability of interfaces, which serve as a connection between the test bench and the top module, clocking blocks, which provide synchronised signals to and from the test bench and top module, and OOPS concepts like classes and handles, which ultimately lead to efficient randomization to provide random values to the Design Under Test (DUT), as well as functional coverage, which explicitly determines the percentage of bins (Verification Requirements) which have been covered by the use of the test bench. Due to its user-friendly features and potent OOPS ideas, System Verilog is the language of choice for the Hardware Verification of such complex entities, including the LC-3 Microcontroller. In the study, the control path and data path of an explicitly non-pipelined LC-3 are verified (RISC). This paves the door for the verification of any DUT and offers a highly effective method for using the System Verilog Hardware Verification Language (HVL) for debugging the LC-3 Microcontroller[3]. The following is how the paper is set up: In Section II, the methodology used is described. From a hardware perspective, the LC-3 Microcontroller is described in Section III. Section IV discusses the outcomes with the examination of output. The paper is concluded in Section V.

II. METHODOLOGY

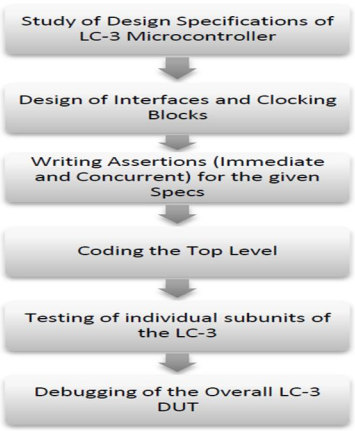


Fig 1: Flow chart of the methodology adopted.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VII July 2022- Available at www.ijraset.com

The methods used for testing and debugging the LC-3 Microcontroller is highlighted in Fig. 1 below[1]. The LC-3 microcontroller's design specifications are first thoroughly examined. Following the construction of interfaces and clocking blocks, a top-level model is created for debugging the entire microcontroller. Concurrent and instantaneous assertions are built for the debugging test bench [2]. The microcontroller's separate components are then examined, which essentially reveals the flaws that exist throughout the complete LC-3 Microcontroller DUT. The preferred language for this verification is System Verilog[3].

III. DESCRIPTION OF THE LC-3 MICROCONTROLLER

A. LC-3 Microcontroller and its Design Specifications

The LC-3 Microcontroller is believed to be a straightforward RISC Processor [4] capable of executing Arithmetic and Logical Operations as well as Memory Operations for the purposes of Verification and Debugging using System Verilog.

The Program Counter (PC) is presumptively set to 3000H whenever the LC-3 is reset. There is a signal called enable updatePC that, when set to high, causes the current PC value to be increased by one. The signal npc out stores the Program Counter's increment value. In accordance with the design specifications for carrying out the specific arithmetic, logical, or memory operation, the Instruction Register (IR) is a 16-bit register that contains the bits 0 through 15 in it. The data types on the LC-3 Microcontroller are integers with the 2's complement. The LC-3 can carry out a total of 15 commands, including a 4 bit Op-Code. The offset is comprised of the final 11 bits. The Microcontroller has a total of 7 Registers, R0-R7, which are chosen from bit Indices 9–11 (Destination Register) and 6–8 (Source Register), respectively, as shown in the accompanying Table.

Table 1: Bit Index Values and Corresponding Register Selected

Bit Indices (9-11 for DR)/(6-8 for SR)	Register Selected
000	R0
001	R1
010	R2
011	R3
100	R4
101	R5
110	R6
111	R7

The LC-3 has a Program Status Register Called the NZP (Negative-Zero-Positive) [5] for storing the flag register changes due to Arithmetic/Logical Operations. The NZP is used as follows:

N- High-> Negative Number
Low-> Positive Number
Z- High->all bits of Number 0s
Low->all bits of Number 1s
P- High->Positive Number
Low-> Negative Number
In the conditional jump instructions, the NZP is combined with Masks.

According to the procedure, there is a reduction operation together with a bitwise AND/ Addition/ NOT (Inversion). The sign extension of the bit values according to the addressing mode is denoted by the symbol sxt(n):

1) *Immediate Addressing Mode:* SR1 is used to operate on and move the "Contents of Immediate Value Register" (imm5 sign extended) to the Destination Register.
2) *Register Addressing Mode:* SR1 and SR2 are used to operate on and move the "Contents of Source Register 2 (SR2)" to the Destination Register.

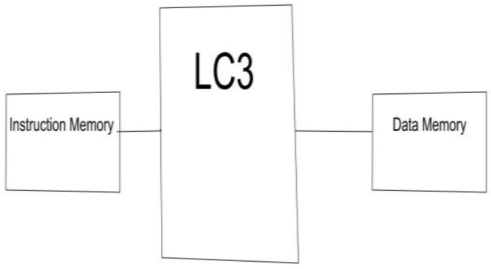


Fig 2: Interfacing of LC3 to the Instruction and Data Memory

B. FETCH Subunit of LC-3

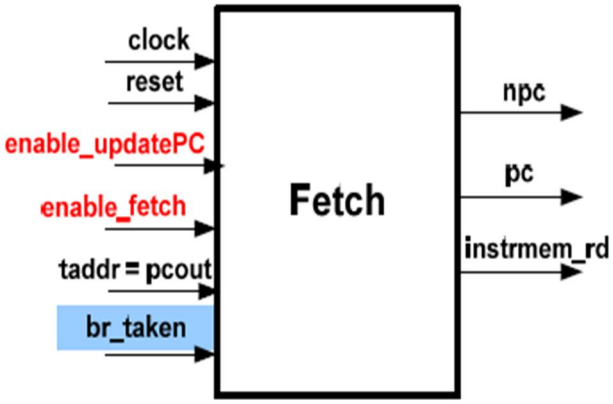


Fig 3: fetch subunit of LC3

The most fundamental function of the Fetch Subunit, which is an integral part of the DUT, is to fetch[6] the Op-codes from the Instruction memory and send them to the DECODE Subunit for decoding and carrying out the necessary operation through the correct identification of the Arithmetic/Logical/Memory Based Operation[3].
The fetch module generates the appropriate Program counter from which to read values. When fetch is enabled, it sends the instruction's address from the instruction memory (from where the 16 bit instruction is loaded into LC3). This data is delivered in the form of a 16-bit address value known as the programme counter. The programme counter's beginning address is 16'h3000, which is also the value set when LC3 is reset. Fetch also sends an output called instrmem_rd, which tells the decode block to read instructions from instruction memory[2].

C. DECODE Subunit of LC-3

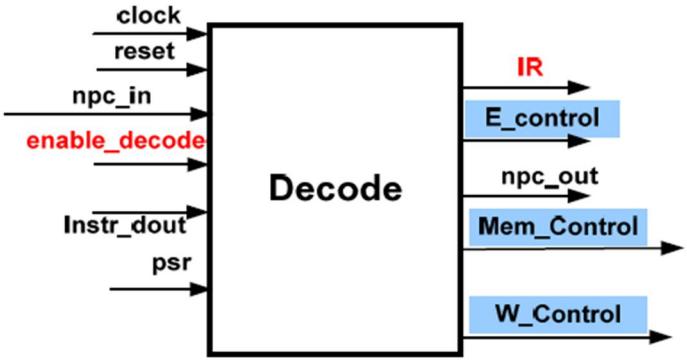


Fig 4 : Decode subunit of LC3

The decode subunit is the LC-3 DUT's second most crucial component. It receives the signal values from the FETCH [8] and decodes them to determine the appropriate operation that has to be carried out[3].
The decode block's goal is to provide suitable control signals for a specific instruction, therefore the data sent by Fetch as npc is received, and the 'decode' is done on the 16bit value. The decoder divides the 16bit address into three outputs: a signal that controls the Execute unit, which in turn controls the type of operation that will be performed, another signal that determines the right choice between the flowing for a write to the register file, and a signal that enables the selection of the right set of states for memory based operations[2].

D. Execute Subunit of LC-3

The 16bit instruction is computed in the execute module, which is the heart of the LC3 microcontroller. This module is closely related to the Writeback module because it collects all of the data from it. This fundamental link between the Execute and Writeback modules is also required for all memory-related and control actions[2].
The LC3 microcontroller's central processing unit, or block, is where data corresponding to a particular instruction is handled. The E Control signal determines the type of manipulation and the kind of data that will be used. Additionally, this block is intimately related to the Writeback unit, from which it derives all of its data, specifically the contents corresponding to SR1 and/or SR2. Additionally, this block handles the manipulations associated to PC operations for LEA (as well as other Memory and Control activities, for that matter)[1].

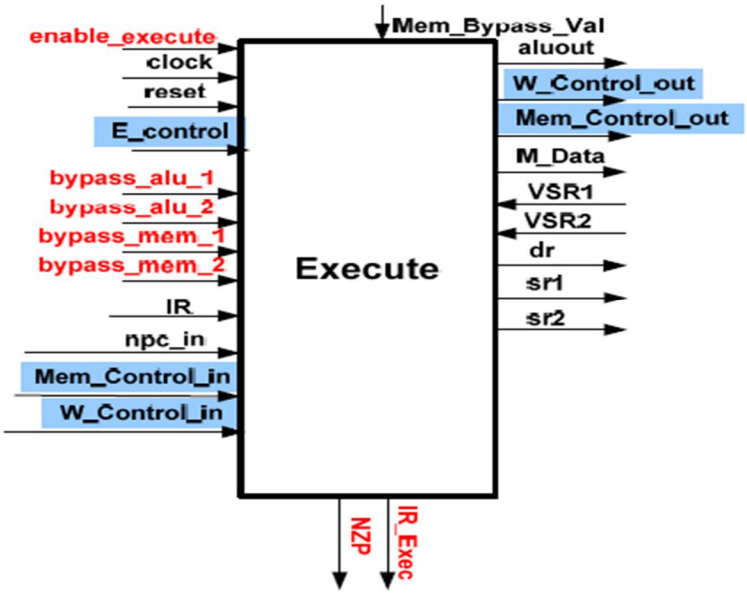


Fig 5 : Execute subunit of LC3

E. Memaccess subunit of LC3

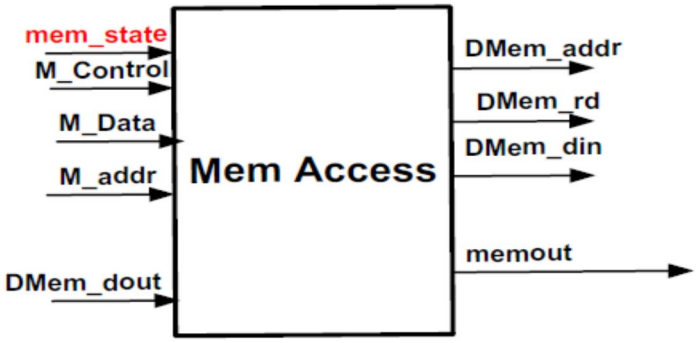
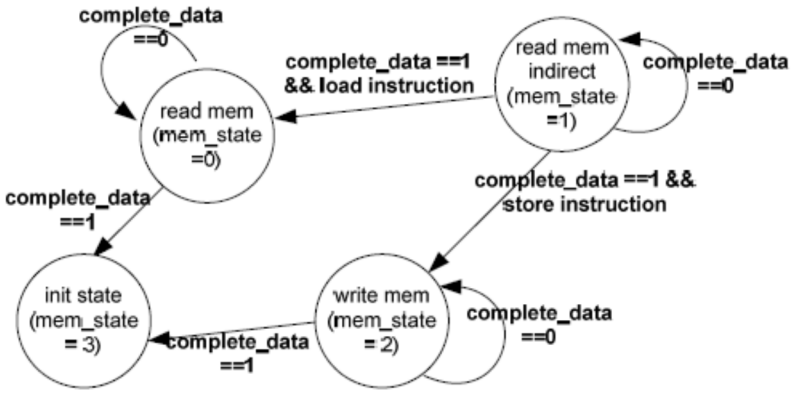


Fig 6 : Memaccess subunit of LC3

When a Memory-based action is detected at the Execute block's output, the mem state value changes into the state depicted below.The rest of the pipeline should be stopped during these transitions while waiting for the completion of memory-related operations.



When
mem state = 0 (reading memory for loads)

1) Mem state = 2, DMem addr = M addr for LDR, LD, and DMem dout for LDI (previous value read in is used as address) (writing memory for stores) For STR, ST, and STI, DMem addr equals M addr (previous value read in is used as address); DMem din equals M data;
2) mem state = 1 (reading from memory for indirect addressing)
3) Dmem rd = 0 for writes and 1 for reads; Dmem addr = M addr; Dmem din = 0.
DMem addr = z, DMem rd = z, DMem dout = z, and mem state = 3;

F. Controller

The controller manages the transition from one stage of the computation to the next. This 16 bit instruction passes through a 4-stage processing mechanism to perform each instruction, and after the processing is through, a new instruction is fetched. This can be illustrated using a basic ALU operation: the instruction is first retrieved, decoded, and then executed. The controller generates a signal at the end of this cycle to fetch the next instruction from the instruction memory's next address.

IV. RESULTS

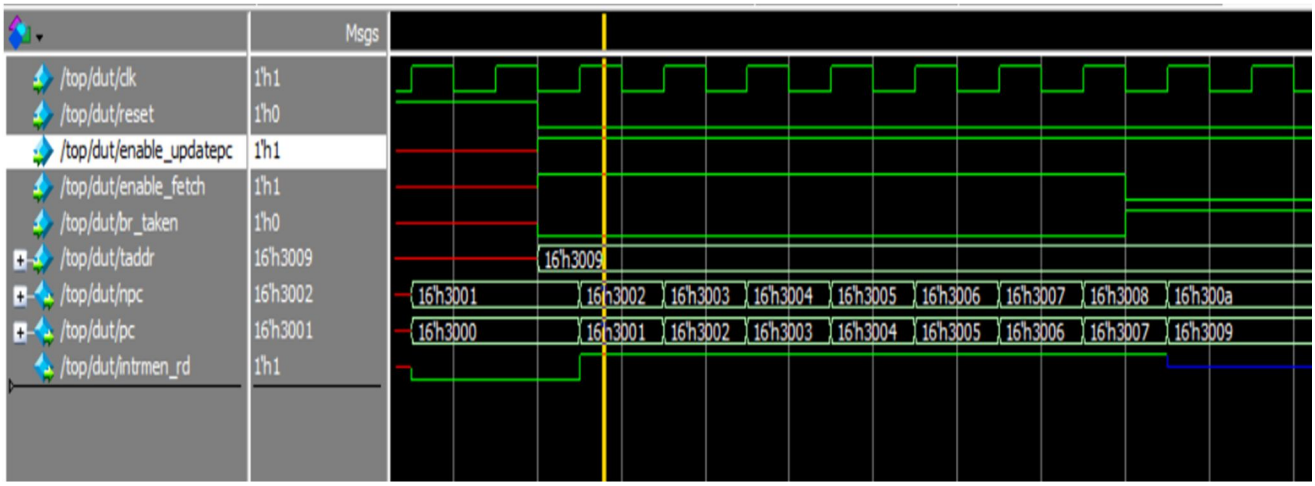


Fig 6 : Fetch Module output

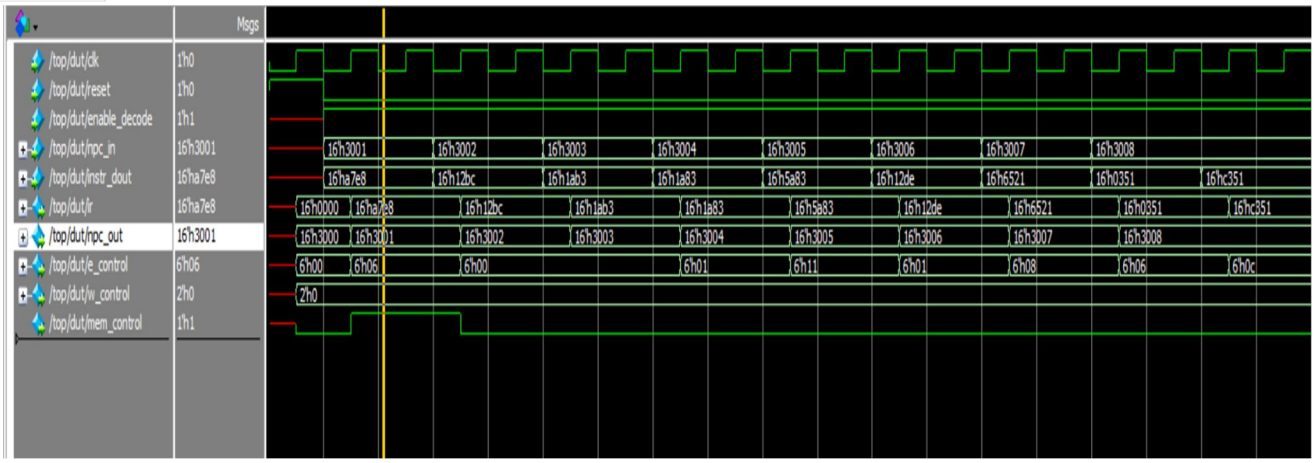


Fig 7 : Decode Module Output

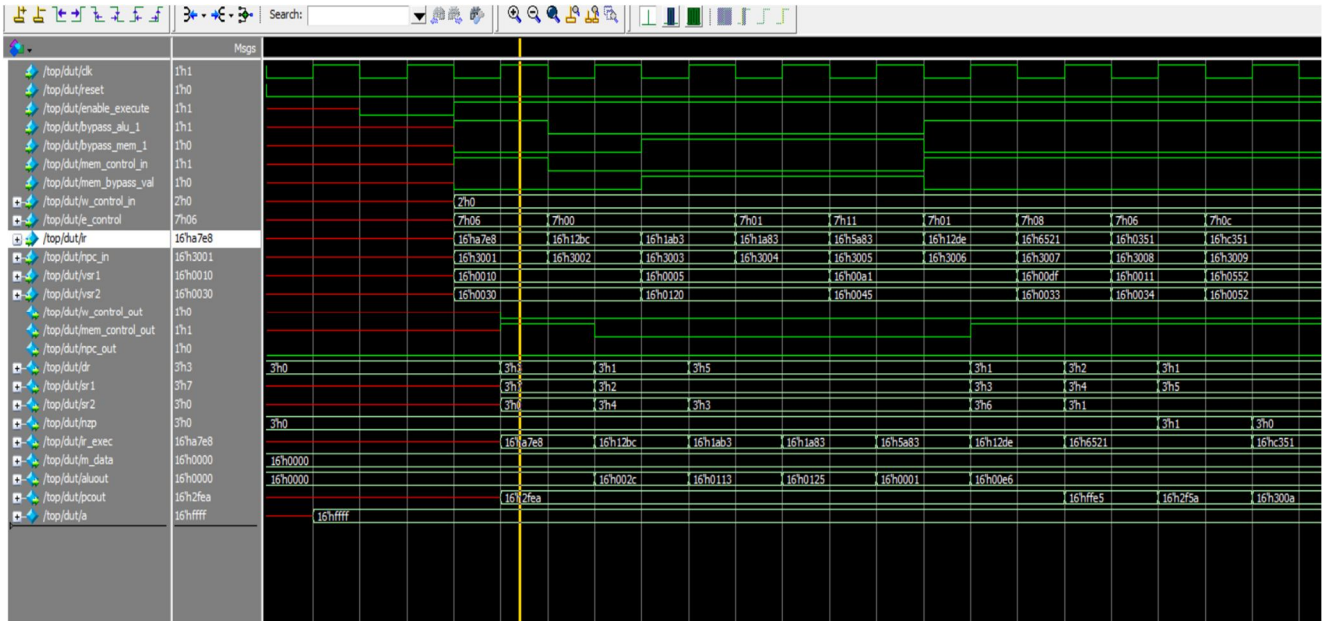


Fig 8 : Execute Module Output

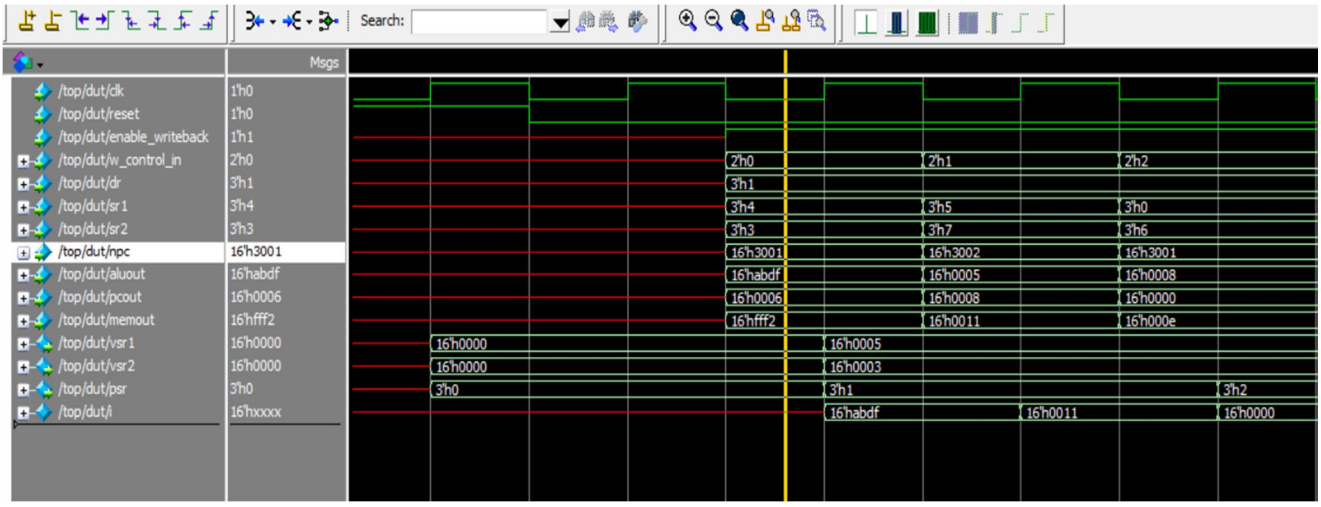


Fig 9 : Writeback Module Output

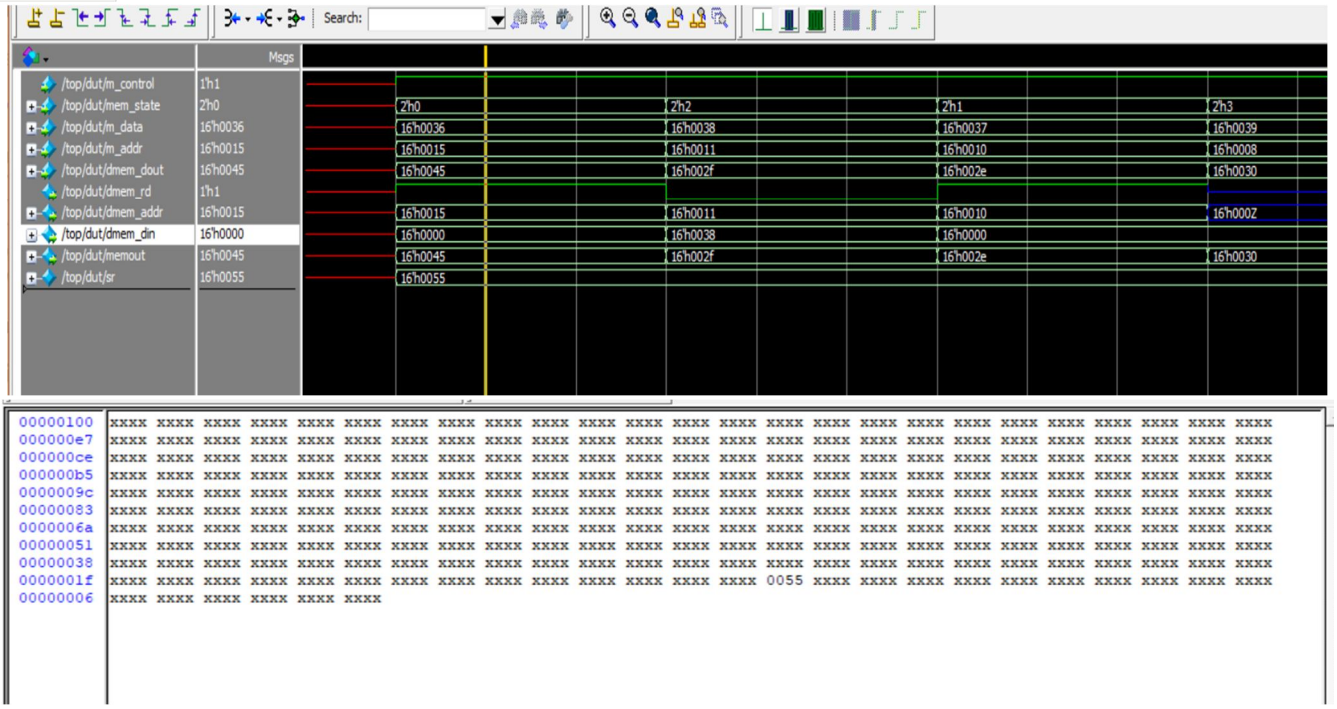


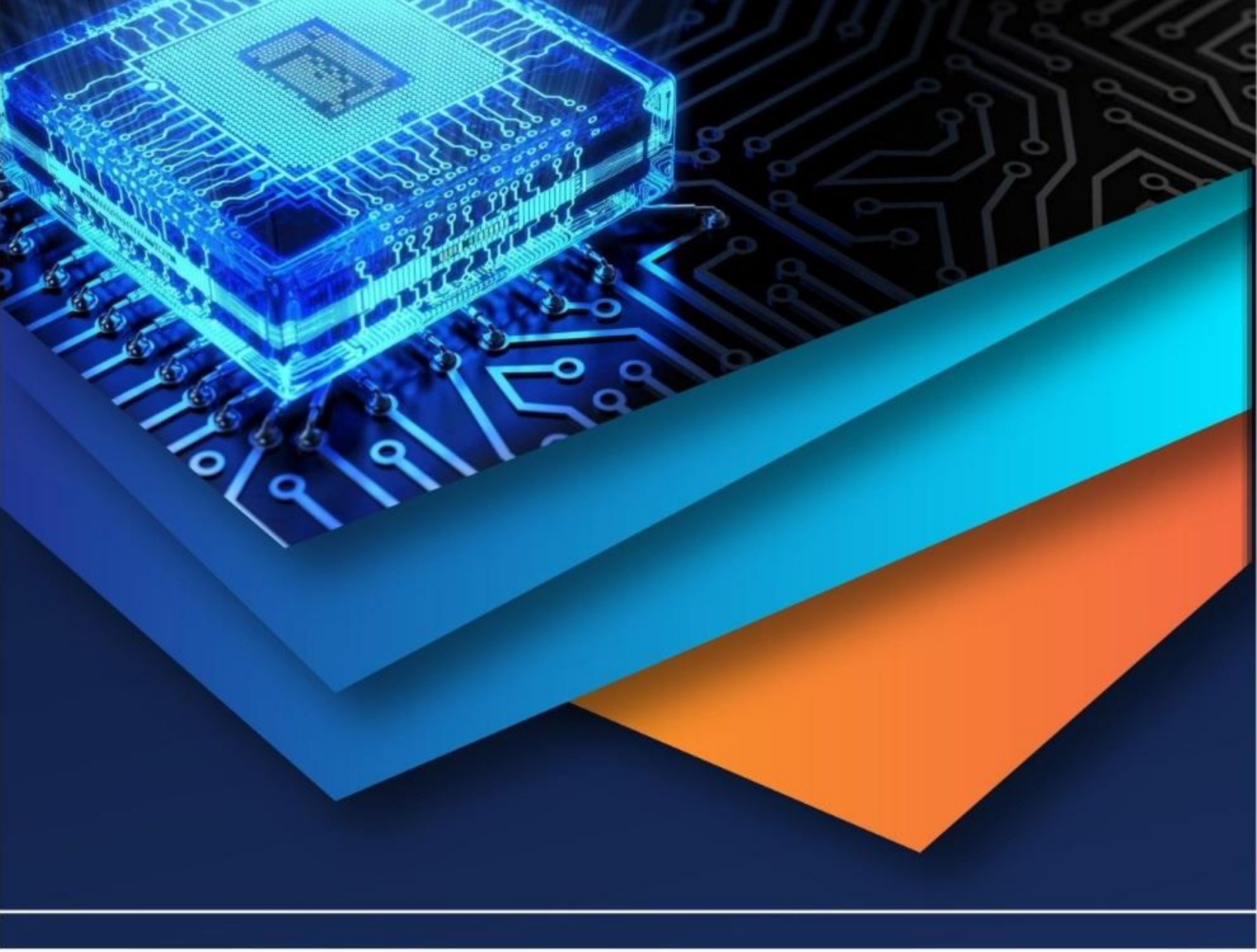
Fig 10 : Mem Access Module Output

V. CONCLUSION

The here mentioned LC3 controller has been designed using the newer RISC design strategy. This controller has used Non-pipelined methodology for design, and therefore the chance of getting an error due to non-availability of previous values is forestalled. The memory operations performed is Load Effective address.

REFERENCES

[1] "LC-3 ASIC Design Verification", ECE 745, North Carolina State University, USA
[2] Devyani Gera, Mehul Garg, "Design of Non-Pipelined LC3 RISC Microcontroller", IOSR Journal of Computer Engineering, Sep 2014
[3] J Bhasker, "Verilog HDL Synthesis", Oct 1998
[4] R Nikhil, "Bluespec System Verilog: efficient, correct RTL from high level specifications", Formal Methods and Models for Co-Design, 2004
[5] S. Das, R Mohanty, P. Dasgupta, "Synthesis of system verilog assertions", Proceedings of the Conference on Design, automation and test in Europe: Designers' forum, March 2006
[6] M. Keaveney, A. McMahon, N.O'Keeffe, K.Keane, J.O'Reilly, "The development of advanced verification environments using System Verilog",, IET Irish Signals and Systems Conference, February 2008
[7] Rakhi Nangia, NK Shukla, "Functional Verification of I2C core using System Verilog", International Journal of Engineering, Science and Technology, Vol. 6, No. 4, April 2014
[8] Anuja Dhar, Ekta Dudi, Hema Tiwari, "Coverage driven verification of I2C protocol using System Verilog", International Journal of Advanced Research in Engineering and Technology, Volume 7, Issue 3, May- June 2016



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY