Design and Integration of NAND Flash Memory Controller for Open Power-based Fabless SoC

Shanmukha Sai Nikhil Myramuru, S. Chandra Mohan Reddy, Gannera Mamatha

Abstract: ***NAND Flash Memory has replaced EEPROM and hard drives as Non-volatile. Data is stored in sequential order in NAND Flash Memory. NAND Memory is a type of flash memory widely used in mobile phones and System on Chips (SoC). The Memory controller supports an 8-bit NAND Flash interface and streaming interface towards AXI4. The data transfer between AXI4 and NAND Flash Memory is carried out by using NAND Flash commands sequences. The AXI4 Interface enables the usage of various protocols. To improve the flash memory controller's data access speed. This project intends to design, develop, and integrate a NAND Flash memory controller using an AXI4 interface for an open POWER Processor A20 fabless SOC. The Flash Memory Controller includes Finite State Machines (FSM) and AXI4 bridge logic. Using Mentor Graphics® and Xilinx's Vivado design suite, the test results were based on behavioral simulation and synthesis.***

Keywords: ***SoC, Memory Controller, NAND Flash Memory, AXI4, A2O***

I. INTRODUCTION

NOR and NAND is two non-volatile flash memory technologies. Nand Flash is occupying U disk at a growing rate in terms of development. In the embedded market, such as MP3/MP4 and mobile phones, Nor Flash is used in embedded devices with smaller storage capacities [1]. Nand Flash controller design idea and source code analysis, Non-volatile flash memory technologies include NOR and NAND. It altered the circumstance where EPROM and EEPROM were used. NAND Flash was introduced by Toshiba. NAND Memory is a low-cost per-bit memory with good performance. The code must be read back into the system RAM. NOR has a high transmission efficiency and is particularly cost-effective at small capacities [6]. NAND Flash has a high storage cell density but a slow erase and write speed. The data is stored in memory blocks. The blocks are used for erasing and programming[3]. The erase procedure is completed first, and the writing procedure in Flash Memory is initiated only after all empty NAND devices have been wiped. Creating an erase operation in NAND is more straightforward when compared to NOR. Erasing NAND devices, on the other hand, are implemented in blocks ranging from 8 to 32KB, and it only takes 4ms to perform the same operation[6].

A. A20 CORE

The A20 core is a 64-bit instruction that is out-of-order and multi-threaded. Set of architecture cores that were created as a processor for embedded use in the system-on-chip. It's best for optimising single-threaded performance. It has the same modular design approach and fabric structure as its parent high-strength throughput A2I predecessor. The auxiliary ExeCUtion Unit (AXU) is tightly tied to the core, opening up a plethora of options for special-purpose designs for new markets facing the challenges of modern workloads. The A20 has improved single-threaded performance and supports 2-way SMT with POWER IS 2.07. The A20 core was designed to optimize single-threaded performance and is targeted at 3+ GHz in 45 nm technology.

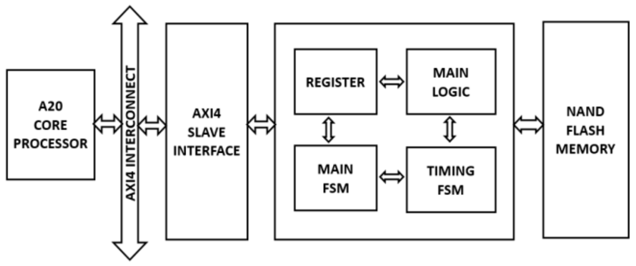


Fig 1. A2O core with Memory Controller Block

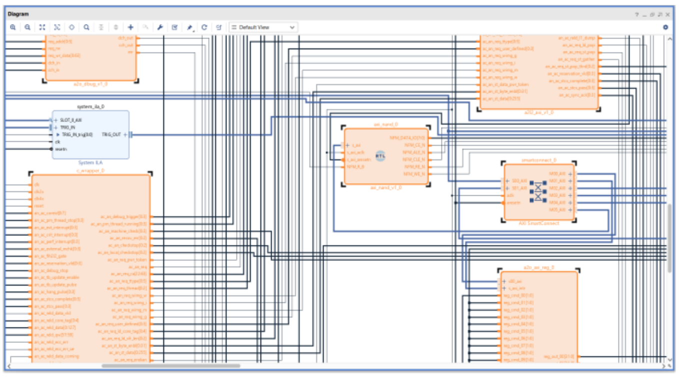


Fig 2. SoC Intergration Block Design

Manuscript received on 15 March 2022 | Revised Manuscript received on 10 December 2022 | Manuscript Accepted on 15 December 2022 | Manuscript published on 30 December 2022.

*Correspondence Author(s)
Shanmukha Sai Nikhil Myramuru*, M.Tech Student, Department of Electronics and Communication Engineering, JNTUACEA, Ananthapur, (A.P), India. E-mail: mshanmukha1997@gmail.com, ORCID ID: https://orcid.org/0000-0002-5199-9091
Dr. S. Chandra Mohan Reddy, Associate Professor, Department of Electronics and Communication Engineering, JNTUACEA, Ananthapur, (A.P), India.
Dr. Gannera Mamatha, Assistant Professor, Department of Electronics and Communication Engineering, JNTUACEA, Ananthapur, (A.P), India.

Fig 2 contains a2o reg, a2o debug, core wrapper, NAND Flash Controller, AXI Smart connect, a2o_axi_reg and a2l2_axi. These modules Intergrate with AXI Smartconnect

II. AXI4 NAND FLASH MEMORY CONTROLLER

1. Design Features

This controller will handle NAND Flash memory with a capacity of 2GB. Reset, Page Program, Page Read, Block Erase, and Read ID are all functions that are supported. All programs and erase actions are supported by the read status. An AXI4 host interface was used to implement this strategy for SoC connections. The AXI4 slave interface can perform a variety of tasks[3].

2. Functional Description

An AXI4 host interface is included in this system, which is utilized to connect NAND Flash Memory. Functions are included in the AXI4 slave interface. Valid, ready, id, response, and handshake are some of the signals it transmits. An 8-bit input/output link multiplexes the Flash Command, Data, and Address. Input/Output data is latched to the rising edge of WE n while CE is low[2]. The flash command is latched if CLE is high and ALE is low. The flash address is latched when both ALE and CLE are high. The Page Program is activated when WE n increases and CE decreases. Page Read is conducted while RE n is growing and CE is low.

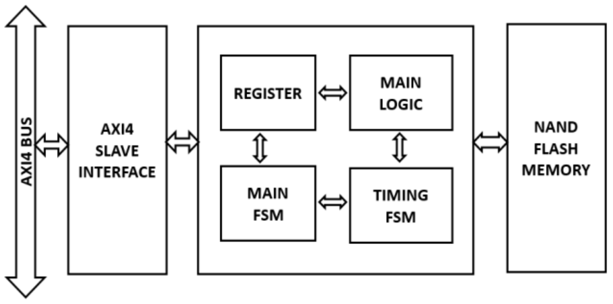


Fig 2. AXI4 NAND Flash Controller

III. PROPOSED DESIGN

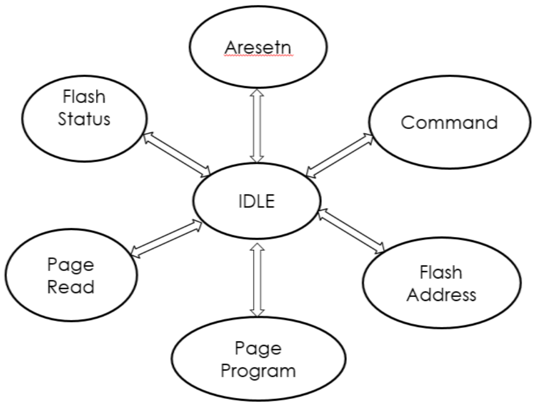


Fig.3.FSM Design

The Finite State Machines (FSM) are main modules in the Flash Memory Controller, and Main and Timing FSM are main modules in the Flash Memory Controller. The AXI4 bridge logic is used in these two FSMs[4]. The Main FSM's state machines translate commands from the AXI4 slave interface and send them to the Timing FSM. NAND Flash Memory Controller design based on Finite State Machines (FSM). These two FSMs manage and control NAND flash memory. A command, an address, write data, read data, and flash status are all included in the state transition[3].

The main FSM enters the idle state when the reset procedure is done. The address state sends 5 address cycles (two-row and three-column) to Flash memory, whereas the command state sends Flash commands[4]. The wait address and the end address are based on address cycles. Data is sent into flash memory by the page program state. The read state of a page is the read data in flash memory. The NAND flash operation status in the controller is displayed in the Flash status. The Main FSM is used by the Timing FSM. It sends commands from the A20 core processor to the Main FSM, which then performs a certain action at the appropriate time and changes the FSM's state. These Timing FSM run a page program and read a page in Memory Controller[5].

Memory Controller Operations

A. Reset
B. Page Program
C. Page Read
D. Block Erase
E. Read Status

A. Reset Operation

When the aresetn and nf r/b (Flash ready/busy) input signals are both active, The Main FSM calls the Timing FSM to perform the Memory Controller's Reset action. When the Reset operation is completed, the Timing FSM executes Flash commands (FFh) to Flash Memory and waits for the next operation before returning to the idle state [6].

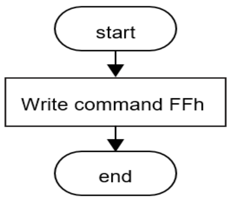


Fig.4 reset flow chart

B. Page Program

When the awvalid (address valid), wvalid (write valid), and nf r/b (Flash ready/busy) input signals are active high, the AXI4 starts page program operation. The Main FSM transitions from idle to the Page Program in Timing FSM. The awaddr (write address) and wdata (write data) input signals are used by the AXI4 slave interface to write flash command and write flash address in Memory. The AXI4 slave interface supplies the write flash address in advance via register logic. The Timing FSM initiates the command and address states based on control signals from the main FSM. The flash address state sends the 5 address cycles (two rows and three columns) to the Flash memory. The program state machine transmits write data to flashmemory in pages[6].

The AXI4 interface transmits the input wlast (write last) signal to the flash controller when the last byte of data is sent. When all of the preceding stages have been accomplished, the command 10h is written to the Flash by the state machine.

The state machine recognises the signal nf r/b after tWB seconds. If the r/b is set to 1 by the Flash, the page programme method is complete. To determine whether the page programme operation was successful, the state machine sends the read status command code 70h to the Flash. If the status returned by Flash is 1, the page program execution is successful. The page program operation is terminated by the principal FSM in Controller. The Flash status-ready signal is transmitted to the AXI4 interface when the page program is completed. The timing FSM goes into standby mode and awaits the controller's next action. A command, address, and page program state machine that generates valid signals in flash memory are known as a timing FSM [6].

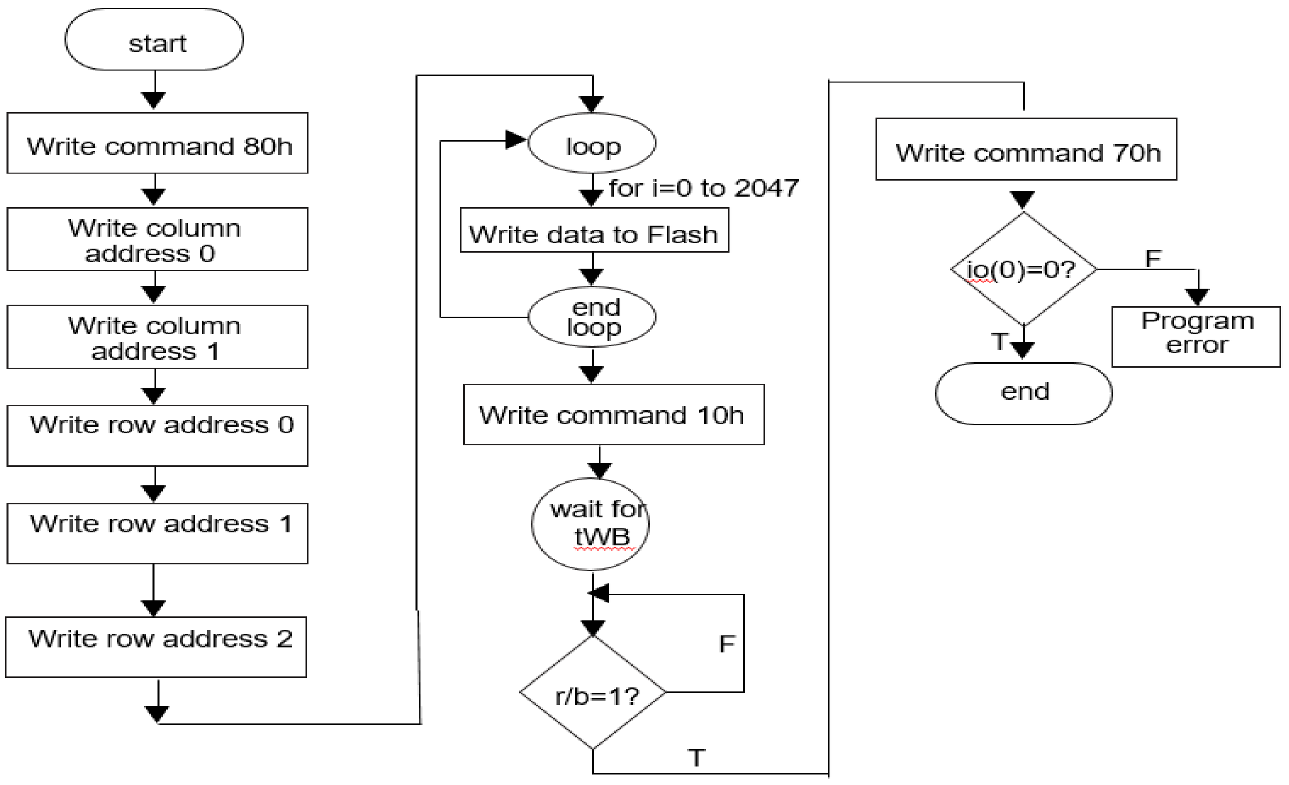


Fig.5 Page Program flow chart

C.Page Read

When the arvalid (address read valid), rready (read ready), and nf r/b (Flash ready/busy) input signals are active high, the AXI4 starts the page read process. In Timing FSM, the Main FSM switches from idle to Page read. The AXI4 slave interface uses araddr (read address) and rdata (read data) input signals to read flash command and read flash address in Memory. The read flash address is delivered in advance via register logic in the AXI4 slave interface. The Timing FSM initiates the command (00h) and addresses states based on control signals from the main FSM. The 5 address cycles (two rows and three columns) are transmitted to the Flash memory by the flash address state. The state machine then sends the Flash the command code 30h. The state machine recognises the signal nf r/b after tWB seconds[6].

The flash address state sends the 5 address cycles to the Flash memory (two rows and three columns). After that, the state machine sends the command code 30h to the Flash. The signal nf r/b is recognised by the state machine after tWB seconds. The page read operation is terminated by the main FSM in Controller. The Flash status-ready signal is transmitted to the AXI4 interface when the page read is complete. The timing FSM goes into standby mode and awaits the controller's next action. For commands, addresses, and page read state machines, timing FSM to generate valid signals in flash memory[6].

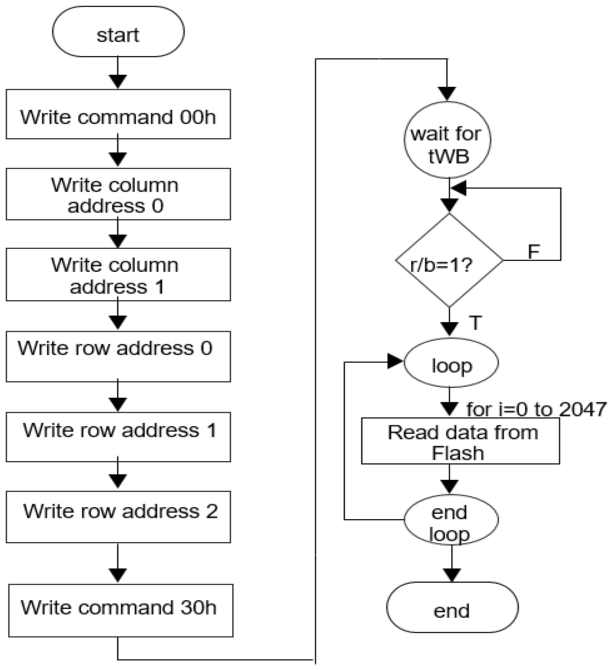


Fig.6 Page Read flow chart

D.Block Erase

When the awvalid (address valid), wvalid (write valid), and nf r/b (Flash ready/busy) input signals are active high, the AXI4 starts block erase operation. The Main FSM moves from idle to block erase state in Timing FSM. To write a flash command and a flash address into memory. The block is frequently used as the essential unit in the erasing operation. The erasing procedure must transmit three address cycles. Three address cycles are transmitted when the erase operation's first command 60H is sent, and the memory status indication signal R/B is pulled low when the block erase operation's second command DOH is sent. When the data in the target block is erased, the NF R/B becomes high, and the block can be further evaluated by the read status operation. Whether the erase procedure was successful or not[6].

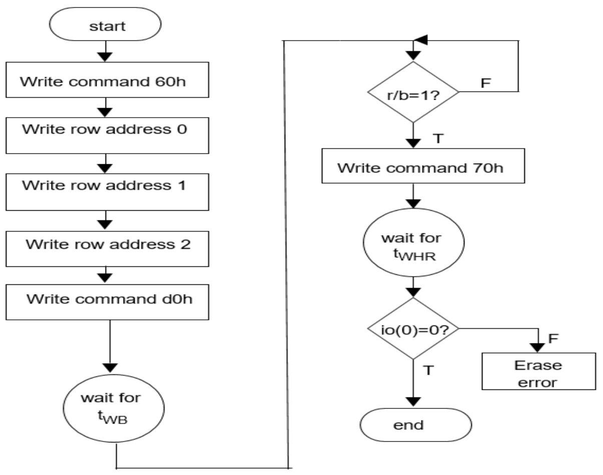


Fig.7 Block Erase flow chart

E. Read Status

The signal R/B can be used to determine if the read and write operation is in progress after executing the page program or page read instruction. Users can also utilize the read status operation to see if the page read or write page is successful by using the read status operation. The read status action only sends one instruction cycle (70h) before returning to I/O. Read through the port the status value. If the continuous read status operation does not necessitate delivering the read status command regularly. Read the data on the I/O under the control of the RE_n signal to get the status value[6].

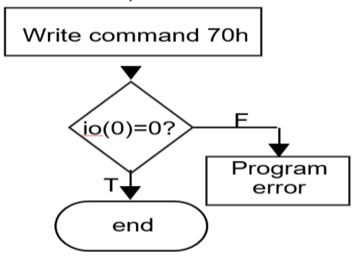


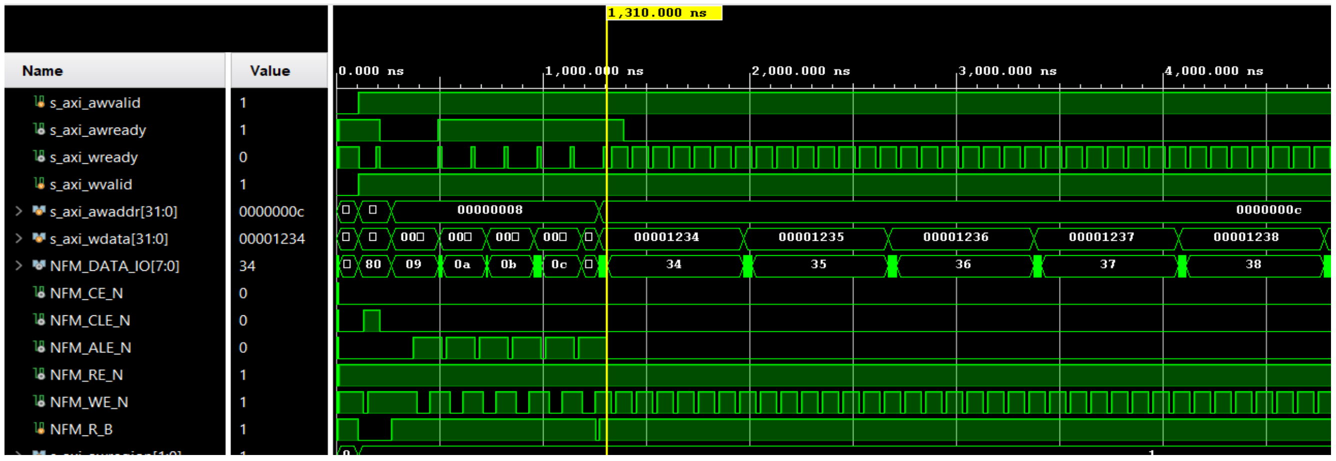
Fig.8 Read Status flow chart

IV. SIMULATION RESULTS

Page Program, Page Read, Block Erase, and Read Status are included in the test bench for this design.

A.Page Program

The page programme command is shown in Figure 9. The controller initiates this procedure by sending an 8-bit flash command, followed by 5 cycles of column and row addresses. On the positive edge of the WE n signal, write data will be given to the I/O because the memory is now ready for writing[6].



Name	Value
s_axi_awvalid	1
s_axi_awready	1
s_axi_wready	0
s_axi_wvalid	1
s_axi_awaddr[31:0]	0000000c
s_axi_wdata[31:0]	00001234
NFM_DATA_IO[7:0]	34
NFM_CE_N	0
NFM_CLE_N	0
NFM_ALE_N	0
NFM_RE_N	1
NFM_WE_N	1
NFM_R_B	1

Fig.9 Page Program

B. Page read

The page read command is shown in Figure 10. This method is started by delivering an 8-bit flash instruction, followed by 5 cycles of column and row addresses from the controller. On the positive edge of the RE n signal, read data will be supplied to the I/O because the memory is now ready for reading [5].

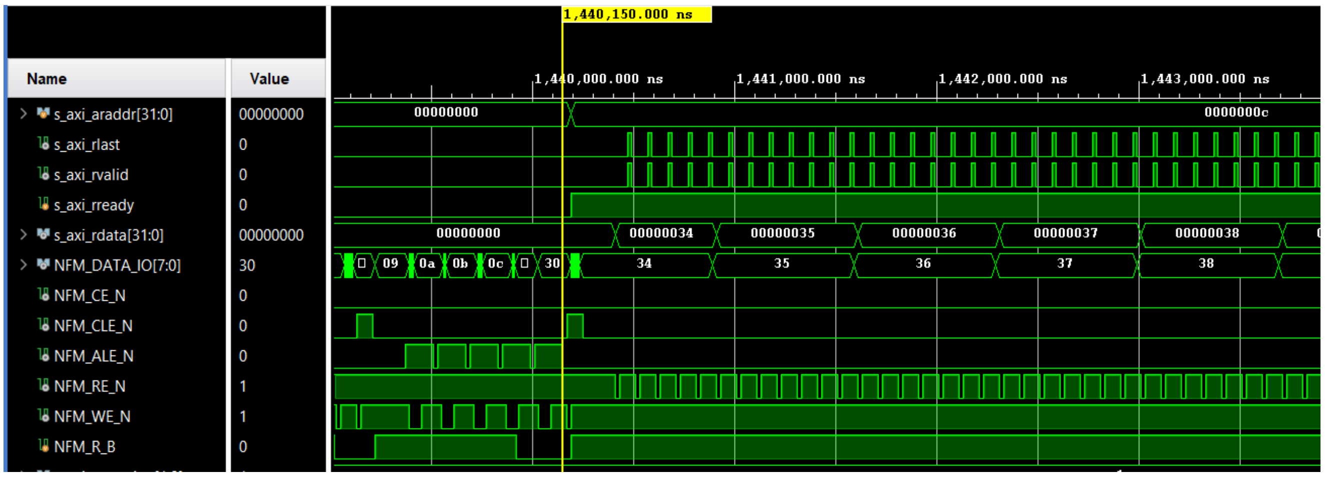


Fig 10. Page Read

C.Block Erase

The block delete command is seen in Figure 11. When CLE is high, the FSM will send the initial command. On the positive edge of WE n, when ALE is high, it will send a row address every three cycles. Then mail the order a third time. Erasing the block after this remembrance takes some time [4].

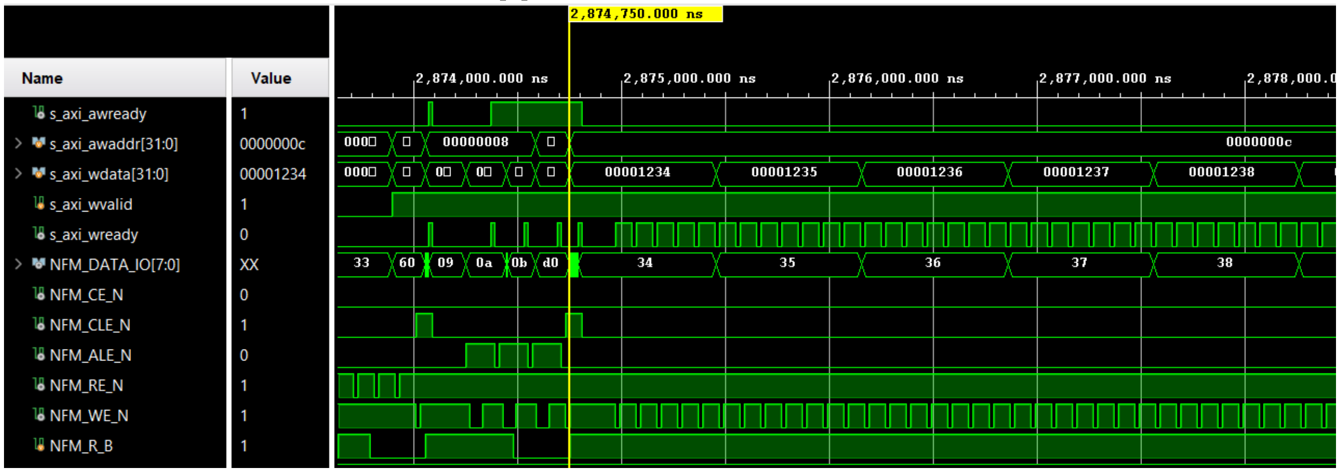


Fig 11. Block erase

D.Read Status

After each activity, the host will send a read status command to check the memory status. The read status simulation result is shown in Figure 12. When the reset operation instruction 70h is latched, the status will be latched on the positive edge of RE n after some time [6].

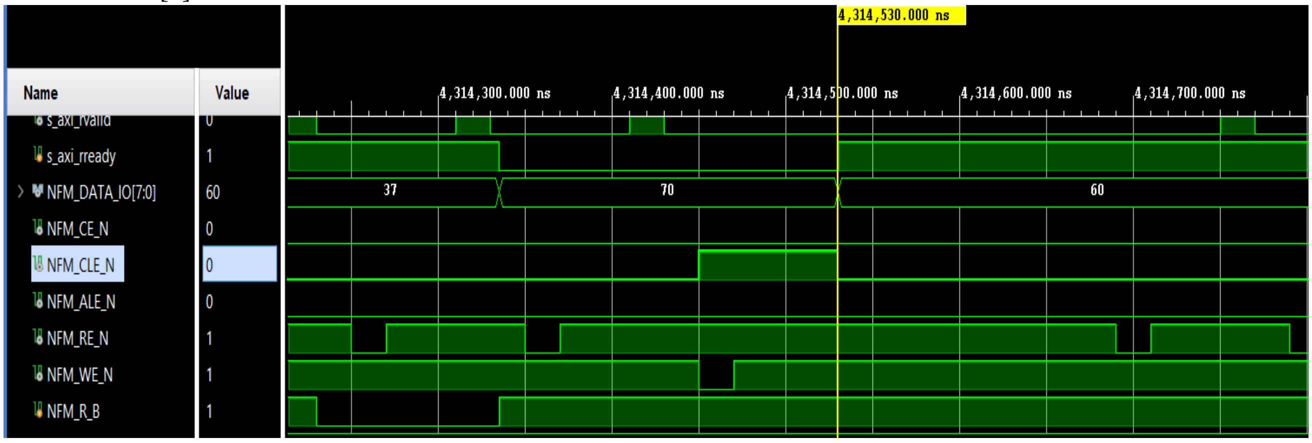


Fig 12. Read Status

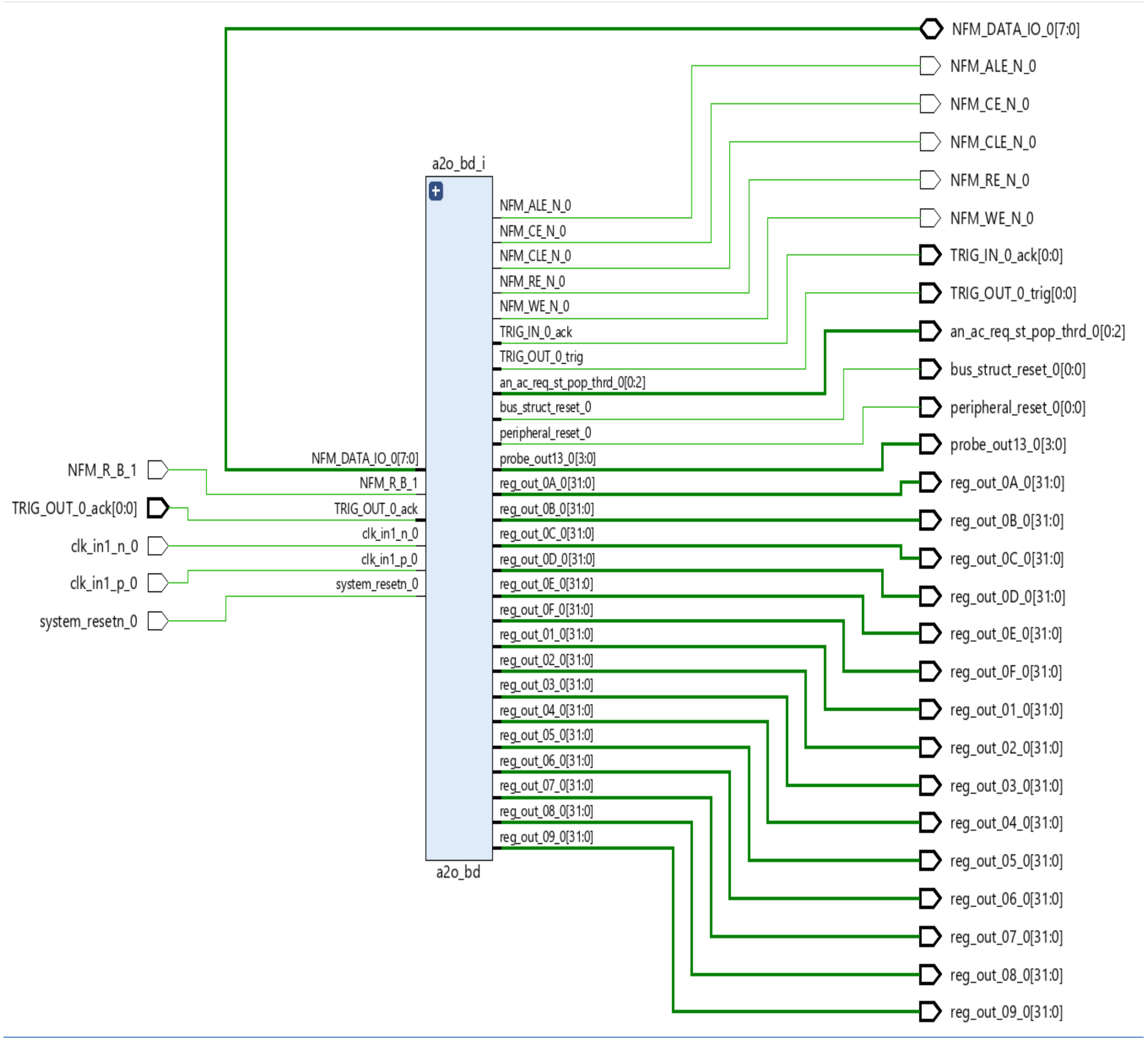
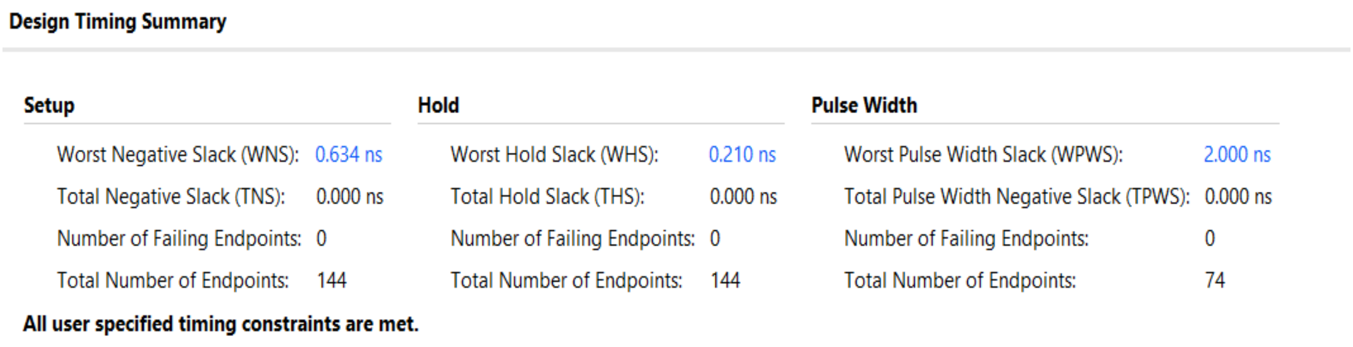


Fig 13. Block Diagram



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.634 ns	Worst Hold Slack (WHS):	0.210 ns	Worst Pulse Width Slack (WPWS):	2.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	144	Total Number of Endpoints:	144	Total Number of Endpoints:	74

All user specified timing constraints are met.

Fig 14. Static Timing Analysis

After implementing the design with Verilog hardware description language, I used EDA tools to test and synthesise it. According to simulation results, the design's timing and function are both performing properly. This design should be added to SoC as an IP, and two FPGA development boards should be used to test it. The Xilinx Zynq®-7000 SoC-Z-7007S- XC7Z007S FPGA development boards are connected to a bespoke circuit board. 250 MHz is the system clock. Use MT29F2G08AABWP as the physical NAND Flash device to test the design. The test results indicate that the technologies described in this paper can greatly improve the performance of a NAND Flash controller.

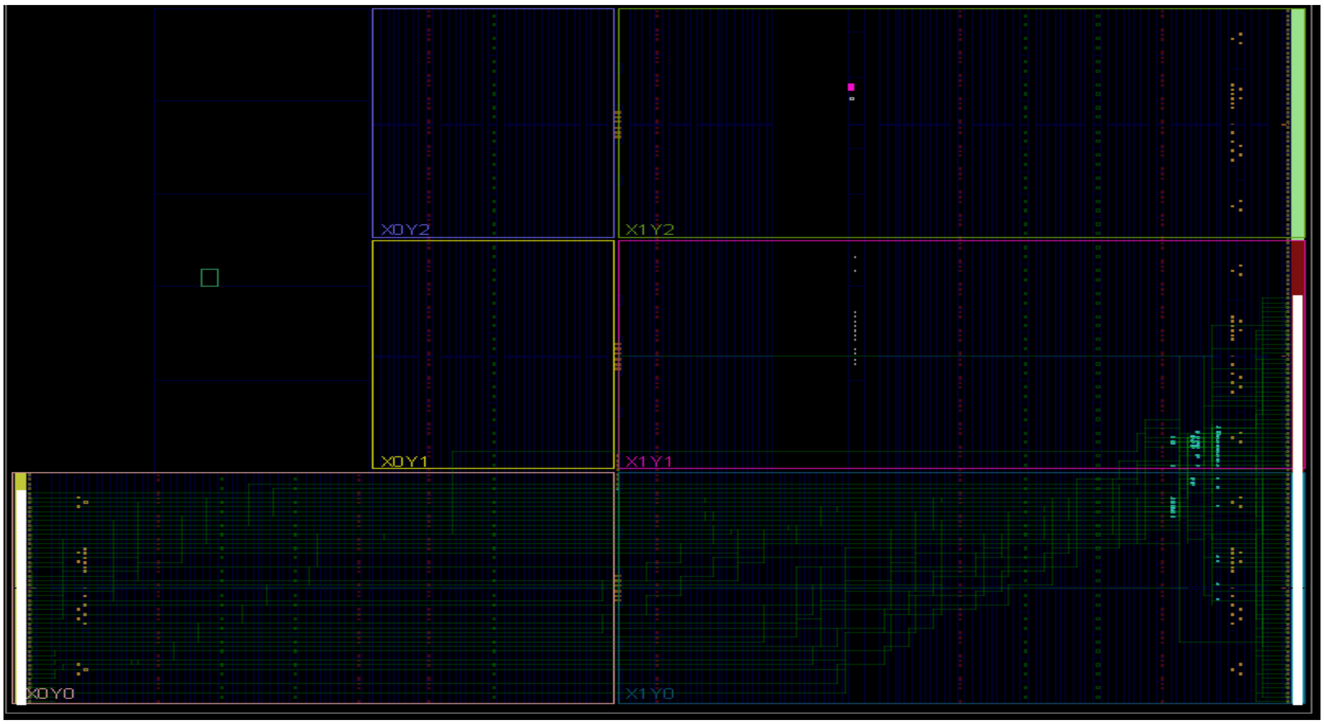
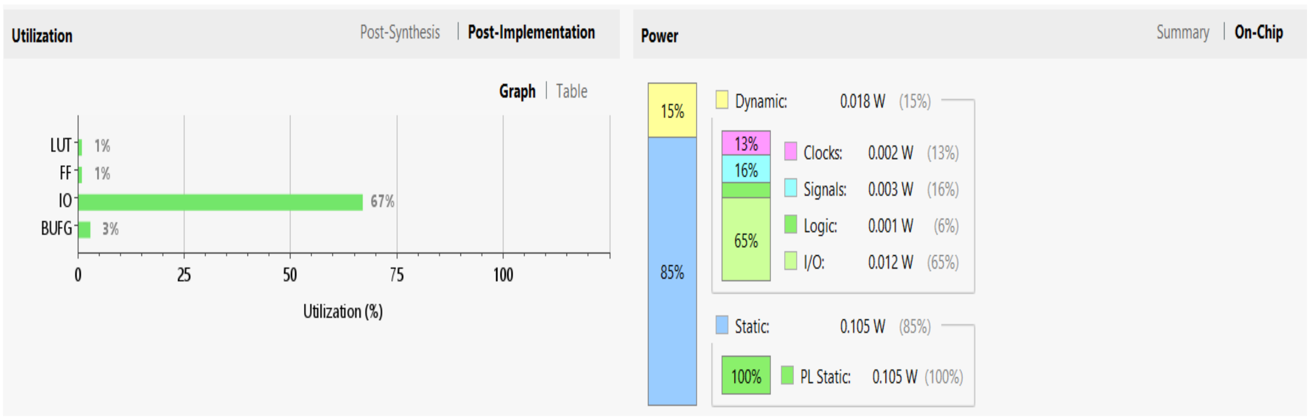


Fig 15. Implemented Design



Resource	Utilization (%)
LUT	1%
FF	1%
IO	67%
BUFG	3%

Power		
Dynamic:	0.018 W	(15%)
Clocks:	0.002 W	(13%)
Signals:	0.003 W	(16%)
Logic:	0.001 W	(6%)
I/O:	0.012 W	(65%)
Static:	0.105 W	(85%)
PL Static:	0.105 W	(100%)

Fig 16. Utilization and Power Report

V. CONCLUSION

This work developed, simulated, synthesized and implemented the AXI4 NAND Flash Controller, which was utilized in A20 processor-based fabless soc. The Controller achieves great data write and read speeds in NAND Flash Memory. Controller works at This proposed architecture has the potential to improve data transmission performance using AXI4. This design is verified in Xilinx Vivado and Mentor Questa using memory-based function verification. The plan for the future is to increase memory capacity and data transfer speed.

DECLARATION

Funding/ Grants/ Financial Support	Not receive.
Conflicts of Interest/ Competing Interests	No conflicts of interest to the best of our knowledge.
Ethical Approval and Consent to Participate	No, the article does not require ethical approval and consent to participate with evidence.
Availability of Data and Material/ Data Access Statement	Not relevant.
Authors Contributions	Myramuru Shanmukha Sai Nikhil M.Tech studies in jntua college of engineering jntua university, sole author for complete. My project guide is Chandra Mohan Reddy and G Mamatha given support for guidance. complete project did by me. remaining authors given support in educational, EDA tool access.

REFERENCES

1. Gong Xin, Dai Zibin, Li Wei, Feng Lulu, "Design and Implementation of a NAND Flash Controller in SoC" IEEE International Conference of Electron Devices and Solid-State Circuits, 2011. .(cited)
2. C. -S. Lin and L. -R. Dung, "A NAND Flash Memory Controller for SD/MMC Flash Memory Card," in IEEE Transactions on Magnetics, vol. 43, no. 2, pp. 933-935, Feb. 2007, doi: 10.1109/TMAG.2006.888520.(cited) [CrossRef]
3. T. Li and Z. Lei, "A novel multiple dies parallel NAND Flash Memory Controller for high-speed data storage," 2017 13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI), 2017.
4. S. T. Jose and C. Pradeep, "Design of a multichannel NAND Flash memory controller for efficient utilization of bandwidth in SSDs," 2013 International MutliConference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), 2013. .(cited)
5. K. Agarwal, V. K. Magraiya and A. K. Saxena, "Verification and Simulation of New Designed NAND Flash Memory Controller," 2013 International Conference on Communication Systems and Network Technologies, 2013.(cited)
6. Micron Technical note NAND Flash 101. An introduction to NAND Flash.(refer from websites cited)

AUTHOR PROFILE

M. Shanmukha Sai Nikhil, M.Tech. at Jawaharlal Nehru University Anantapur.He is currently an ASIC RTL Design Engineer. successfully delivered numerous ASIC projects and provided assistance up till tapeout while working with multiple clients on FPGA and ASIC projects. interest in system on chip, controller, and vlsi system design

Dr. S. Chandra Mohan Reddy M.E., Ph. D. Motivating and Talented professor driven to inspire students to pursue academic and personal excellence. Consistently strive to create a challenging and engaging learning in which students become life-long scholar and learner. Deeply invested in achieving tenure through administrative service committee contributions and an accomplishment-oriented approach to teaching. With my Credentials and Enthusiasm to Teaching Excellence for 20+ years of experience, as a skilled, Research-Oriented, and Dedicated Teacher with a recent Doctorate Degree from JNTUA.

Dr. Gannera Mamatha, M.Tech, Ph.D Assistant Professor in Electronic & Communications engineering, IEEE Senior Member Jawaharlal Technological University Anantapur College of Engineering, 515002. Having 18 years of Professional and Academic Experience in Teaching and A talented and inspiring professor who is driven to encourage students to strive for both academic and personal success. Always work to make learning challenging and interesting so that students develop into lifelong scholars and learners. deeply committed to getting tenure through contributions to administrative service committees and a teaching style that emphasises accomplishments. As a qualified, committed, and research-oriented teacher with a recent doctorate degree from JNTUA, I can bring my credentials and enthusiasm for teaching excellence to the table. I have over 15 years of experience.