Verification of NAND Flash Controller

Prajwala M J, Kush Desai, and Lili He, *Member, IAENG*

***Abstract*—NAND structure-based Flash memory-made SSDs are turning more popular than HDDs in several applications notably in consumer microelectronics devices and enterprise memory storage devices. The NAND controller is the heart of this system. It acts as a bridge for communication between the host system (usually the Personal Computer system) and the NAND Flash memory storage device. Unarguably, it can be said that it is an essential part of the NAND flash-based memory device and it helps to manage the directory containing file systems. The controller is also accountable for managing system features like wear levelling, error code correction (through ECC), and garbage data collection. With Flash memory devices growing larger in storage space and smaller in area or form-factor, the NAND controller design is converting to a smarter one and complex one, too. Notably, this outline is developed to determine a way to check the features and functionalities that the controller residing into the NAND Flash memory controller provides. To do so, the structured verification approach is used which is based on SV/System Verilog.**

***Index Terms*— Flash Controller, NAND Flash Memory, System Verilog, Verification**

I. Introduction

AMONG all available memory standards, flash memory has the lowest price per bit in today's world. The non-volatile property of flash memory helps retain the contents even when power is turned off. EEPROMs are a class of memories where the contents can be electrically programmed and erased. Flash memory is one such element in the family, where entire blocks of memories can be erased. Flash memories are more dense and can pack more memory in a given silicon area, faster in terms of read time, but has a shorter lifecycle (the number of times read-write operations can take place) [1]. This is why flash memory is used as program memory, whereas other EEPROMs are used as data memory. NAND flash memory has lower power for read/write operations compared to NOR flash memory and shows superior response times enabling faster programming. Hence, it is the preferred type of flash memory.

To interface a NAND flash device, the host machine requires an interface, called the NAND flash memory controller. It needs to be user configurable, so that the users can decide how the application uses the flash memory. The data read from the flash memory, or the data destined to be written in the flash is stored in a bidirectional dual-port buffer. An address translator creates a mapping between the physical address of the flash memory and a virtual address which the host device sees. This facilitates read/write/erase operations in the flash memory, without the host knowing the exact physical address. In the process of data transfer between the host and the flash memory, data can get corrupted due to issues with the transmission channel, clock jitter and other coupling issues. An error correction algorithm in the controller monitors any errors in transmission and reports it to the host for further analysis.

Manuscript received March 30, 2022; revised May 25, 2022. The Authors are with San Jose State University, Department of Electrical Engineering, San Jose, CA, USA (corresponding author to provide phone: 408-924-4073; fax: 408-924-3925; e-mail: lili.he@ sjsu.edu).

II. The Proposed NAND Flash Controller Design and Verification

A. Flash Controller Block Design

The DUT is a controller that communicates with the NAND Flash memory and the user defined host device. The flash memory is accessed by the controller in order to perform different operations like erasing a block at a time or reading the ID of the flash device or resetting the flash device, etc. The host device uses a virtual address to refer to the data present in it and provides a command to the controller (cmd) to indicate either a reset or read ID or block erase or a page read operation. The host device acts as a master to the controller (DUT)[2]. The controller executes the command sent by the host, either reads the data or clears a block of data from the flash memory or resets the flash memory. The NAND flash memory receives the command data and address through an 8-bit I/O port. The NAND flash memory acts as a responder to the read operation command and writes back into the controller dual-port buffer for write operation command.

The NAND flash interface mainly contains 7 pins some are input pins and some output pins. All the data is sent and received over the DIO [7:0] pin. The command and address are also sent through the DIO [7:0] pin which is a bi-directional pin. When WE_n is low, there is data on DIO [7:0] pin and CE is also low, the data gets written to the Flash device, similarly when RE_n and CE are low, read operation will take place. The address is obtained by the device when ALE goes high indicating that address is ready to be fetched but CLE has to be low in this case. Similarly when the CLE signal is high and the ALE signal is low, the command gets latched and the flash device receives the command to the operation that is expected to be performed.

Figure 1 represents the detailed block diagram of the controller that represents different modules that ensure the data moves perfectly from the user defined host device to the receiver which is the flash memory.

The controller contains various modules, where each

ISBN: 978-988-14049-3-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

module carries out a specific operation.

• Main FSM: The main FSM is one of the most important modules in the design, as it interprets the commands sent from the user defined host and outputs the accurate control signals to timing FSM [4].

• Timing FSM: The timing FSM is the module that controls the operation of the flash device, as it supervises that the flash device performs all the operations while following all the timing requirements.

• Data Buffer module: The data that is read from the flash memory device and the data to be written to the flash device has to be stored in a RAM. Hence, the buffer module within the controller is a dual port RAM that stores all the data, to and from the flash device.

• Address Counter module: In order to write or read the data the flash device needs a parameter called address, hence there is a module that is used to generate address signal, these addresses would be sent to the data buffer module and the data present in that specific address in the buffer is written to the flash device and vice versa. The module that enables to do all this is the address counter.

• ECC generator and detector: The special feature of the NFC (NAND Flash Controller) is its ECC generator module, which produces ECC when there is any operation performed by the flash device. This ECC code that is generated is used towards the end of different operations like host read, host write or block erase, to catch the errors in the operation and ensures to re do the task in case the operation fails.

The NFC supports various operations like reset, Read ID, block erase, program page (writing to the Flash) and Read page (the flash memory reads from the internal buffer). Each of these operations have their own flow and are stored in the main FSM module. Every time there is a command (nfc_cmd from the host), a specific FSM flow is triggered in the Main FSM module and the operation starts.

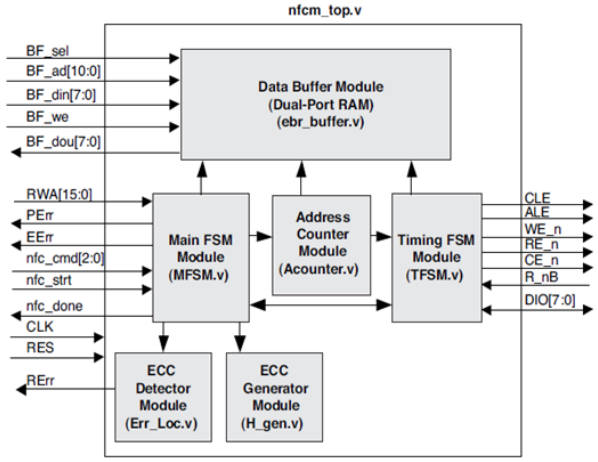

Figure 1: NAND Flash controller Block diagram

B. *System Verilog Verification Flow*

The verification process is used to test the functionality of the design called DUT which is the Design Under Test. The verification is carried out by generating different set of inputs, driving them through the design, capturing the obtained output and finally comparing the obtained results with the estimated output. The verification architecture has different classes to perform each operation, like initiating stimulus, driving these test inputs in the design, monitoring the working of design, etc. and all these classes will be labelled based on operation.

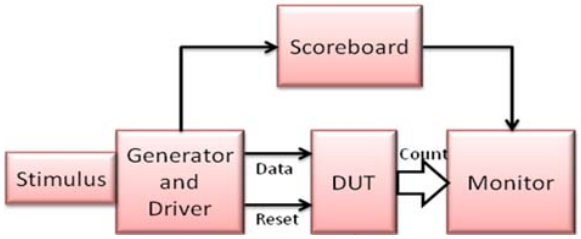

Figure 2: Basic flow of SV verification

In case of small and simple design architectures, the traditional method of verification with a fixed set of inputs to test the design often neglect the corner cases of verification. Hence showcase a limited capability to find faults in the design. On the contrary with the use of constrained random inputs to test the design, using score-boarding technique, also checking for functional coverage not only helps verify a small design but also carry out verification of complex and huge designs successfully [3]. The verification flow consists of a number of elements from transaction, generator, monitors, agents, scoreboards, environment and test to a top module. The Figure 2 shows a basic flow of System Verilog verification.

• SV Verification Hierarchy: The SV verification hierarchy has different classes as shown in Figure 3, each of which are used on a specific operation. The different classes do specific tasks and are connected to each other to transfer data to each other forming a good verification flow. The test class and the environment class start the hierarchy. Number of agents are defined in the environment class. A number of agent classes contain the definition of generator, driver and monitor classes .The scoreboard class is an important component of the hierarchy that verifies the functionality of the design and displays error messages in case the design fails to function as expected.

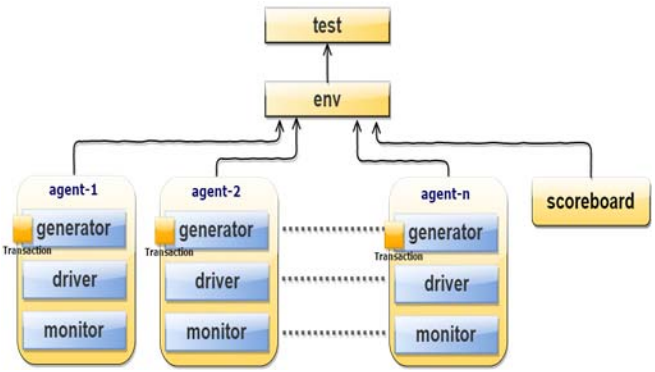

Figure 3: SV class hierarchy

ISBN: 978-988-14049-3-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

• SV Transaction class: The transaction class defines the activity generated by its particular agent, in order to drive to the DUT by using the driver class. In addition, the transaction class also acts as a placeholder for activities on the DUT signals which are represented in the monitor class.

• SV Generator class: The generator class creates the stimulus and randomizes the constraints in the transaction class. The stimulus that is created covers all the corner cases which is sent to the DUT through the driver. The generated stimulus is random and large in number so it is helps in verification of the most complex designs. Figure 4 shows the flow of transaction packets from one component to another.

• SV Driver Class: The driver class is a class that acts as a bridge between the generator and the DUT. The driver class receives the stimulus and drives packets of data to the DUT. The virtual interface object is initialized in the test top module. The virtual interface object helps the driver establish connection with the DUT in order to send and receive the data packets. The interface class acts as a bridge between the driver class and the DUT.

• SV Monitor class: The monitor class keeps track of the output of the DUT and further passes it onto the scoreboard. The monitor essentially keeps track of the activities of the signals defined in the interface class and converts it to packet level information that is forwarded to the scoreboard. The monitor is responsible for displaying the output of the DUT continuously and forwarding the information to the further blocks. In any given SV verification environment, there can be multiple monitors each to save the input and the output data. Hence named as input monitors or output monitors.

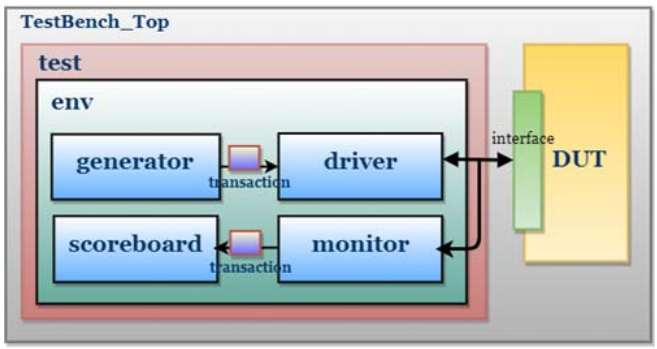

Figure 4: Data flow through verification environment

• SV Agent class: The agent class is referred to as a container class, as it is a collection of other components like monitors, drivers and generator that is specific to a protocol. It could be passive agents which do not include any active components like driver or sequencer which are involved in driving the data packets from the generator to the interface and vice versa. Monitors are hence categorized into the group of passive components and grouped in passive agents as they only monitor the signals on the interface. The drivers and sequencers on the other hand are grouped into the active agents. The Figure 5 shows the flow of data between the agent and DUT.

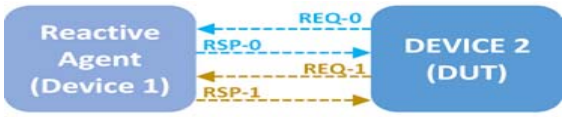

Figure 5: Data flow between Agent and DUT

• SV Scoreboard: The scoreboards are most important components in the verification environment as it is used to ensure the functionality of the design, which is achieved by comparing the data that is sent into the scoreboard class by two monitors (input monitor values and output monitor values) or comparing the obtained output data from the DUT to the golden reference values or expected values.

• SV Environment: The environment class is present at the top level of the verification hierarchy as it contains all the agents and different scoreboards. All the agents that belong to different protocols and all the scoreboards are defined in the environment class.

• SV Test: The test class contains the environment class in it and holds the responsibility of not only configuring the test bench but initiating the construction process of various test bench components like environment component that contains the agents and the agents that contain the drivers and generators. The test class also plays a crucial role in initiating the verification process by initiating the stimulus for the given design.

• SV Top Module: The top module is the root of the verification hierarchy. The components from generator to environment all lie within the top module. The module also has the clock initialized in it as well as the interface handle and the waveform dump tasks.

C. Operation Details

• Reset Operation: The RESET operation starts when the host sends a command 011 to the NFC through the nfc_cmd input pin. The nfc_start pin should also go high to trigger the operation. This nfc_cmd is interpreted by the main FSM module which starts the FSM flow that is specific to the reset operation.Hence the initial state of the FSM is triggered when it receives the nfc_cmd to be 011. Once the main FSM decodes that the operation is a reset operation, specific control signals are sent to the timing FSM. The timing FSM further writes a command FFh over the DIO [7:0] triggering a reset operation of the flash device. Once the command is written the main FSM switches back to initial state and indicates the same to the host by ensuring nfc_done goes high, and is set for the next task.

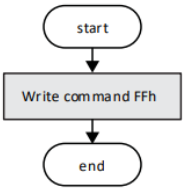

Figure 6: Reset Operation Flow Chart

ISBN: 978-988-14049-3-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

• Read ID Operation: The Read ID operation starts when the host sends a command 101 to the NFC through the nfc_cmd input pin. The nfc_start pin should also go high to trigger the operation. This nfc_cmd is interpreted by the main FSM module which starts the FSM flow that is specific to the Read ID operation. Hence the initial state of the FSM is triggered when it receives the nfc_cmd to be 101. Once the main FSM decodes that the operation is a Read ID operation, specific control signals are sent to the timing FSM. The timing FSM further writes a command 90h on the DIO pin, followed by an address 00h on the DIO pin. Once both the command and address are written to the flash, the readID is sent from the flash through the DIO pin in 4 consecutive clock cycles. Once the operation is done the main FSM switches back to initial state and indicates the same to the host by ensuring nfc_done goes high, and is set for the next task.

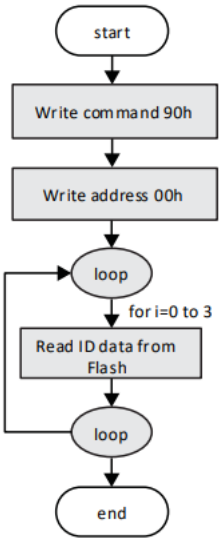

Figure 7: Read ID Operation Flow Chart

• Block Erase Operation: The block erase operation starts when the host sends a command 100 to the NFC through the nfc_cmd input pin. The nfc_start pin should also go high to trigger the operation. This nfc_cmd is interpreted by the main FSM module which starts the FSM flow that is specific to the block erase operation. Hence the initial state of the FSM is triggered when it receives the nfc_cmd to be 100. Once the main FSM decodes that the operation is a block erase operation, specific control signals are sent to the timing FSM. The timing FSM further writes a command 60h on the DIO pin, followed by an address code on the DIO pin (the address code is pre-set in the host). Once the address is written to the flash another command d0h is sent to the flash and waits for a fixed amount of time tWB. Towards the end of time tWB the FSM checks on the R_nB pin on the flash device, once the R_nB goes high, a command 70h is sent to the flash to test if the operation is successful. If a 1 is sent on the DIO by the flash device in return to the previous 70h command, the main FSM switches back to initial state and indicates the same to the host by ensuring nfc_done goes high, and is set for the next task else if DIO goes to 0, the EErr signal at the host goes high indicating the block erase error, hence the task has to be performed again.

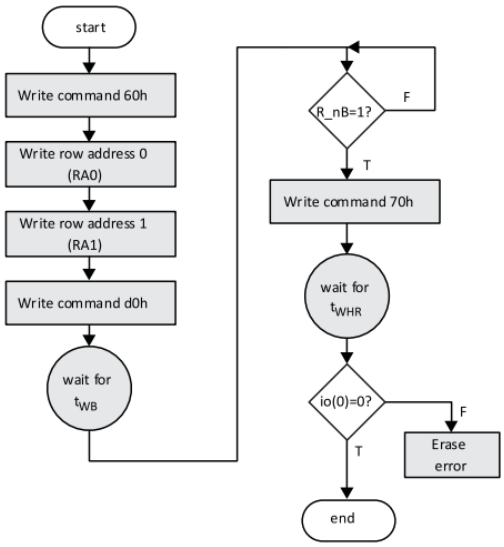

Figure 8: Block Erase Flow Chart

• Page Program Operation: The page program operation starts when the host sends a command 001 to the NFC through the nfc_cmd input pin. The nfc_start pin should also go high to trigger the operation. This nfc_cmd is interpreted by the main FSM module which starts the FSM flow that is specific to the page program operation. Hence the initial state of the FSM is triggered when it receives the nfc_cmd to be 001. Once the main FSM decodes that the operation is a page program operation, specific control signals are sent to the timing FSM. The timing FSM further writes a command 80h on the DIO pin, followed by an address code on the DIO pin (the address code is pre-set in the host). The FSM transmits the data in the dual port buffer to flash through the DIO pin.The data written to the flash is 2048 bytes. Command 85h is sent through DIO pin and 12 ECC bytes are written to the flash device. Once all the above commands and data is written to the flash, another command 10h is sent to the flash and waits for a fixed amount of time tWB. Towards the end of time tWB the FSM checks on the R_nB pin on the flash device, once the R_nB goes high, a command 70h is sent to the flash to test if the operation is successful. If a 1 is sent on the DIO by the flash device in return to the previous 70h command, the main FSM switches back to initial state and indicates the same to the host by ensuring nfc_done goes high, and is set for the next task else if DIO goes to 0, the PErr signal at the host goes high indicating a page program error, hence the task has to be performed again.

• Page Read operation: The Page Read operation is a read operation,where data is read from the flash device, where the data read from the flash is stored in the dual port buffer. The following steps and Figure 10 represents the flow of Page read operation. The page read operation starts when the host sends a command 010 to the NFC through the nfc_cmd input pin. The nfc_start pin should also go high to trigger the operation. This nfc_cmd is interpreted by the main FSM module which starts the FSM flow that is specific to the page read operation. Hence the initial state of the FSM is triggered when it receives the nfc_cmd to be 010. Once the main FSM decodes that the operation is a page read operation, specific control signals are sent to the timing FSM. The timing FSM further writes a command

ISBN: 978-988-14049-3-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

00h on the DIO pin, followed by an address code on the DIO pin that has the address to the data that would be read. The FSM transmits a command 30h and waits for time twB,once R_nB goes high the data can be read from the flash. 2048 bytes are read and saved in the buffer, followed by sending a command E0h on DIO pin to the flash and reading the 12 ECC bytes from the flash device to check if the Page Read operation was successful. After all the above flow the main FSM switches back to initial state and indicates the same to the host by ensuring nfc_done goes high, and is set for the next task else if DIO goes to 0, the RErr signal at the host goes high indicating a page read error, hence the task has to be performed again.

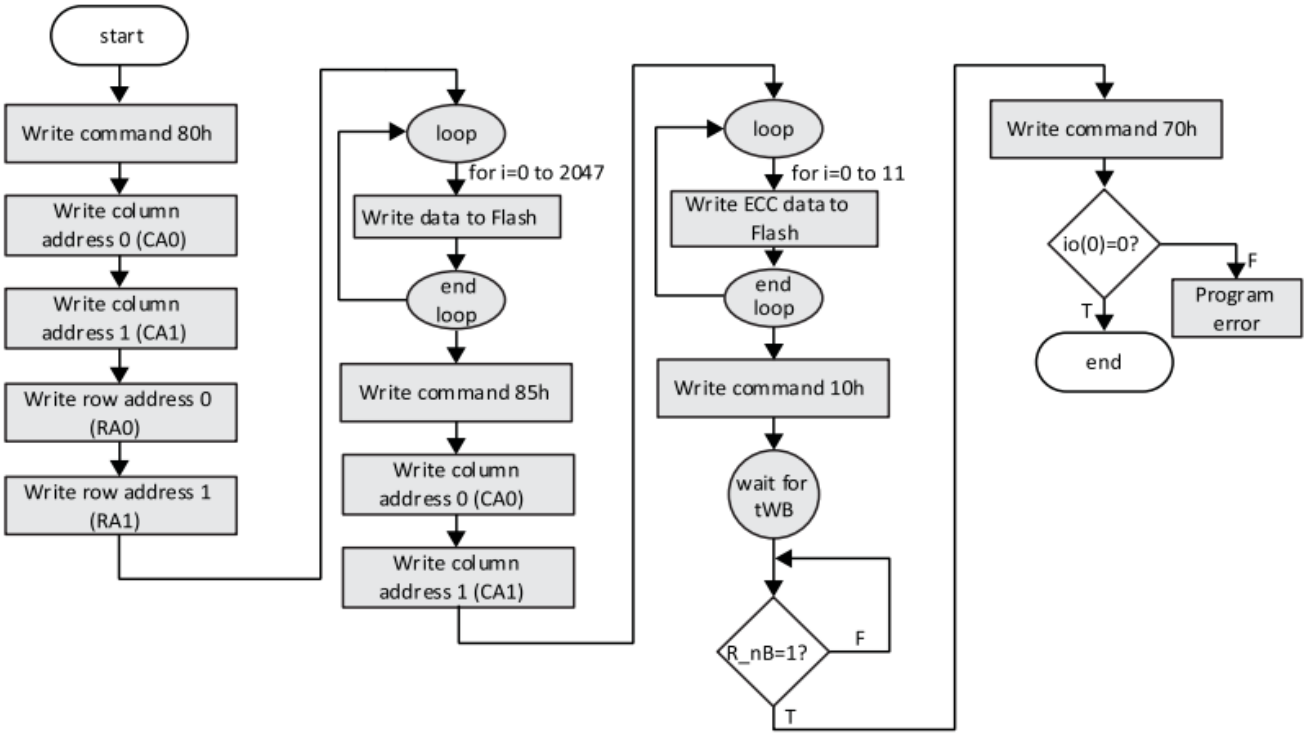

Figure 9: Page Program Flow Chart

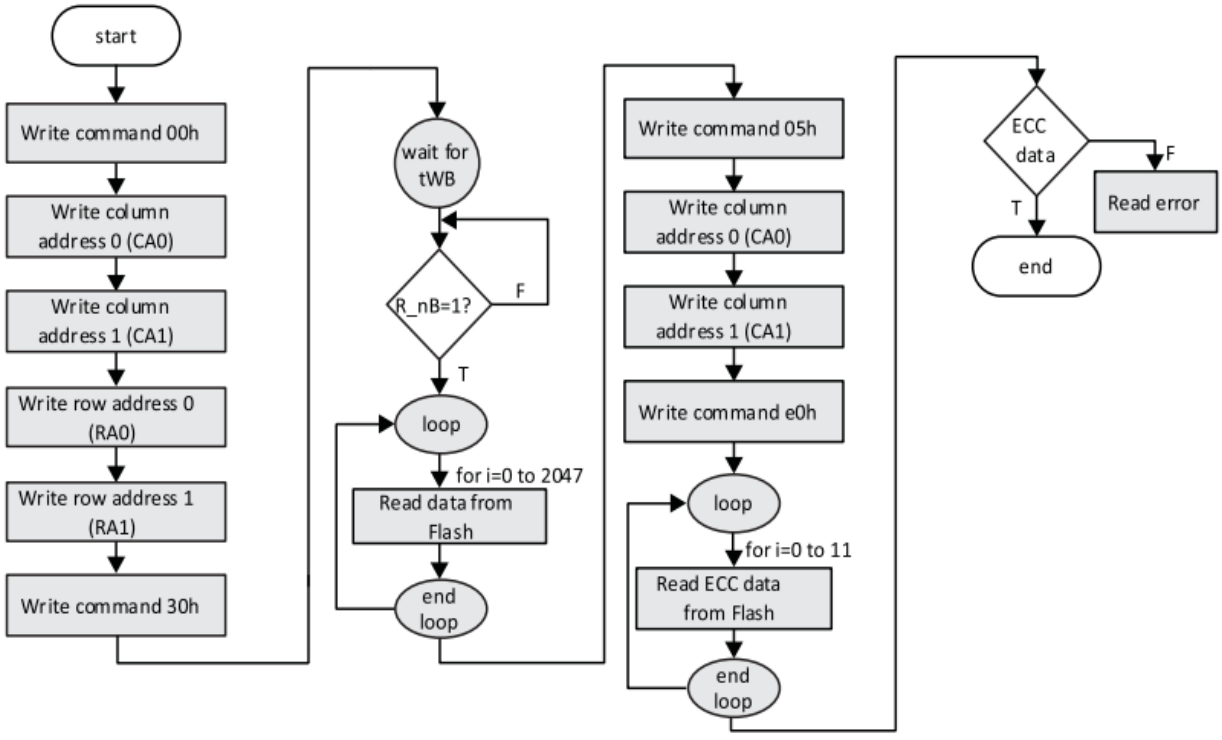

Figure 10: Page Read Flow Chart

ISBN: 978-988-14049-3-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

A. SV Verification for NFC Design

• NFC Top Module: The NFC top module contains the instantiation of the nfc_topmodule, the nfc_interface and nfc_testcase modules. The clock definition is also in the module and it also contains the dumpvars initialized to dump the waveforms of the NFC verification [6].

• NFC Test Case Module: The nfc_testcase module is below the nfc_toplevel module in the verification hierarchy. The testcase instantiates the nfc_environment module which is next in the hierarchy. The entire run of the verification starts with the run() command in the nfc_testbench module. The run() command triggers the verification process. The module also defines all the user defined host signals like BF_sel, BF_ad, BF_din, BF_we etc.

• NFC Environment Module: The nfc_environment module is present below the nfc_testcase module in the verification hierarchy. The nfc_environment module has scoreboard,driver and receiver are built in it and ran in the run_phase. The environment is responsible in connecting different agents,scoreboards and receivers.

• NFC Packet Module: The nfc_packet module is a module that creates data packets and constrains the data which are further used as inputs to verify the design. The data packets are sent to the driver which is further sent to the DUT. The data or stimulus is randomly constrained to ensure all the corner cases are covered. In the nfc_packet module the nfc_cmd is constrained to ensure that all the 5 different operations like read id, reset, block erase, page read and page write are covered. The ecc codes are also generated which are given as inputs to check for the success of operations like block erase, page read and write.

• NFC Driver Module: The nfc_driver module is the module that channels the stimulus generated from nfc_packet module to the DUT (NFC).The driver has tasks defined for each operation, hence the driver triggers the different operations based on the commands sent by the nfc_packet module.

• NFC Receiver Module: The nfc_receiver is a module that duplicates the operation of the Nand Flash device.The nfc_receiver module checks on the command received and performs the particular operation. For instance in case of a block erase operation it receives a command 100, followed by which the receiver module keeps track of the address and erases the block indicated by the particular address sent from the host device. Further waits for a time twb and sends a 0 on DIO pin to assure that the block has been erased and the receiver is ready to carry out the next operation. Hence the nfc_receiver module contains task for each operation, tracks addresses, reads and writes data depending on the operation.

• NFC Scoreboard Module: The nfc_scoreboard is the most important module as it verifies if the design works as expected by keeping track of the output of each operation and comparing it with the expected result. The module keeps track of data, command and addresses sent from the nfc_driver module and the operations carried out by the receiver for the respective commands sent from the driver. By comparing each output obtained from the nfc_receiver to the golden reference or expected output the scoreboard displays if the DUT functionality is a success or a failure. The scoreboard contains various display commands to show the success and failure of each operation carried out by the DUT.

• NFC Interface Module: The nfc_interface modules is a container of all the signals of the nfc_driver and nfc_receiver module. The module contains the mod port that has the directions of each signal of both the sides which is the driver and receiver module.

• NFC Coverage Module: The nfc_coverage module is a module that does the functional coverage of the NFC design(DUT). The module contains the coverpoints and a task that checks for the functionality of the design by checking if all the expected and the corner cases. All the output data is collected and analyzed to check for the functionality of the DUT and provides a file thst reflects the percentage of coverage of the design,with a coverage of 90% and above shows that a test verification is a good verification environment and covers almost all corner cases of the DUT.

B. Simulation and Waveforms

The output of the verification of NFC is obtained in form of waveforms and simulations which ensures the functionality of the design and gives a visual clarity to analyze, debug and understand the NFC design better. The following waveforms and simulations represent all the 5 operations from reset to page write operation done by the NFC and the functional coverage results show that the verification environment is successful in covering almost all the corner cases of the design making it a productive and reliable test environment.

• Reset operation waveform and simulation:

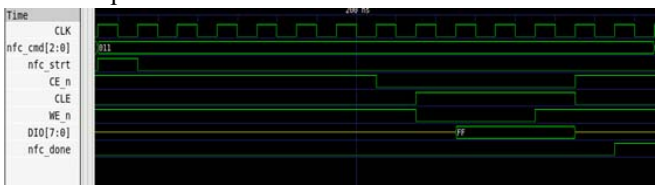

Figure 11: Reset operation waveform

• Read ID operation waveform and simulation:

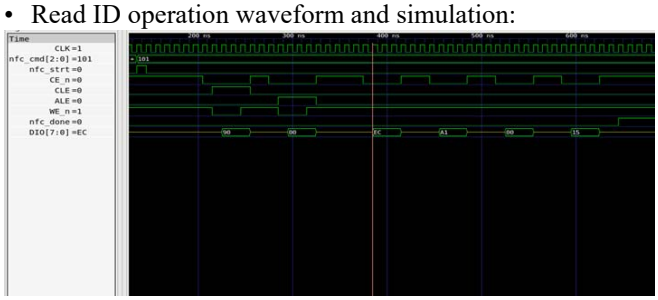

Figure 12: Read ID operation waveform

ISBN: 978-988-14049-3-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

- Block Erase operation waveform and simulation:

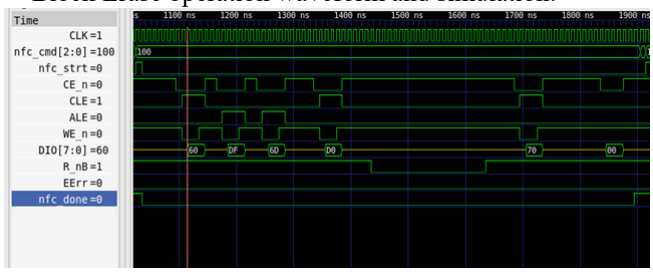

Figure 13: Block Erase operation waveform

- Page Program operation waveform and simulation:

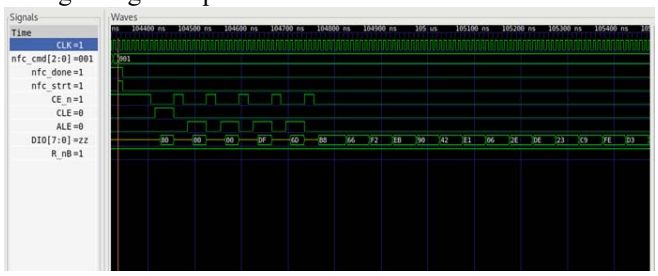

Figure 14: Page Program operation waveform

- Page Read operation waveform and simulation:

Figure 15: Page Read Operation waveform

III. Conclusion

Writing structured test benches takes considerable human effort upfront. However, once it is developed, it becomes easy to re-use it across the project or platform. Also, when a large team of people is working on the same project of Design Verification, the modularity that this kind of test benches provides becomes very vital. Once the initial development phase is passed, the work becomes very easy for the engineers working on the verification part. As this kind of test benches allows the engineers save time and just focus on the stimulus generation part. So far, we could successfully create the components required for a typical SV based test bench to complete the verification environment for NAND architecture based Flash Controller and observe the features and functionalities it provides using the Waveforms. And to achieve this, we entirely used the black-box verification approach.

References

[1] A. Seiichi, “NAND Flash Memory Revolution”, Proceedings of 8th International Memory Workshop (IMW), IEEE, 2016

[2] K. Yu-Hsiang, H. Juinn-Dar, “High-performance NAND flash controller exploiting parallel out of order command execution”, Proceedings of 2010 International Symposium on VLSI Design, Automation and Test, IEEE, 2010.

[3] M. Purvi, “SoC Level Verification Using System Verilog”, Proceedings of 2009 Second International Conference on Emerging Trends in Engineering and Technology, IEEE, 2009

[4] X. Gong et.al, “Design and implementation of a NAND Flash controller in SoC”, Proceedings of IEEE Conference on Electron Devices and Solid-State Circuits, IEEE, 2011

[5] A. Koushel et.al, “Verification and Simulation of New Designed NAND Flash Memory Controller”, Proceedings of IEE International Conference on Communication Systems and Network Technologies, IEEE, 2013

ISBN: 978-988-14049-3-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)