Accelerated Verification of NAND Flash Memory using HW Emulator

Seyeol Yang, Byungwoo Kang, Seoyeon Bae, Choi Jaehyeon, Jintae Kim, Dongeun Lee, Junho Ahn
Samsung Electronics, Hwaseong-si, Gyeonggi-do, Korea
sy101.yang@samsung.com, bw88.kang@samsung.com, seoyeon.bae@samsung.com, jae1313.choi@samsung.com, jtae47.kim@samsung.com, dongeun1.lee@samsung.com, jh0119.an@samsung.com

***Abstract-* In NAND Flash Memory, logic design becomes more complex as generations go by, and verification test cases are also increasing. In order to overcome the above situation, a verification methodology using an emulator rather than a verilog simulator is needed. The logic design of NAND Flash Memory is not composed of RTL only, but also includes analog behavioral model and bidirectional Transistors. In addition, NAND Flash Memory also needs a core model that is made for checking cell's operation. In this paper, emulation using Cadence's PZ2 emulator is proposed in NAND Flash Memory designed in various design types such as transistors, gates, and analog behavior models. In order to improve emulation speed, testbench optimization, interface signal minimization between testbench and DUT and transaction-based acceleration (TBA) were applied to a previous simulation environment. As a result, the accelerated emulator improves the verification speed by more than 100X compared to the Verilog simulator. Using the emulator verification environment enables verification of many test cases in a limited verification period, and the NAND Flash Memory operation at the system level is also verified. It will be the best solution for the Flash Memory verification process.**

Keywords- Emulator, Virtual Emulation, Verification, Transaction Based Acceleration, Transistor Level Design, Flash

I. Introduction

Recently, features required for NAND Flash Memory are increasing, while the period of development and verification has not changed much. Whenever a new algorithm is developed, number of verification test cases is exponentially increasing. In a given verification period, it has become hardly to verify all test cases because of limited simulation resources and a few test cases that require very long simulation time. We adopted HW emulation not only to reduce regression turn-around time but to verify more complicate behaviors of NAND Flash Memory. But our device under test(DUT) was consist of various design style, i.e. RTL, gates, transistors, and analog behavioral model. In this paper, we introduce what were difficult points to setup HW emulation environment for out designs and how we solved it.

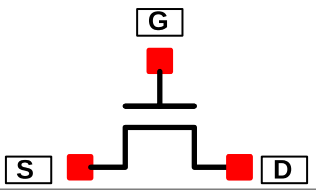
II. Emulator Setup for Flash Memory

1. Synthesizable Model Development of various design style

NAND Flash memory is not only designed with RTL, but consists of transistor and analog behavior model that are not suitable for emulation synthesis. Therefore it is necessary to change them to synthesizable components and check final emulation correctly operating.

1-1. Change bidirectional transistors to unidirectional

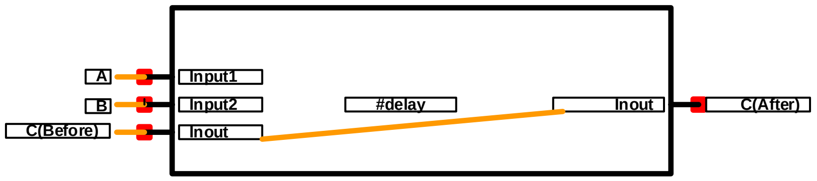
Because tranif0 and tranif1 are not supported in Emulator, the direction must be changed correctly to use NMOS and PMOS. For example, As shown in Fig 1, transistors using tranif1 do not need to distinguish between source and drain. Most transistor level designs are likely to be designed without considering the part, so it is necessary to check the part and modify it to NMOS and PMOS according to the correct direction.



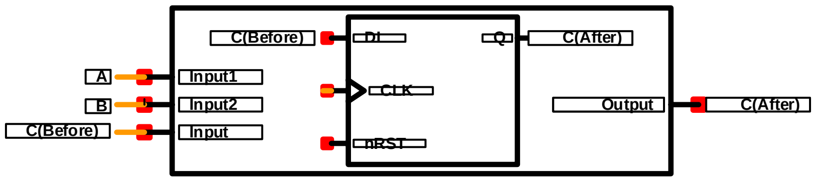
	tranif1,tranif0	nmos,pmos
form	(inout1,inout2,control)	(out,in,gate)

Figure 1. Transistor Verilog form

1-2. Change some bidirectional ports to unidirectional
Emulator cannot handle # delay in verilog language. In case of bidirectional ports controlled by # delay, these ports must be separated. For example, in some analog units, feedback loops can be formed by inout port, which will be shown in Fig. 2 below. In simulation, the precedence relationship is clear due to #delay, but in emulator, it will be synthesis without delay, which will cause malfunction. As shown in the example, it can be solved by D-FF.



Simulator Netlist



Emulator Netlist

Figure 2. How to change using bidirectional port

1-3. Create or change synthesizable code for Analog Unit
Many non-synthesizable behavioral model i.e. Delay Unit, Pump, Voltage Generator, are used in simulation. For example, delay unit (delay 10ns) will be described as follows, but it cannot be operated on emulator, therefore, the delay operation was performed using the counter. (Fig. 3)

```
Module DELAY10(Out,in);
   Input In;
   Output Out;
      not(#0.15,9.7)INV0 (INV0Out,In);
      rnmos #(0.15,0.15)N5(N5D,gnd,In);
      not(#0.15,9.7)INV0 (INV1Out, INV0Out);
      not(#0.15,9.7)INV0 (ND2_1Out,INV1Out);
      rnmos #(0.15,0.15)N5(INV0OutNn5D,INV2Out);
      pmos #(0.15,0.15)N5(INV0Out,VDD,In);
      nand(#0.15,9.7)INV0 (Out,In,INV1Out);
      not(#0.15,9.7)INV0 (INV20Out,ND2_1Out);
endmodule
```

(a)

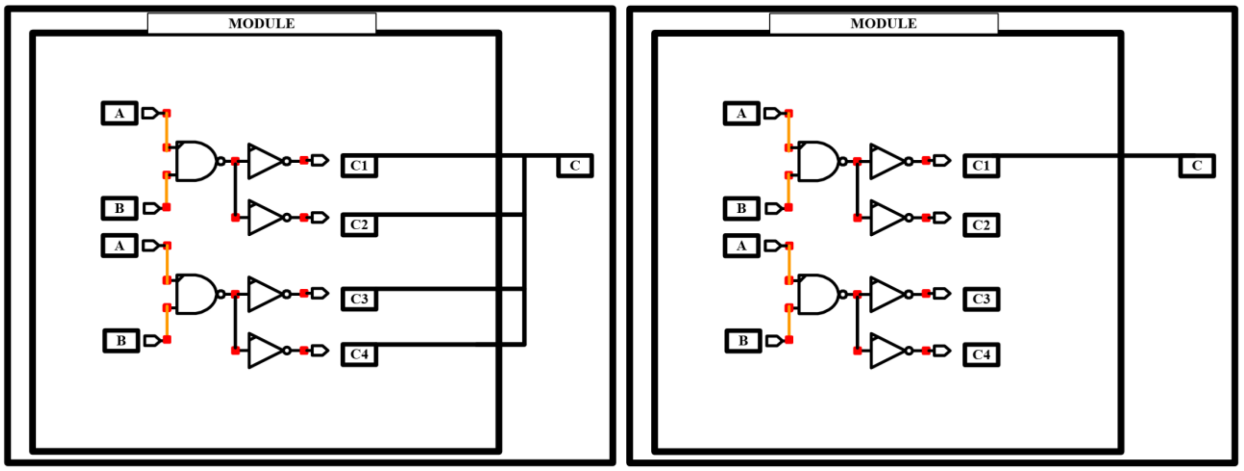
```
Module DELAY_MODEL(Out, In);
   Input In; Output out; Parameter delay =0;
   wire iOSC = CLK10;
      always@(posedge iosc or negedge In) begin
         if(In) cnt[2] <=cnt[1];
         else cnt[2] <=1; end
      always@(posedge iosc or negedge In) begin
         if(In) cnt[1] <=cnt[0];
         else cnt[1] <=1; end
      always@(posedge iosc or negedge In) begin
         if(In) cnt[0] <=1'b0;
         else cnt[0] <=1;
         end
      assign out = (delay ==20) ?cnt[2]:
                        (delay ==10) ?cnt[1]:
                        1'b0;
endmodule
```

(b)

Figure 3. (a) Example of non-synthesizable code, (b) Converted synthesizable code

1-4. Remove multi-driving signal

Some multi-driving signals caused malfunctions in the emulator. In order to increase the driving ability when designing, multiple outputs can be designed to drive the same signal. The emulator does not support multi-driving, so only one output of the circuits must be connected or Or Gate must be connected at the higher level. (Fig. 4)



(a) **(b)**

Figure 4. (a) Example Multi-driving Netlist, (b) Converted Netlist

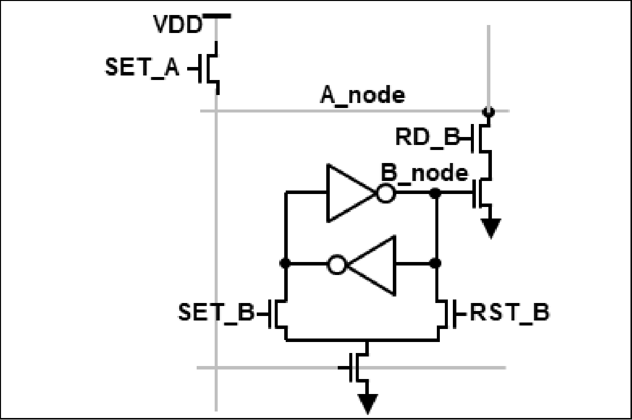
2. Core operation Behavior Model

NAND Flash Core is modeled to describe the behavior of the data stored and read and also to determine whether the nand-flash operation is normal. It is composed of Memory Cell and data latch that is designed with several non-synthesizable 6T SRAM structures per bit line. Memory Cell modeling has a problem with synthesizing when implementing all word lines and bit lines because it needs more than 100G bytes registers. In addition to describe the nand flash operation, the Vth of each cell was modeled.

- Data latch modeling

  Fig. 5 is a circuit that single page buffer and its modeled Verilog code.

  This is a simple method that can be simulated that page buffer, but it is not compatible for synthesis



```
always @(SET_A) begin
  A_node <= 1 ;
end
always @(RD_B) begin
  A_node <= A_node & (~B_node) ;
end
always @(SET_B) begin
  B_node <= A_node | B_node ;
end
always @(RST_B) begin
  B_node <= (~A_node) | B_node ;
end
```

(a) **(b)**

Figure 5. (a) Page buffer circuit, (b) Non-synthesizable code of page buffer

A new Scheduler Module was designed to implement the asynchronous operation of data latches, which made it synthesizable. (Fig 6.)

The Synchronizer in Scheduler Module synchronizes the asynchronous the control signals such as SO_PRECHARGE, MON_S, SET_S, RESET_S to Clock/Reset.

In addition, Re-timer make the synchronizer output signals mutually exclusive.

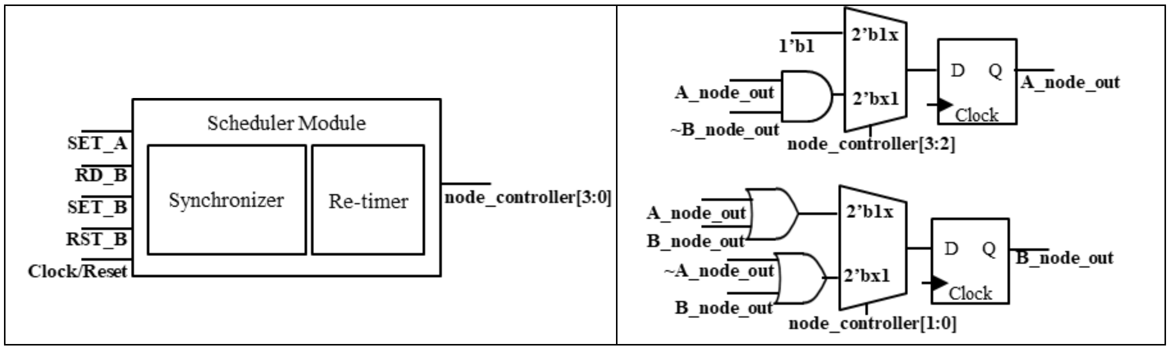


Figure 5. A New Scheduler Module.

- Memory cell modeling:.

  The storage space required to model Memory cell is number of Blocks x Word line x String x Bit line (bits).
  The logic size is reduced by using RAM provided by emulator instead of synthesizing memory cells with gates and flip-flops.
  Memory cells are programed the data from the page buffer by peripheral, and on the contrary, the Read Command transfers data from the memory cell to the page buffer.
  The stored value in Memory cell model indicates programed cell with 0 and erased cell with 1.

- Vth modeling:

  Nand flash memory is a nonvolatile data storage that stores information by varying the Vth of floating-gate transistor device [5].
  One dimension with 16bits was added to the Memory Cell Model, and the Vth value of each memory cell was modeled in order to verify detailed memory cell behaviors.
  However, the Vth Cell Model that is implemented is too large to synthesize from emulator to logic, so following methods are applied
  Fig. 7 shows how to operate the Vth Cell Model.
  The left is the part that is operated in the hardware and the right side is operated in the software.
  Erase/Program/Read Processor calculates the Vth value and reads/writes it to Vth Cell Model pointed to the index [N].

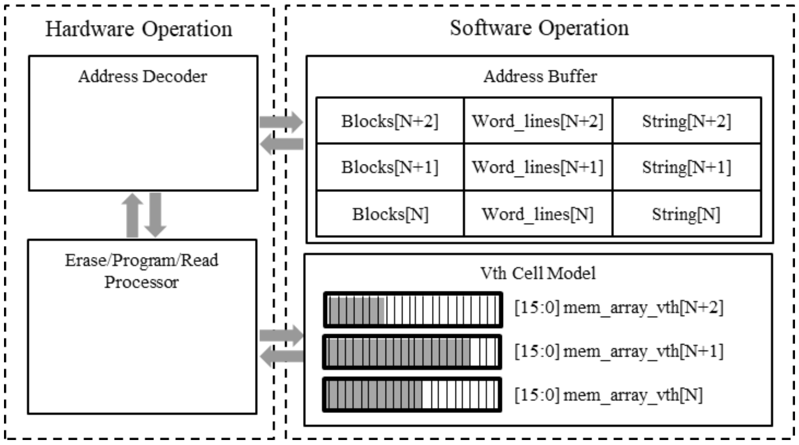


Figure 6. Vth modeling implementation method.

3. Emulation performance improvement skills.

After making a synthesizable model using above 1 and 2 steps, the performance of emulation is improved by more than five times compared to the simulation. In order to maximize the performance of emulation, several skills were adopted.

3-1. Ports Reduction

The simulator performs signal communication with each other using the DUT environment and interface module in the SW environment.(Fig. 8) However, for verification using an emulator, communication connection between the UVM interface and the emulator interface, that is, interface connection between SW and HW, is required. (Fig. 9)

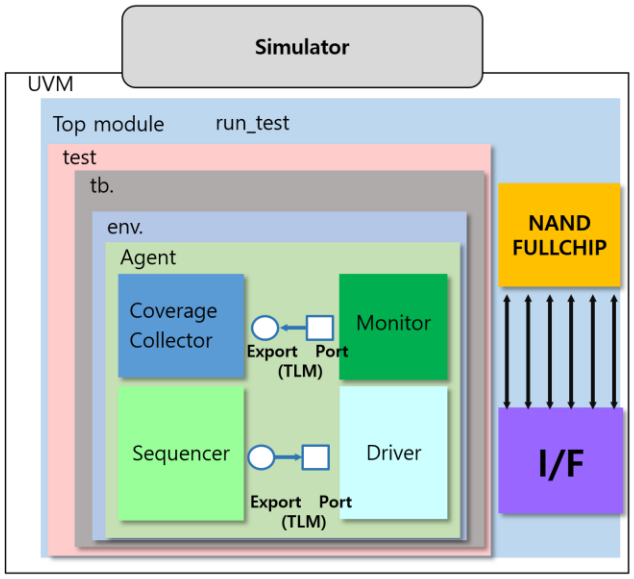


Figure 7. Simulator Test Environment

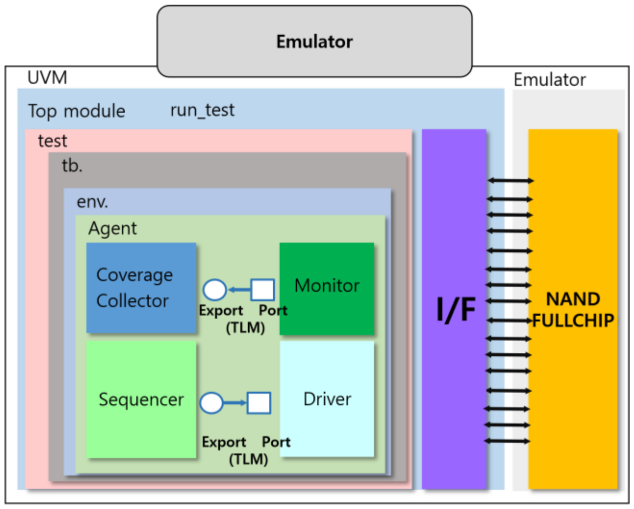


Figure 8. Emulator Test Environment

NAND Flash basically has pins such as IO, ALE, and CLE, and uses them to recognize CMD and operate like Transaction Level. However, NAND Flash has numerous operation options, so a lot of ports had to be connected between the simulator and the emulator for verification. And if the assertion is declared to the UVM, the port will be required for checker. But, this is the one of reasons to slow down emulation performance. When communicating between the simulator and the emulator, the status of all ports is transmitted after storing them in the temporary memory to check the state of all connected ports whenever one port moves. Therefore, the signal related to the operation option has been changed to change the forcing in the corresponding node when performing the emulation, and the assertion can be described and used inside the emulator DUT, the necessary signals were changed so that they could be synthesized in the top module. As a result, more than five times the runtime has been improved by optimizing the number of ports.

3-2. Direct register (memory) access

The I/O Datapath consists of buffers and latches that are transferred to the register according to timing. It is more accurate to verify this operation with a simulation that reflects timing than to verify it in the emulator. To reduce communication between simulator and emulator, the direct register access method is used for the WRITE/READ operation instead of data in/out behavior using I/O ports. (Fig 9.)

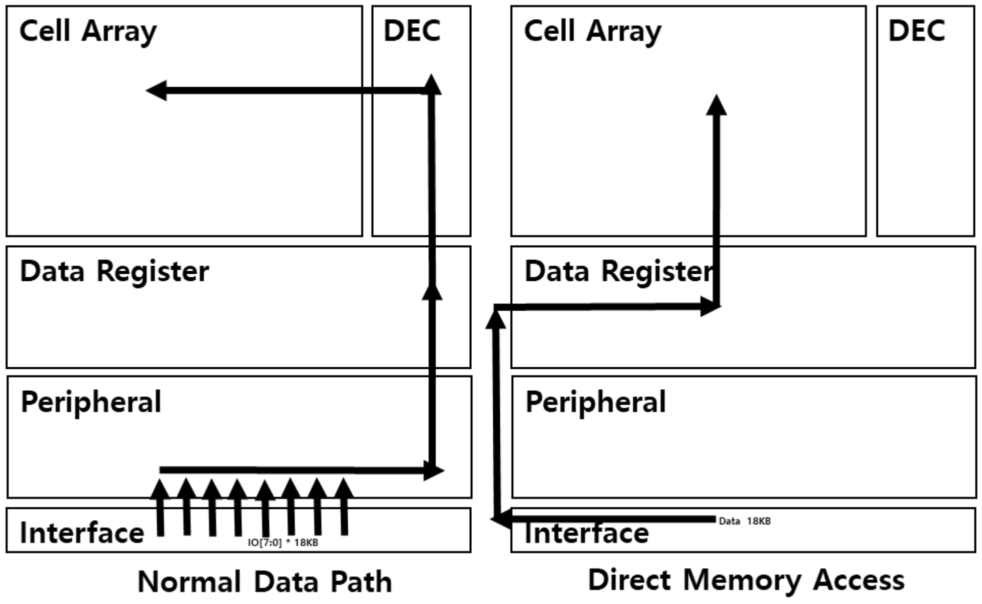


Figure 9. Direct register (memory) access.

And we used cell presets for unnecessary duplicate verification. It has led to an effect of improving the emulator speed more than four times.

III. EXPERIMENTAL RESULT

To verify NAND Flash, millions of case verifications are required. To compare simulation and emulation speed, we selected 5,000 test cases and summarized each runtime as follows, including the application of the method to increase performance described above. Also, it has a similar effect to interface optimization by assertion checker movement. (Table 1.)

Table 1.
Run Time Comparison (Total case: 5000)

	Verilog Simulation	Emulation	
Data Size 18KB (using I/O interface)	**3751h**	**749h**	**X5**
Using Memory Direct Access		**186h**	**X20.2**
Using Memory Direct Access, +Interface optimization		**42h**	**X89.3**
Using Memory Direct Access +Interface optimization +Assertion Checker move		**37h**	**X101.4**

Table 2. is summarized emulation performance compared to Simulation. Because design changes and modifications were frequently occurred in the development stage, compile time was also one of important factors in this evaluation.

Table 2.
Performance Comparison (Total case: 5000)

	Verilog Simulation	**Emulator Simulation**
Compile Time	**0.3h**	**2.5h**
Run time	**3751h**	**37h**
Dump Time	**5179h**	**283h**

Although emulation's compile time is about eight times slower than simulation, run time and dump time of emulation are 101X and 18X faster, respectively.

IV. SUMMARY

By configuring the Emulation verification environment, it is possible to verify the more complex Flash memory behaviors within the same period. It is thought to be very useful verification methodology as a replacement means of function verification and long-time simulation. As long time simulation is possible through the emulator verification environment, it is expected that NAND operation at the system level can be verified in advance.

References

[1] Lee DongChan, Yu minsu "Transaction-Based Simulation Acceleration for SSD Controller Using HW Emulator", Samsung, 2021

[2] Cadence Emulator Manual. (vxeUser Guide)

[3] Cadence UVMA Manual.

[4] Nayoung Choi, et al., "Modeling and simulation of NAND flash memory sensing systems with cell-to-cell V_{th} variations," ICCAD: Proceedings of the 39th International Conference on Computer-Aided Design November 2020 Article No.: 52 Pages 1–8, Dec. 2020.

[5] K. Parat and A. Goda, "Scaling trends in NAND flash," in Proc. of IEEE Int'l Electron Devices Meeting (IEDM), pp. 2.1.1- 2.1.4, Dec. 2018.