

ECE-GY 6442 – VLSI System and Architecture Design

Project 3: SRAM MBIST

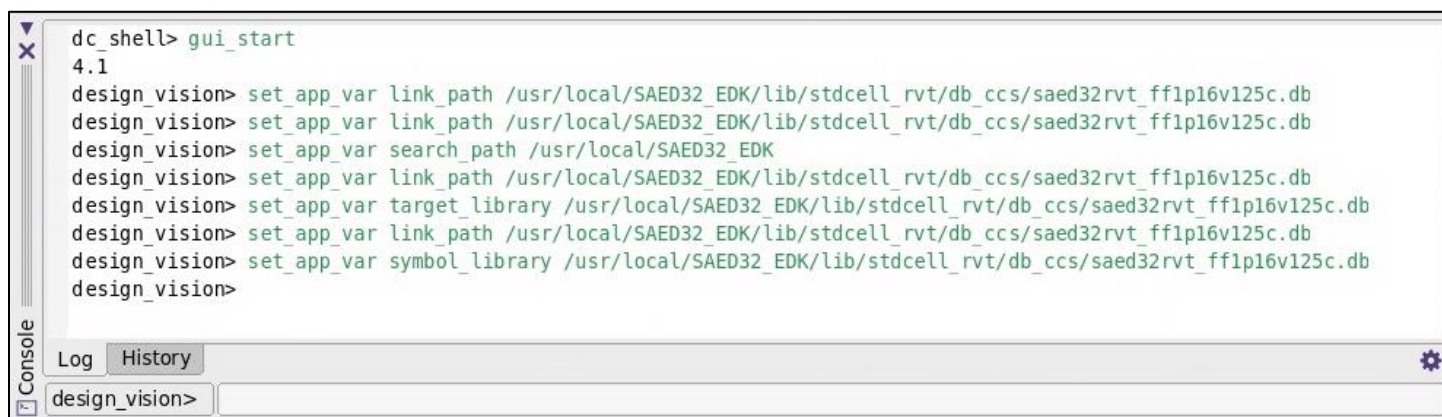
Due May 13th, 2022

Md Raz
N17762874
mr4425@nyu.edu

Siddharth Kandpal
N10799721
sk8944@nyu.edu

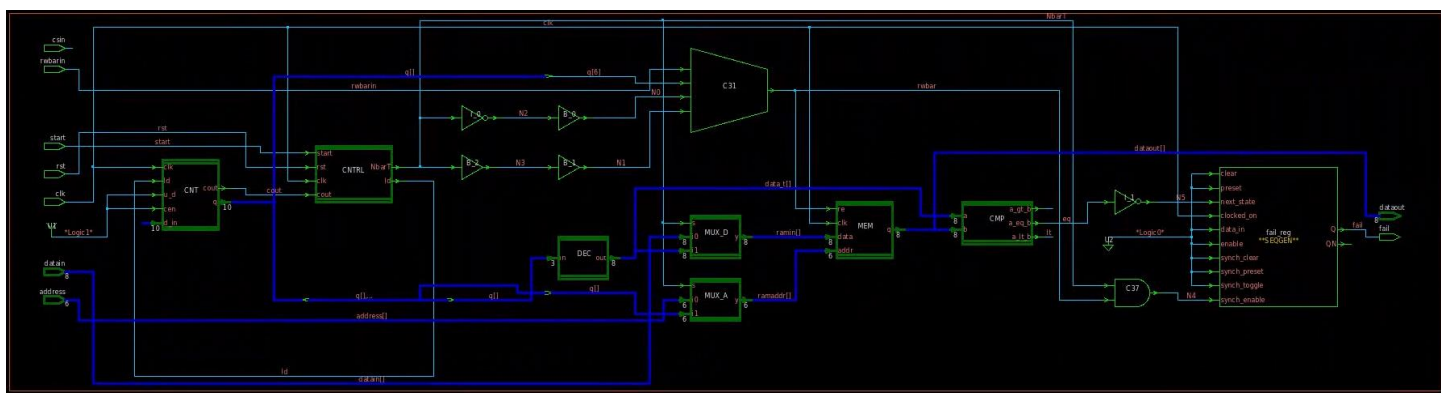
1. Synthesizing the BIST

Synopsis Design Vision was used to synthesize the BIST. After opening the GUI of Design Vision, the software environment was set up to use the SAED32 EDK cell library, which is the generic 32 nm technology node library provided by Synopsys for evaluation. The search path, link path, target library, and symbol library variables were changed to point to this 32nm library, as shown below.



```

dc_shell> gui_start
4.1
design_vision> set_app_var link_path /usr/local/SAED32_EDK/lib/stdcell_rvt/db_ccs/saed32rvt_ff1p16v125c.db
design_vision> set_app_var link_path /usr/local/SAED32_EDK/lib/stdcell_rvt/db_ccs/saed32rvt_ff1p16v125c.db
design_vision> set_app_var search_path /usr/local/SAED32_EDK
design_vision> set_app_var link_path /usr/local/SAED32_EDK/lib/stdcell_rvt/db_ccs/saed32rvt_ff1p16v125c.db
design_vision> set_app_var target_library /usr/local/SAED32_EDK/lib/stdcell_rvt/db_ccs/saed32rvt_ff1p16v125c.db
design_vision> set_app_var link_path /usr/local/SAED32_EDK/lib/stdcell_rvt/db_ccs/saed32rvt_ff1p16v125c.db
design_vision> set_app_var symbol_library /usr/local/SAED32_EDK/lib/stdcell_rvt/db_ccs/saed32rvt_ff1p16v125c.db
design_vision>
  
```

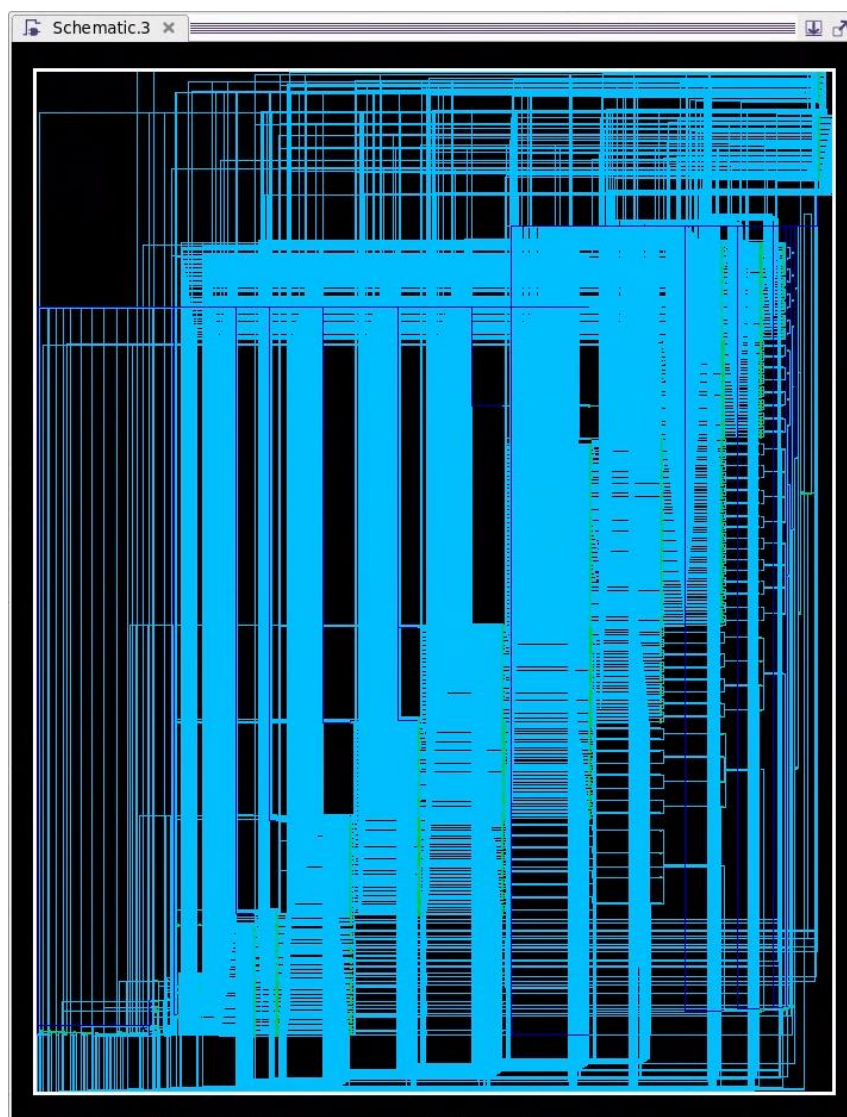


Next, the Built-In-Self-Test (BIST) top module file, named BIST.v, was read into the program. Before compiling the BIST, we are able to obtain a preliminary circuit schematic, as shown above. This signifies that the BIST, along with all the internal modules, were read properly and the Verilog was correctly instantiated. During the reading of the Verilog file, Design Vision was able to infer 19 flip-flops for memory such as counters and the test-fail register, and 512 flip-flops for the integrated RAM. The BIST was then set as the current design and was linked to the SAED32 EDK cell library.

Compilation of the BIST will be done in two ways, the first being the “Compile Design” option, which compiles and maps the design, with medium considerations to mapping, area, and power efforts. The second compilation option, which is the “Compile Ultra” option, will compile the BIST more thoroughly and rigorously, while also paying mind to the provided synopsis design constrains file, which indicates the desired timing constraint for the BIST. The “Compile Design” option was set using the default values and completed successfully, which returned an optimized area usage of 6638.2 units, as shown below. Afterwards, the “Compile Ultra” compilation option was executed and returned an area of 6296.2 units, as shown below. For both of the compilation options, there were not timing violations within the BIST. Through multiple iterations of optimizations, the “Design Ultra” option was able to decrease both the area consumption and

leakage power of the BIST module. Between the normal and ultra design compilations, the area was reduced by about 400 units. The schematic for the synthesized BIST was then obtained, as shown below.

	Elapsed Time	Area	Worst Neg Slack	Total Cost Setup	Design Rule Cost	Endpoint	Leakage Power
Compile Design	0:00:04	6638.2	0.00	0.0	0.0	N/A	N/A
Compile Ultra	0:00:07	6296.2	0.00	0.0	0.0	N/A	154318752.00



Next, the timing reports were generated and checked for both the Maximum timing, which signifies the setup time violations, and the minimum timing, which signifies hold time violations. The reports are given in their entirety in the table below.

For the Maximum Delay type timing report, the timing path starting point was taken from the node for the address register within the RAM memory instantiation, and the end point was taken as the node of the output port. Between the start and end point, the signal travels through several logic gates and utilizes a total of 0.29 ns. The minimum delay time was found between the

starting point of the fail register, and an end point of the fail output port, where it travels through a single flip flop and consumes a total time of 0.06 ns.

BIST Timing Report, Maximum	BIST Timing Report, Minimum
<pre>***** Report : timing -path full -delay max -max_paths 1 -sort_by group Design : BIST Version: R-2020.09-SP5 Date : Sat May 7 12:53:37 2022 ***** Operating Conditions: ff1p16v125c Library: saed32rvt_ff1p16v125c Wire Load Model Mode: enclosed Startpoint: MEM/addr_reg_reg[3] (rising edge-triggered flip-flop) Endpoint: dataout[2] (output port) Path Group: (none) Path Type: max Des/Clust/Port Wire Load Model Library ----- BIST 8000 saed32rvt_ff1p16v125c Point Incr Path ----- MEM/addr_reg_reg[3]/CLK (DFFX1_RVT) 0.00 0.00 r MEM/addr_reg_reg[3]/Q (DFFX1_RVT) 0.07 0.07 f U1188/Y (AND2X1_RVT) 0.03 0.09 f U1167/Y (AND3X1_RVT) 0.08 0.17 f U1528/Y (AO22X1_RVT) 0.03 0.20 f U1532/Y (NOR4X1_RVT) 0.04 0.24 r U1538/Y (AO21X1_RVT) 0.03 0.26 r U1550/Y (NAND4X0_RVT) 0.03 0.29 f dataout[2] (out) 0.00 0.29 f data arrival time 0.29</pre>	<pre>***** Report : timing -path full -delay min -max_paths 1 -sort_by group Design : BIST Version: R-2020.09-SP5 Date : Sat May 7 12:55:11 2022 ***** Operating Conditions: ff1p16v125c Library: saed32rvt_ff1p16v125c Wire Load Model Mode: enclosed Startpoint: fail_reg (rising edge-triggered flip-flop) Endpoint: fail (output port) Path Group: (none) Path Type: min Des/Clust/Port Wire Load Model Library ----- BIST 8000 saed32rvt_ff1p16v125c Point Incr Path ----- fail_reg/CLK (DFFX1_RVT) 0.00 0.00 r fail_reg/Q (DFFX1_RVT) 0.06 0.06 r fail (out) 0.00 0.06 r data arrival time 0.06</pre>

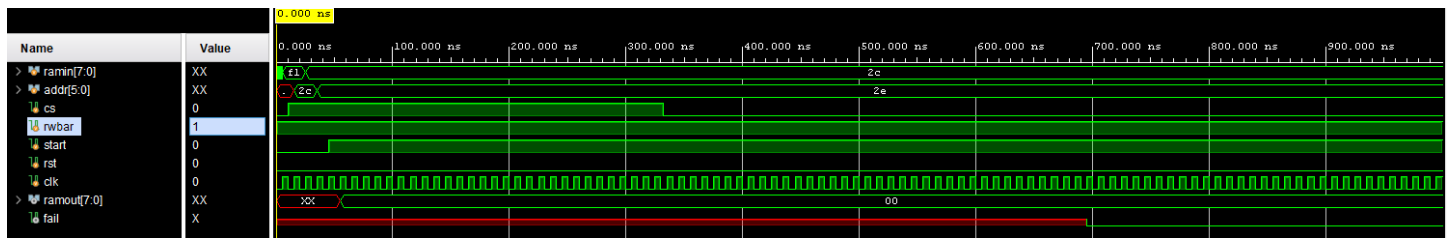
After completion of this segment, the BIST was tested with clock timing constraints. It was found that the minimum clock cycle constraint for which the BIST did not violate any data paths and the slack was met was at a clock constraint of 0.55 ns. This means that the clock has a period of 0.55 nanoseconds and a pulse width of 0.275. The following constraint SDC file parameters were used. Using this time constraint, both the maximum and minimum clock path reports were generated. The timing reports are included with the report within the 'Timing Reports' Folder.

```
create_clock -period 0.55 -name clk [get_ports clk]
set_input_delay 0.1 -clock clk [all_inputs]
set_output_delay 0.15 -clock clk [all_outputs]
set_load 0.1 [all_outputs]
set_max_fanout 1 [all_inputs]
set_fanout_load 8 [all_outputs]
set_clock_uncertainty .01 [all_clocks ]
set_clock_latency 0.01 -source [get_ports clk]
```

Since the SAED32 EDK cell library does not contain an SRAM cell, layout synthesis for this particular module is not possible. Due to this, the scope of this part was limited to the RTL synthesis of the BIST.

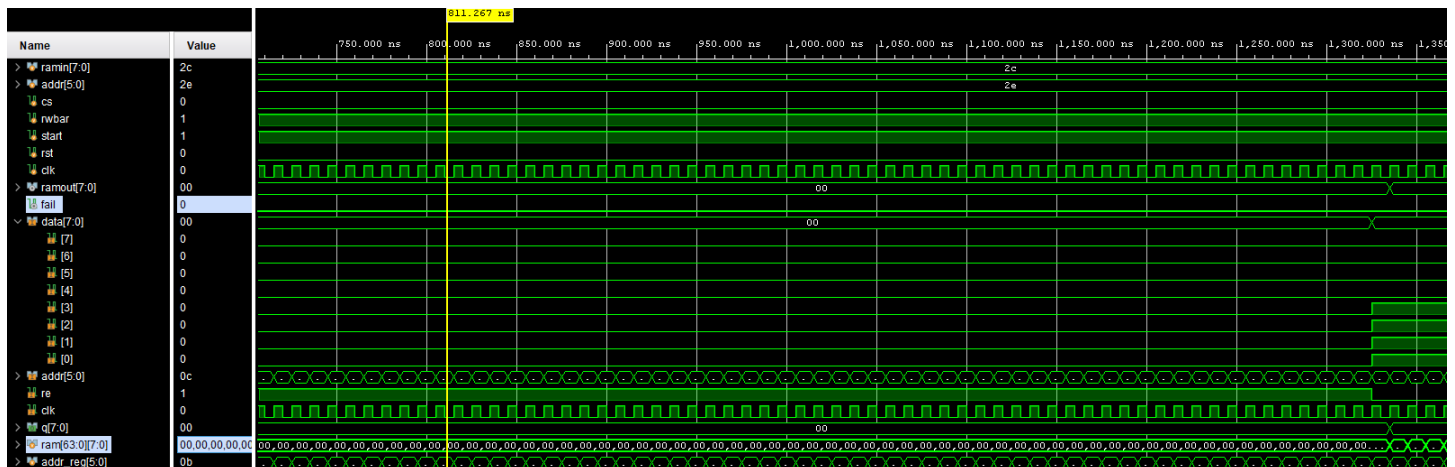
2. Verifying Logic

The provided testbench file modified to remove any errors and a waveform of the input and output data was captured.



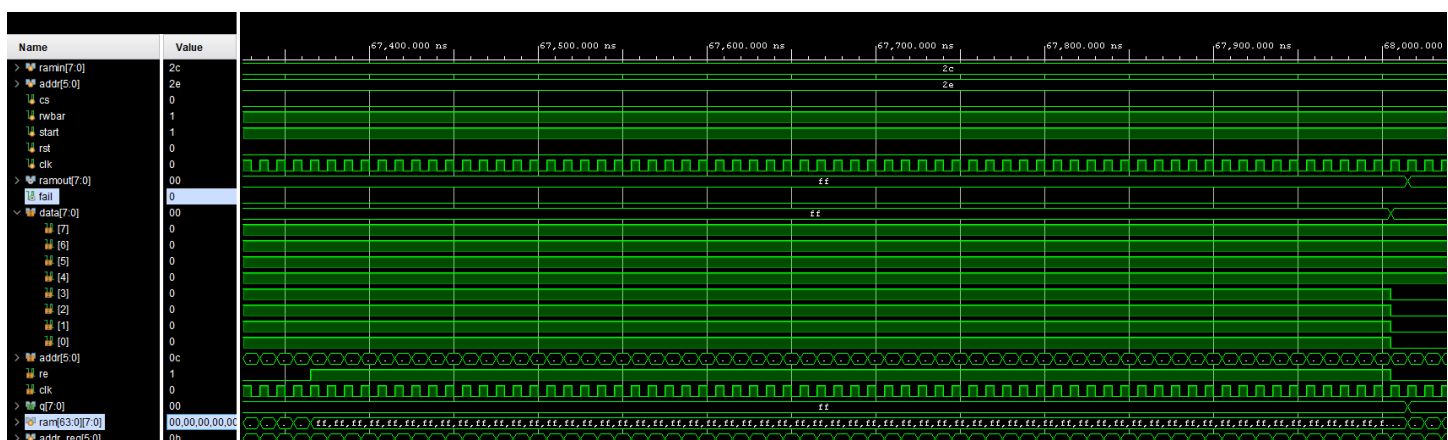
Once the Verilog test bench was confirmed working via the simulation, both the Verilog and test bench files for the BIST were modified. The following algorithms were tested within the BIST:

Blanket 0:



A wave of zeros was applied to the entirety of the RAM module and each time an address was written to, it was read from afterwards to ensure there were no erroneous read and write operations. It can be seen in the simulation screenshot that the wave output does not go to one signifying that our testbench was successful.

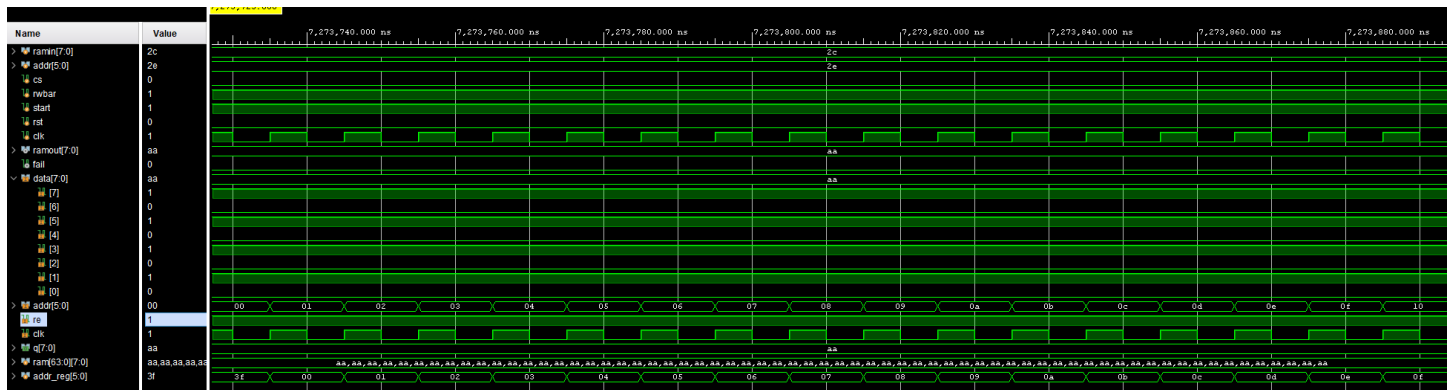
Blanket 1:



A wave of ones was applied to the entirety of the RAM module and each time an address was written to, it was read from afterwards to ensure there were no erroneous read and write operations.

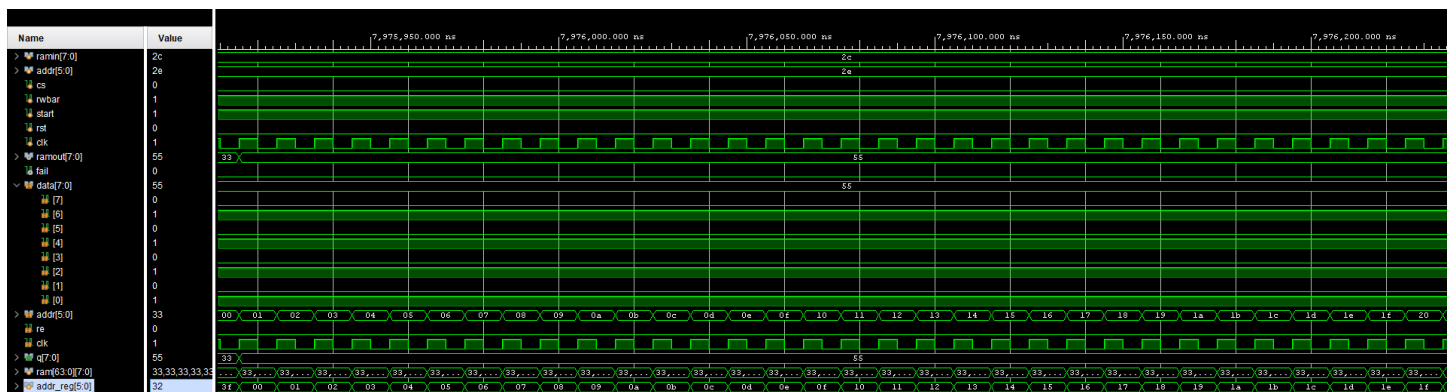
operations. It can be seen in the simulation screenshot that the wave output does not go to zero signifying that our testbench was successful.

March A: 0,1,0,1 with increasing address



Alternate zeroes and ones were applied to the RAM module to ensure its functionality. Inferring from the screenshot, we can conclude that our read and write operation is perfectly in sync with zeroes and ones which leaves no room for error, bolstering the validation of our testbench simulation results.

March C:



Zeroes and ones were written to arbitrary addresses and read from for the authentication of the testbench. On successful implementation, a plausible output was attained which hence verified our testbench functionality.