TFE 4152 Project Final Report

Ia Tsomaia

Saltuk Çelik

Norwegian University of Science and Technology

Abstract

This paper presents the 64-bit memory array project. The emphasis of the project is power consumption. We aimed to minimize static power dissipation with several design choices, which we will discuss in the following passages, alongside the reasons behind each decision.

We used Verilog to describe the hardware and simulate the design, checking the correctness. Furthermore, we constructed the circuit of the single bit, which we call bitcell, in AIM-Spice, to simulate the analogue circuit and observe the power dissipation.

As a result, we got a memory unit that stores and retrieves the valid data. Furthermore, as we designed the analogue circuit of the bitcell, we could scrutinize how our design choices affected the leakage current and in what circumstances it was the lowest.

We got the results we anticipated. Although, there are plenty of possibilities for further advancements. For instance, if we monitored the power consumption for the whole memory module, it would assist us in refining the design and further reducing the power consumption.

1 Introduction

Day by day, the topic of low power consumption gains more attention and popularity, as it is tightly linked to critical issues, like environmental or economic. The devices that do not need much power help us save money and, at the same time, allow us to reduce the negative effect on the environment.

Realizing the vitality of the matter, we attempted to design the memory unit with low power dissipation, specifically with low static power consumption, that is present due to the current leakage.

Our memory unit contains eight words, each word having a length of eight bits. The user should provide the address they want to store data in/read data from, the action (Read/Write) and the input bits in case of writing in the memory.

Regarding the project's focus, we made several design choices. Specifically, we minimized the number of transistors throughout the memory array, chose a specific gate width/length of the transistor, and adjusted the voltage.

2 Methods

2.1 Bitcell Design

The smallest unit of our memory array, the module that holds 1 bit of information, is the bitcell.

It consists of SR-latch, input logic for the latch's inputs and the transmission gate, as shown in figure 1.

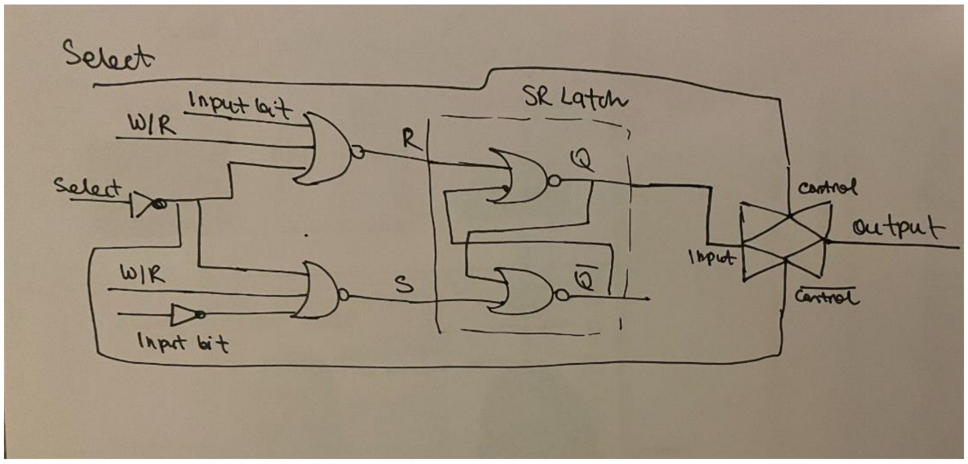


Figure 1: Bitcell Circuit Diagram

To get the input logic for the bitcell, first, we constructed the truth table shown in table 1 and converted it into two Boolean expressions. Then we modified the expressions to get the minimum number of transistors. We got the following logic for S and R: S = (A'+B+C')' (A- Select, B- W/R, C- Input) and R = (A'+B+C)'.

We use the transmission gate to get the output only when the select signal is high. The gate requires only two transistors, less than other gates we could use to get the desired result.

Select	W/R	Input	S	R
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	0	0
1	1	1	0	0

Table 1: Truth table for Latch's Inputs

2.2 Analog Circuit Design

In terms of analogue circuit simulations, to lower the static power dissipation, firstly, we attempted to reduce the subthreshold leakage, as it is one of the most significant sources of it. To decrease the Subthreshold leakage, we had to increase the threshold voltage. One way to increase the threshold voltage is by reducing the short-channel effects. To do so, we expanded the gate length as much as possible (300 nm).

Furthermore, we chose the width of 1500 nm for PMOS to have better transconductance and 500 nm for NMOS, three times less than NMOS because of the difference in mobility of carriers.

Finally, we set the supply voltage to 1V after trying different simulation values and comparing the dissipated static powers.

2.3 Word Design

We use bitcells to construct the words, larger units of memory. Each word, in our case, consists of eight bitcells. Figure 2 shows the circuit diagram of the word. Bitcells share RW and Select signals, but the input bits are different for each of them. In the 8-bit output of the word, each bit is an output of a different bitcell.

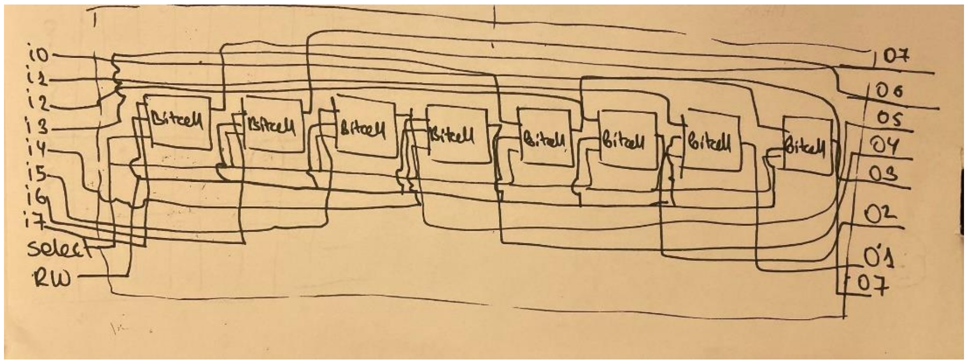


Figure 2: Word Circuit Diagram

2.4 Memory Array Design

The memory array contains eight words. Each word gets the same Read/Write and input bits, but the select signal differs. The Select signal depends on the address input. To get the optimal number of transistors for the input signal bits logic, we once again modified the Boolean expressions for each select signal. For example, for the select input of "word 0", we use the Boolean expression: s0 = (a+b+c)' (a = address [2], b = address [1] c = address [0]). To construct it, we only need six transistors. Figure 3 shows the whole logic circuit diagram of the memory.

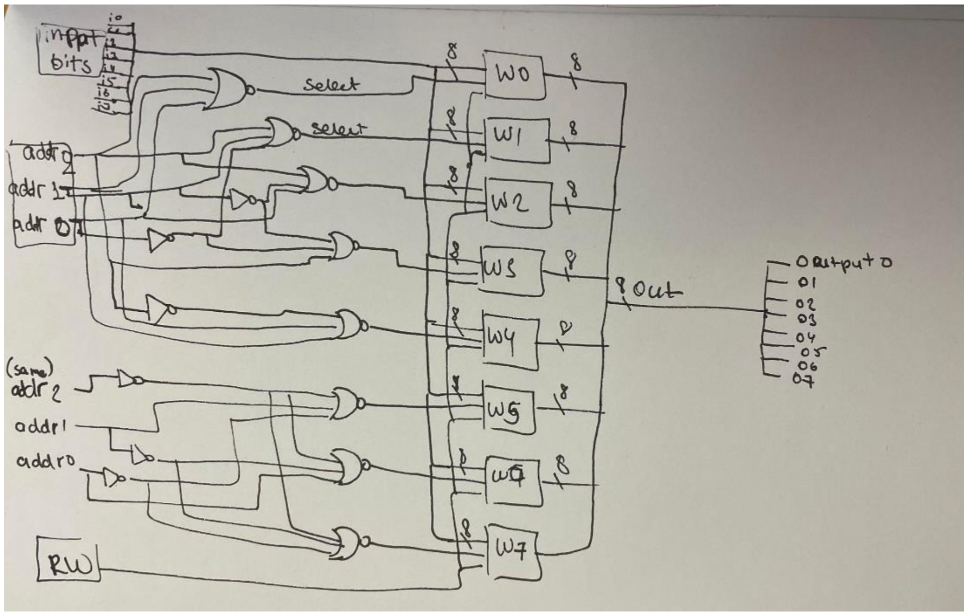


Figure 3: Memory Circuit Diagram

2.5 FSM Design

The main objective of the FSM is to make sure that racing conditions do not occur. The effect of input signals of the memory having different delays is explained in the memory part of Verilog simulation results section. To get rid of this issue, FSM returns to sleep state after reading and writing states.

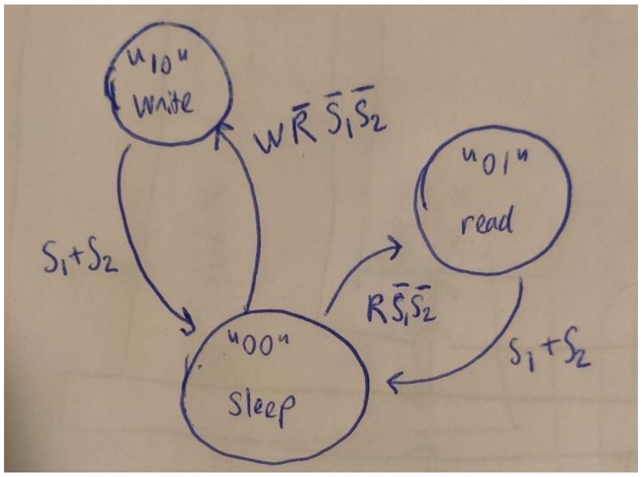


Figure 4: FSM state transitions

If the read and write signals are both high, the read signal has priority.

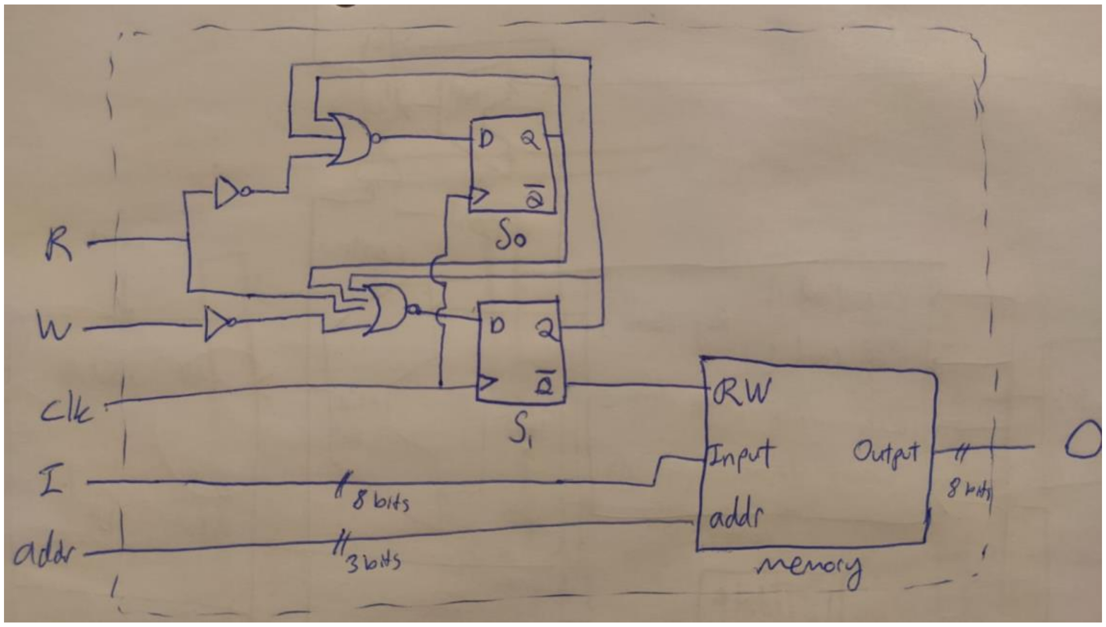


Figure 5: FSM diagram

FSM module also wraps the memory module. Thus, the input word and address signals are directly connected to the memory. The output signal of the memory is tied to the output of the module. FSM only controls RW input of the memory, so the user must ensure the input and address signals are kept for enough time until they are processed.

In the testbench for the simulation we issued write and read commands with half frequency of the clock so that FSM can insert the sleep state between reads and writes.

3 Results

3.1 Spice Simulation Results

We tested transient behavior of the bitcell by doing transient analysis in AIM-Spice. We repeated the simulation with five distinct design corners and three different temperatures. Then we did DC operating point analysis at various temperatures to report the static power dissipation.

3.1.1 Transient analysis for five different design corners

The temperature is set to 27 degrees for these simulations.

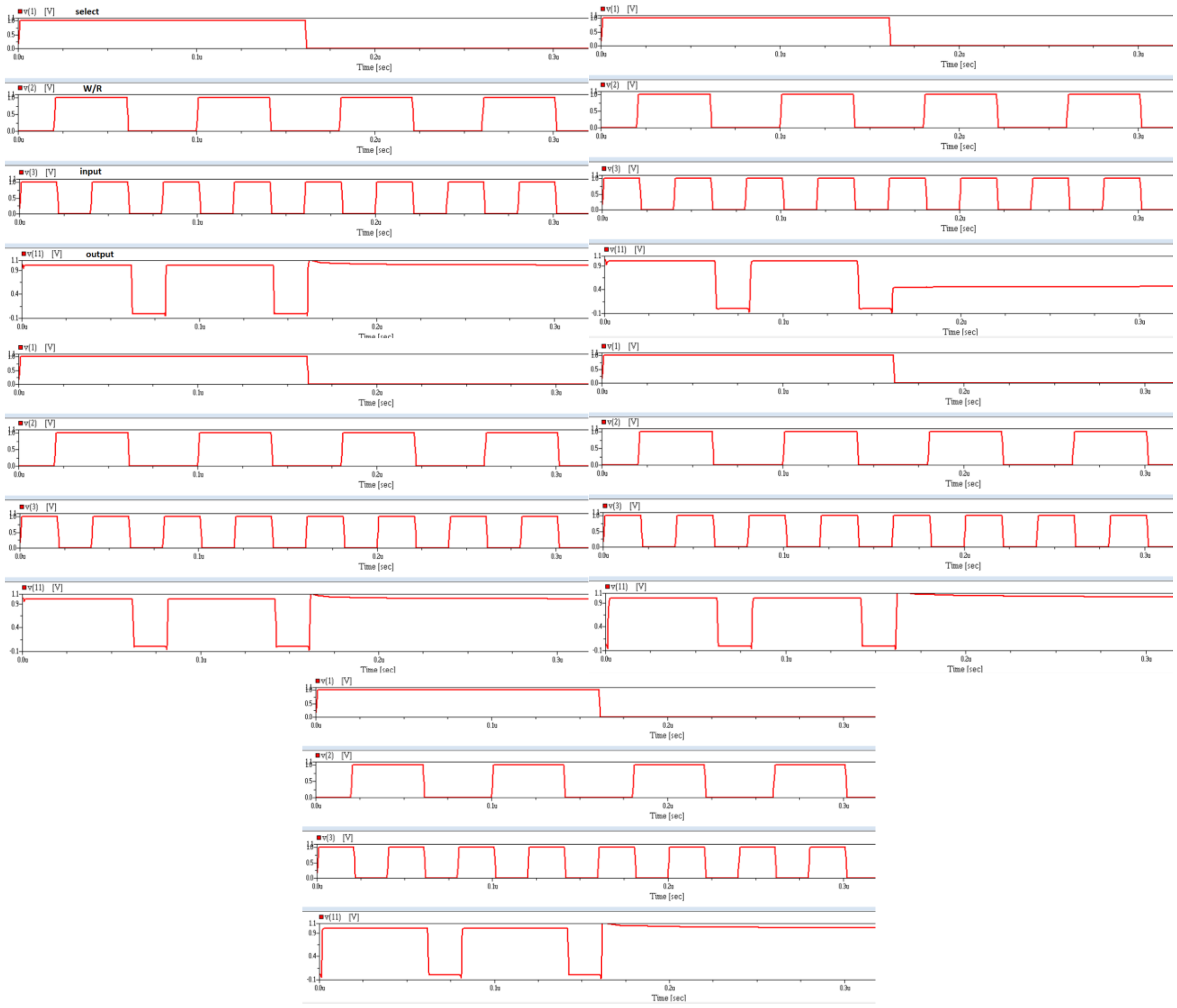


Figure 66: Transient analysis for five different design corners

The plots in figure 6 show that there are almost no visible differences between the operations of different design corners. The most obvious one is that the FF corner output has faster switching rate than the SS corner, which results in the SS corner output staying at a mid-voltage level when the output becomes floating.

3.1.2 Transient analysis for three different temperatures

TT corner is used in these simulations.

As we can see in figure 7, there are no visible differences in simulation results, but there must be a decrease in the switching rate because the mobility of the carriers decreases with increasing temperature.

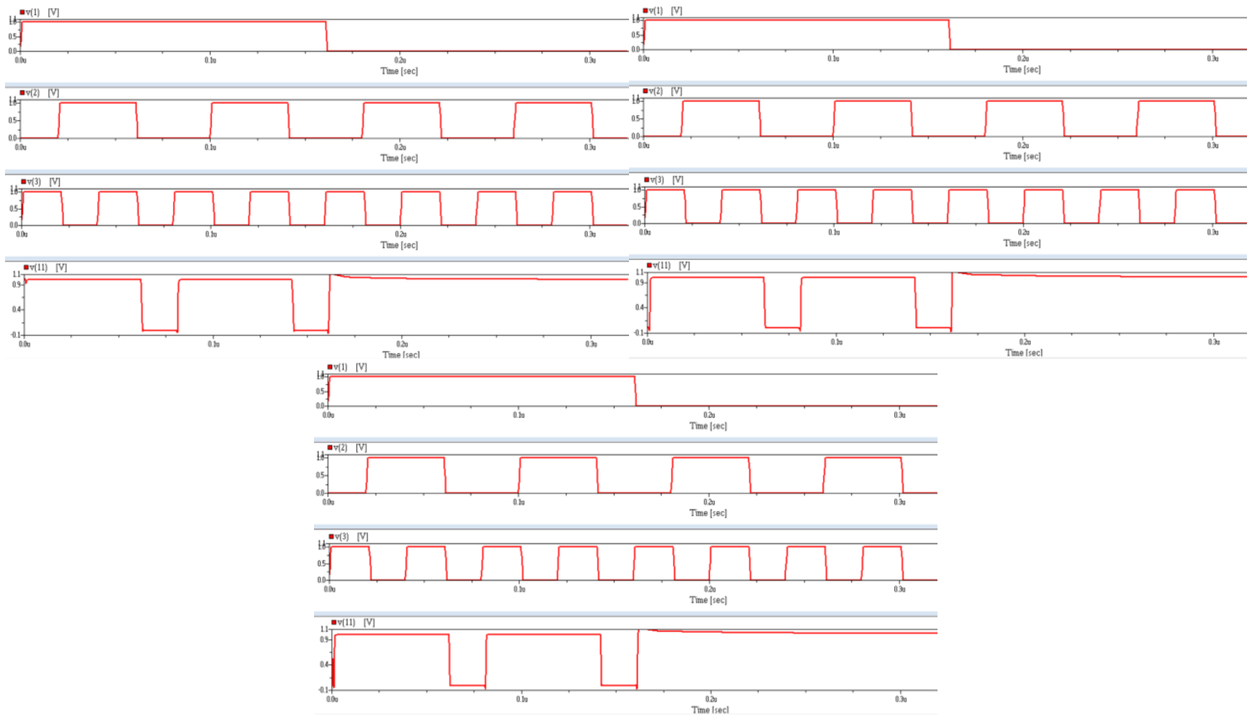
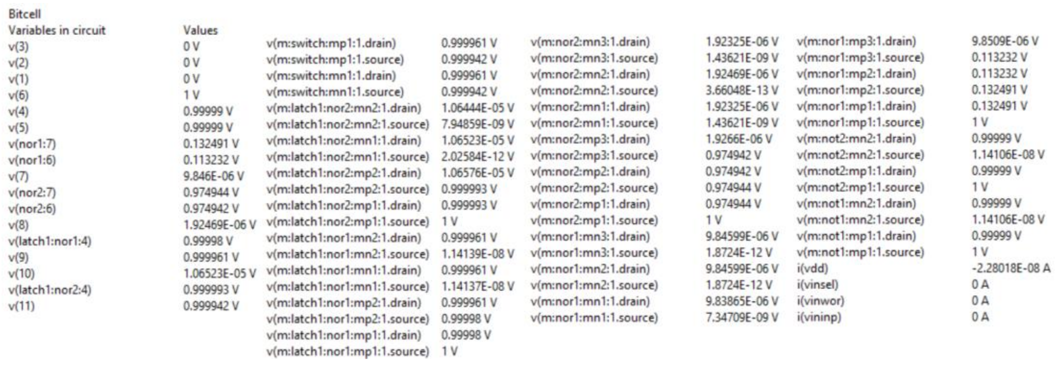


Figure 77: Transient analysis for three different temperatures

3.1.3 Operating Point Analysis for Leakage Power

We did DC operating point analysis of the bitcell in different temperatures. Because the input signals are stable and the transistors are not switching, the product of supply voltage and current gives the leakage power.



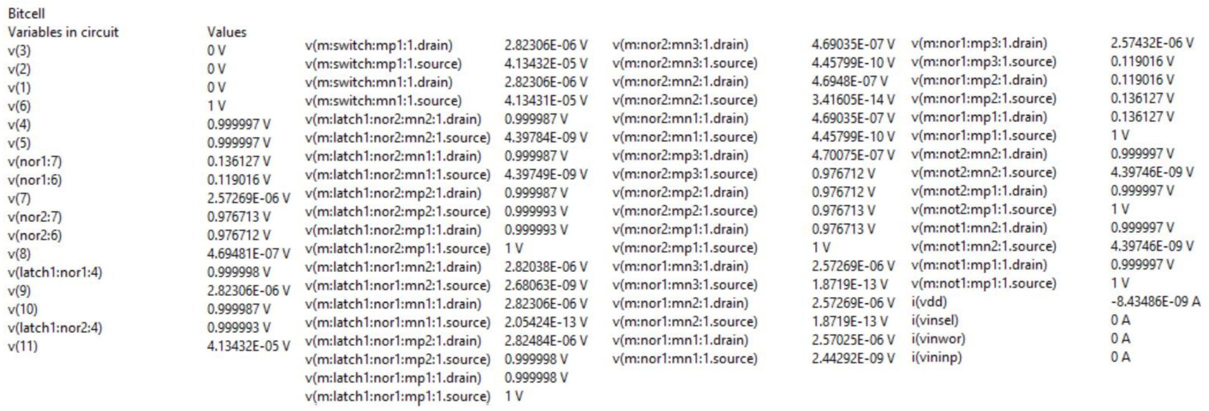
Bitcell

Variables in circuit	Values						
v(3)	0 V	v(m:switch:mp1:1.drain)	0.999961 V	v(m:nor2:mn3:1.drain)	1.92325E-06 V	v(m:nor1:mp3:1.drain)	9.8509E-06 V
v(2)	0 V	v(m:switch:mp1:1.source)	0.999942 V	v(m:nor2:mn3:1.source)	1.43621E-09 V	v(m:nor1:mp3:1.source)	0.113232 V
v(1)	0 V	v(m:switch:mn1:1.drain)	0.999961 V	v(m:nor2:mn2:1.drain)	1.92469E-06 V	v(m:nor1:mp2:1.drain)	0.113232 V
v(6)	1 V	v(m:switch:mn1:1.source)	0.999942 V	v(m:nor2:mn2:1.source)	3.66048E-13 V	v(m:nor1:mp2:1.source)	0.132491 V
v(4)	0.99999 V	v(m:latch1:nor2:mn2:1.drain)	1.06444E-05 V	v(m:nor2:mn1:1.drain)	1.92325E-06 V	v(m:nor1:mp1:1.drain)	0.132491 V
v(5)	0.99999 V	v(m:latch1:nor2:mn2:1.source)	7.94859E-09 V	v(m:nor2:mn1:1.source)	1.43621E-09 V	v(m:nor1:mp1:1.source)	1 V
v(nor1:7)	0.132491 V	v(m:latch1:nor2:mn1:1.drain)	1.06523E-05 V	v(m:nor2:mp3:1.drain)	1.9266E-06 V	v(m:not2:mn2:1.drain)	0.99999 V
v(nor1:6)	0.113232 V	v(m:latch1:nor2:mn1:1.source)	2.02584E-12 V	v(m:nor2:mp3:1.source)	0.974942 V	v(m:not2:mn2:1.source)	1.14106E-08 V
v(7)	9.846E-06 V	v(m:latch1:nor2:mp2:1.drain)	1.06576E-05 V	v(m:nor2:mp2:1.drain)	0.974942 V	v(m:not2:mp1:1.drain)	0.99999 V
v(nor2:7)	0.974944 V	v(m:latch1:nor2:mp2:1.source)	0.999993 V	v(m:nor2:mp2:1.source)	0.974944 V	v(m:not2:mp1:1.source)	1 V
v(nor2:6)	0.974942 V	v(m:latch1:nor2:mp1:1.drain)	0.999993 V	v(m:nor2:mp1:1.drain)	0.974944 V	v(m:not1:mn2:1.drain)	0.99999 V
v(8)	1.92469E-06 V	v(m:latch1:nor2:mp1:1.source)	1 V	v(m:nor2:mp1:1.source)	1 V	v(m:not1:mn2:1.source)	1.14106E-08 V
v(latch1:nor1:4)	0.99998 V	v(m:latch1:nor1:mn2:1.drain)	0.999961 V	v(m:nor1:mn3:1.drain)	9.84599E-06 V	v(m:not1:mp1:1.drain)	0.99999 V
v(9)	0.999961 V	v(m:latch1:nor1:mn2:1.source)	1.14139E-08 V	v(m:nor1:mn3:1.source)	1.8724E-12 V	v(m:not1:mp1:1.source)	1 V
v(10)	1.06523E-05 V	v(m:latch1:nor1:mn1:1.drain)	0.999961 V	v(m:nor1:mn2:1.drain)	9.84599E-06 V	i(vdd)	-2.28018E-08 A
v(latch1:nor2:4)	0.999993 V	v(m:latch1:nor1:mn1:1.source)	1.14137E-08 V	v(m:nor1:mn2:1.source)	1.8724E-12 V	i(vinsel)	0 A
v(11)	0.999942 V	v(m:latch1:nor1:mp2:1.drain)	0.999961 V	v(m:nor1:mn1:1.drain)	9.83865E-06 V	i(vinwor)	0 A
		v(m:latch1:nor1:mp2:1.source)	0.99998 V	v(m:nor1:mn1:1.source)	7.34709E-09 V	i(vininp)	0 A
		v(m:latch1:nor1:mp1:1.drain)	0.99998 V				
		v(m:latch1:nor1:mp1:1.source)	1 V				

Table 2: DC operating parameters of the circuit at 70 °C

Power consumption is

$$P_{stat} = 1V \times 22.8\,nA = 22.8\,nW$$



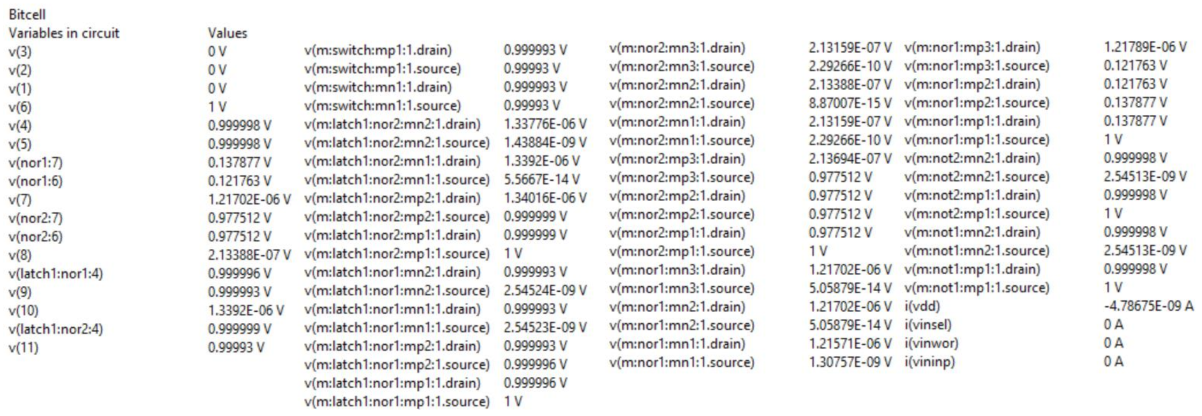
Bitcell

Variables in circuit	Values						
v(3)	0 V	v(m:switch:mp1:1.drain)	2.82306E-06 V	v(m:nor2:mn3:1.drain)	4.69035E-07 V	v(m:nor1:mp3:1.drain)	2.57432E-06 V
v(2)	0 V	v(m:switch:mp1:1.source)	4.13432E-05 V	v(m:nor2:mn3:1.source)	4.45799E-10 V	v(m:nor1:mp3:1.source)	0.119016 V
v(1)	0 V	v(m:switch:mn1:1.drain)	2.82306E-06 V	v(m:nor2:mn2:1.drain)	4.6948E-07 V	v(m:nor1:mp2:1.drain)	0.119016 V
v(6)	1 V	v(m:switch:mn1:1.source)	4.13431E-05 V	v(m:nor2:mn2:1.source)	3.41605E-14 V	v(m:nor1:mp2:1.source)	0.136127 V
v(4)	0.999997 V	v(m:latch1:nor2:mn2:1.drain)	0.999987 V	v(m:nor2:mn1:1.drain)	4.69035E-07 V	v(m:nor1:mp1:1.drain)	0.136127 V
v(5)	0.999997 V	v(m:latch1:nor2:mn2:1.source)	4.39784E-09 V	v(m:nor2:mn1:1.source)	4.45799E-10 V	v(m:nor1:mp1:1.source)	1 V
v(nor1:7)	0.136127 V	v(m:latch1:nor2:mn1:1.drain)	0.999987 V	v(m:nor2:mp3:1.drain)	4.70075E-07 V	v(m:not2:mn2:1.drain)	0.999997 V
v(nor1:6)	0.119016 V	v(m:latch1:nor2:mn1:1.source)	4.39749E-09 V	v(m:nor2:mp3:1.source)	0.976712 V	v(m:not2:mn2:1.source)	4.39746E-09 V
v(7)	2.57269E-06 V	v(m:latch1:nor2:mp2:1.drain)	0.999987 V	v(m:nor2:mp2:1.drain)	0.976712 V	v(m:not2:mp1:1.drain)	0.999997 V
v(nor2:7)	0.976713 V	v(m:latch1:nor2:mp2:1.source)	0.999993 V	v(m:nor2:mp2:1.source)	0.976713 V	v(m:not2:mp1:1.source)	1 V
v(nor2:6)	0.976712 V	v(m:latch1:nor2:mp1:1.drain)	0.999993 V	v(m:nor2:mp1:1.drain)	0.976713 V	v(m:not1:mn2:1.drain)	0.999997 V
v(8)	4.69481E-07 V	v(m:latch1:nor2:mp1:1.source)	1 V	v(m:nor2:mp1:1.source)	1 V	v(m:not1:mn2:1.source)	4.39746E-09 V
v(latch1:nor1:4)	0.999998 V	v(m:latch1:nor1:mn2:1.drain)	2.82038E-06 V	v(m:nor1:mn3:1.drain)	2.57269E-06 V	v(m:not1:mp1:1.drain)	0.999997 V
v(9)	2.82306E-06 V	v(m:latch1:nor1:mn2:1.source)	2.68063E-09 V	v(m:nor1:mn3:1.source)	1.8719E-13 V	v(m:not1:mp1:1.source)	1 V
v(10)	0.999987 V	v(m:latch1:nor1:mn1:1.drain)	2.82306E-06 V	v(m:nor1:mn2:1.drain)	2.57269E-06 V	i(vdd)	-8.43486E-09 A
v(latch1:nor2:4)	0.999993 V	v(m:latch1:nor1:mn1:1.source)	2.05424E-13 V	v(m:nor1:mn2:1.source)	1.8719E-13 V	i(vinsel)	0 A
v(11)	4.13432E-05 V	v(m:latch1:nor1:mp2:1.drain)	2.82484E-06 V	v(m:nor1:mn1:1.drain)	2.57025E-06 V	i(vinwor)	0 A
		v(m:latch1:nor1:mp2:1.source)	0.999998 V	v(m:nor1:mn1:1.source)	2.44292E-09 V	i(vininp)	0 A
		v(m:latch1:nor1:mp1:1.drain)	0.999998 V				
		v(m:latch1:nor1:mp1:1.source)	1 V				

Table 3: DC operating parameters of the circuit at 27 °C

Power consumption is

$$P_{stat} = 1V \times 8.43\,nA = 8.43\,nW$$



Bitcell

Variables in circuit	Values						
v(3)	0 V	v(m:switch:mp1:1.drain)	0.999993 V	v(m:nor2:mn3:1.drain)	2.13159E-07 V	v(m:nor1:mp3:1.drain)	1.21789E-06 V
v(2)	0 V	v(m:switch:mp1:1.source)	0.99993 V	v(m:nor2:mn3:1.source)	2.29266E-10 V	v(m:nor1:mp3:1.source)	0.121763 V
v(1)	0 V	v(m:switch:mn1:1.drain)	0.999993 V	v(m:nor2:mn2:1.drain)	2.13388E-07 V	v(m:nor1:mp2:1.drain)	0.121763 V
v(6)	1 V	v(m:switch:mn1:1.source)	0.99993 V	v(m:nor2:mn2:1.source)	8.87007E-15 V	v(m:nor1:mp2:1.source)	0.137877 V
v(4)	0.999998 V	v(m:latch1:nor2:mn2:1.drain)	1.33776E-06 V	v(m:nor2:mn1:1.drain)	2.13159E-07 V	v(m:nor1:mp1:1.drain)	0.137877 V
v(5)	0.999998 V	v(m:latch1:nor2:mn2:1.source)	1.43884E-09 V	v(m:nor2:mn1:1.source)	2.29266E-10 V	v(m:nor1:mp1:1.source)	1 V
v(nor1:7)	0.137877 V	v(m:latch1:nor2:mn1:1.drain)	1.3392E-06 V	v(m:nor2:mp3:1.drain)	2.13694E-07 V	v(m:not2:mn2:1.drain)	0.999998 V
v(nor1:6)	0.121763 V	v(m:latch1:nor2:mn1:1.source)	5.5667E-14 V	v(m:nor2:mp3:1.source)	0.977512 V	v(m:not2:mn2:1.source)	2.54513E-09 V
v(7)	1.21702E-06 V	v(m:latch1:nor2:mp2:1.drain)	1.34016E-06 V	v(m:nor2:mp2:1.drain)	0.977512 V	v(m:not2:mp1:1.drain)	0.999998 V
v(nor2:7)	0.977512 V	v(m:latch1:nor2:mp2:1.source)	0.999999 V	v(m:nor2:mp2:1.source)	0.977512 V	v(m:not2:mp1:1.source)	1 V
v(nor2:6)	0.977512 V	v(m:latch1:nor2:mp1:1.drain)	0.999999 V	v(m:nor2:mp1:1.drain)	0.977512 V	v(m:not1:mn2:1.drain)	0.999998 V
v(8)	2.13388E-07 V	v(m:latch1:nor2:mp1:1.source)	1 V	v(m:nor2:mp1:1.source)	1 V	v(m:not1:mn2:1.source)	2.54513E-09 V
v(latch1:nor1:4)	0.999996 V	v(m:latch1:nor1:mn2:1.drain)	0.999993 V	v(m:nor1:mn3:1.drain)	1.21702E-06 V	v(m:not1:mp1:1.drain)	0.999998 V
v(9)	0.999993 V	v(m:latch1:nor1:mn2:1.source)	2.54524E-09 V	v(m:nor1:mn3:1.source)	5.05879E-14 V	v(m:not1:mp1:1.source)	1 V
v(10)	1.3392E-06 V	v(m:latch1:nor1:mn1:1.drain)	0.999993 V	v(m:nor1:mn2:1.drain)	1.21702E-06 V	i(vdd)	-4.78675E-09 A
v(latch1:nor2:4)	0.999999 V	v(m:latch1:nor1:mn1:1.source)	2.54523E-09 V	v(m:nor1:mn2:1.source)	5.05879E-14 V	i(vinsel)	0 A
v(11)	0.99993 V	v(m:latch1:nor1:mp2:1.drain)	0.999993 V	v(m:nor1:mn1:1.drain)	1.21571E-06 V	i(vinwor)	0 A
		v(m:latch1:nor1:mp2:1.source)	0.999996 V	v(m:nor1:mn1:1.source)	1.30757E-09 V	i(vininp)	0 A
		v(m:latch1:nor1:mp1:1.drain)	0.999996 V				
		v(m:latch1:nor1:mp1:1.source)	1 V				

Table 4: DC operating parameters of the circuit at 7 °C

Power consumption is

$$P_{stat} = 1V \times 4.79\,nA = 4.79\,nW$$

The results for 6 degrees Celsius or less are not reliable, because there is an abrupt jump in i(Vdd) afterwards. For instance, at 0 degrees Celsius, for some reason i(Vdd) is calculated as 43.1μA which gives

$$P_{stat} = 1V \times 43.1\,\mu A = 43.1\,\mu W$$

and it doesn't make sense because it is actually supposed to be less than the simulation with 7 degrees.

3.2 Verilog Simulation Results

All Verilog modules for the circuit are created in Active-HDL and are simulated with testbenches.

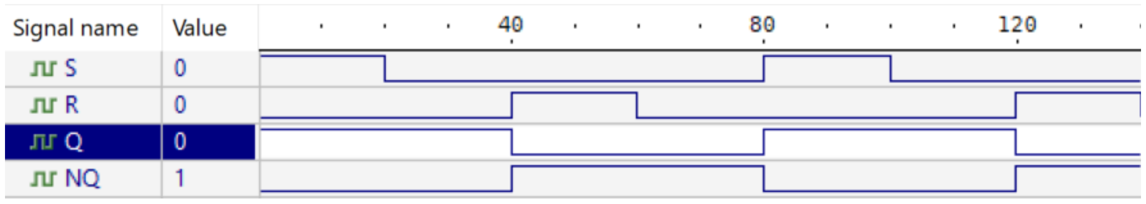


Figure 88: SR-latch simulation

As expected, output is high when S=1 and R=0, low when S=0 and R=1, and is saved when S=0 and R=0.

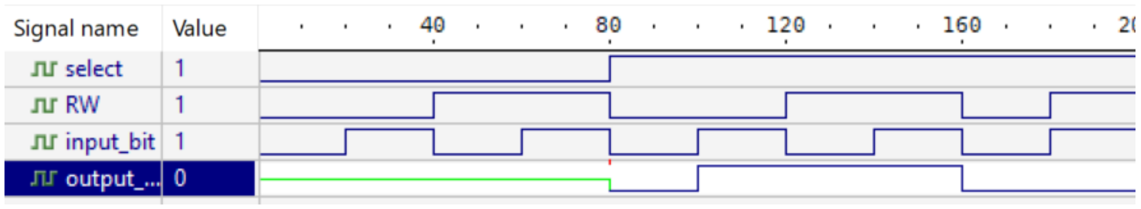


Figure 99: Bitcell simulation

Because of the transmission gate at the output, output is floating when select is low. Then when the cell is selected, the input bit is stored when RW=0, and output bit is equal to stored bit.

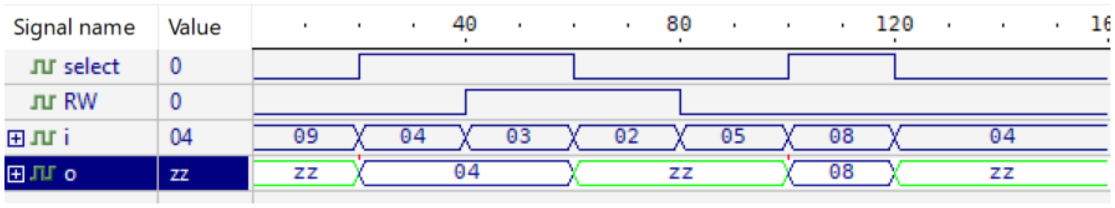


Figure 1010: Word simulation

Word behaves the same as bitcell but is 8-bit wide.

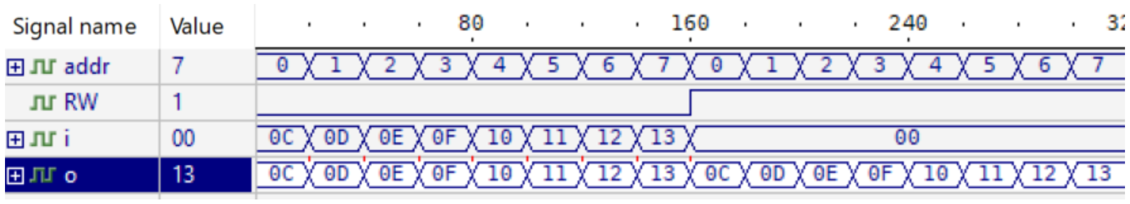


Figure 1111: Memory simulation

Simulation results show that the address bits are successfully decoded to create select signals for the word lines. We can write words to all eight addresses and read back.

Rarely (when the order of writing to the addresses and the words are some specific values) we see that we don't read exactly what we wrote to some addresses.

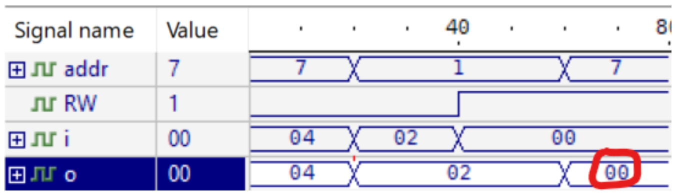


Figure 1212: A specific case where the memory fails, we do not read what we wrote to the address 7

The reason for the failure is that the select signal produced from the address signals and RW signal do not reach word line at the same time, even though we are doing behavioral simulation. If we add buffers to either address or RW lines we can only make sure that one signal reaches before the other one, but that does not help since the memory reliably works only when these signals switch simultaneously. And since we are not using clock for the memory part, we decided we cannot solve this issue in memory module, but we designed FSM module so that there is no immediate read or write operations before or after a write instruction, which solves the issue. Also, we should note that memory works correctly almost always but fails in very specific rare cases. For instance, the memory does not fail if the address is not 7 or the word written to the address is not 0x04 as in figure 12.

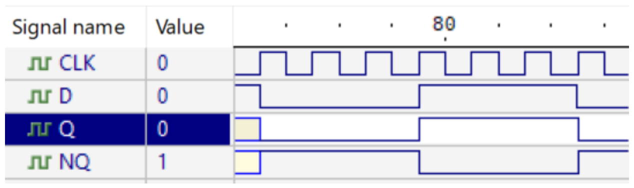


Figure 1313: D flip-flop simulation

The flip-flop used in the FSM works as expected. In the simulation, the output is only updated to the input value at the rising edges of the clock. The input D switches 100 ps earlier than the clock, which is realistic since the flip-flop requires a setup time for the input signal like all sequential circuits.

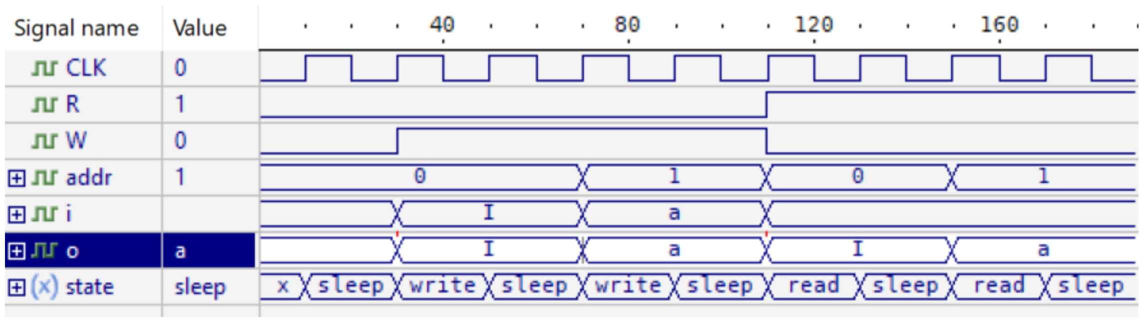


Figure 1414: FSM simulation

As the plots demonstrate, we are able to write our group member Ia’s name to the memory and read back. The frequency of interacting with the memory is half of the clock frequency because there is a sleep state inserted between all read and writes states. As explained before, this solves race conditions.

4 Discussion

The main aim of the project was to design a low power memory and controller. To achieve that, as we mentioned before, primarily, we focused on minimizing the number of transistors. Our bitcell has 26 transistors (64 bitcells in total), and 54 additional transistors for the select signal logic. Therefore, the memory has 26×64+54=1718 transistors in total. The FSM module includes the memory, the flip-flops, and the state transition logic, which sums up to 1718+36×2+18=1808 transistors. We think that this number could be decreased even further if we focused more on the logic elements outside the bitcells. However, the total must be close to the minimum possible, since the hugest portion of transistors of the system are the bitcells.

When we simulated our circuit in the AIM-Spice, we mostly got the results we were expecting, as you can see in figure 7. The lowest leakage current we got was at the temperature of 7 degrees Celsius. As we mentioned before, for temperatures less than 7, our results got worse, and we got a higher leakage current. As for the corners, we could not see much of the difference.

To further improve our results, we could have tried a wider variety of the different voltages. Moreover, we could try other technologies as well and compare the results.

The main problem with the Verilog simulations was that we were doing behavioral simulation, so our expectation was to not see any delays in the signals. However even though when we zoom in into the problematic sections of the plot we cannot see these delays, delays occur, and it sometimes causes incorrect writings to the memory. We are guessing it is a problem with the software. A good idea is to synthesize the circuit and do a simulation afterwards. Then we would be able to see realistic propagation delays and try to solve the problem.

However, the only way to get rid of race conditions inside the memory is to make it clocked. In this case the input signals to the memory would be given at one cycle of the clock and the results of read or write operation would appear at the output in the next cycle.

Overall, we got the memory array with the ability to store and load the correct data, with low static power dissipation, but there is always a room for improvement, we could improve it by more adjustments to the design, voltage or to the transistor properties.

5 Appendix

5.1 Spice codes

5.1.1 not_gate

```
*                 1 - VIN 2 - VOUT  3 - VDD
.subckt not_gate 1 2 3
.param ln_not = 0.3u wn_not = 0.5u
.param lp_not = 0.3u wp_not = 1.5u
*    d g s b
xmp1 2 1 3 3 pmos1v l=lp_not w= wp_not
xmn2 2 1 0 0 nmos1v l=ln_not w= wn_not
.ends
```

5.1.2 nor_gate

```
*                1 - Input A 2 - Input B 3 - VDD 5 - Output
.subckt nor_gate 1 2 3 5
.param ln_nor = 0.3u wn_nor = 0.5u
.param lp_nor = 0.3u wp_nor = 1.5u

*    d g s b
xmp1 4 1 3 3 pmos1v l=lp_nor w=wp_nor
xmp2 5 2 4 3 pmos1v l=lp_nor w=wp_nor

xmn1 5 1 0 0 nmos1v l=ln_nor w=wn_nor
xmn2 5 2 0 0 nmos1v l=ln_nor w=wn_nor
.ends
```

5.1.3 nor_gate3

```
*  1 - Input A 2 - Input B 3 - Input C 4 - VDD 5 - Output
.subckt nor_gate3 1 2 3 4 5
.param ln_nor3 = 0.3u wn_nor3 = 0.5u
.param lp_nor3 = 0.3u wp_nor3 = 1.5u

*    d g s b
xmp1 7 1 4 4 pmos1v l=lp_nor3 w=wp_nor3
xmp2 6 2 7 4 pmos1v l=lp_nor3 w=wp_nor3
xmp3 5 3 6 4 pmos1v l=lp_nor3 w=wp_nor3

xmn1 5 1 0 0 nmos1v l=ln_nor3 w=wn_nor3
xmn2 5 2 0 0 nmos1v l=ln_nor3 w=wn_nor3
xmn3 5 3 0 0 nmos1v l=ln_nor3 w=wn_nor3
.ends
```

5.1.4 switch

```
*  1 - input 2 - Output  3 - control 4 - Not control 5 - vdd
.subckt switch 1 2 3 4 5

.param ln_swtch = 0.3u wn_swtch  = 0.5u
.param lp_swtch = 0.3u wp_swtch  = 1.5u

*     d g s b
 xmn1 1 3 2 0 nmos1v l=ln_swtch w=wn_swtch
 xmp1 1 4 2 5 pmos1v l=lp_swtch w=wp_swtch
.ends
```

5.1.5 sr_latch

```
*                R S VDD Q NQ
.subckt sr_latch 1 2 3 4 5
*      A B VDD OUT
xnor1  1 5 3 4 nor_gate
xnor2  2 4 3 5 nor_gate
.ends
```

5.1.6 bitcell

```
Bitcell
.include 90_nm_gpdk\gpdk90nm_tt.cir
.include nor_gate3.cir
.include nor_gate.cir
.include not_gate.cir
.include switch.cir
.include sr_latch.cir


vinInp 3 0 dc 0 pulse(0 1 0ns 1ns 1ns 20ns 40ns)
vinWoR 2 0 dc 0 pulse(0 1 20ns 1ns 1ns 40ns 80ns)
vinSel 1 0 dc 0 pulse(0 1 0ns 1ns 1ns 160ns 320ns)
vdd  6 0 dc 1

* inverted select signal  1 - VIN 2 - VOUT  3 - VDD
xnot1 1 4 6 not_gate
*inverted input signal
xnot2 3 5 6 not_gate

* NOR gate for R 1 - Input A 2 - Input B 3 - Input C 4 - VDD 5 -
Output
xnor1 4 2 3 6 7 nor_gate3
* NOR gate for S
```

```
xnor2 4 2 5 6 8 nor_gate3

* SR Latch  R S VDD Q NQ
xlatch1 7 8 6 9 10 sr_latch

* Switch  1 - input 2 - Output  3 - control 4 - Not control 5 - vdd
xswitch 9 11 1 4 6 switch
```

5.2 Verilog codes

5.2.1 SR Latch

```
`timescale 1 ns / 1 ps
             module sr_latch ( S ,R ,Q ,NQ );
             input S,R;
 output Q,NQ;
  nor (Q,R,NQ);
  nor (NQ,S,Q);
endmodule
```

5.2.2 SR Latch Testbench

```
`timescale 1 ns / 1 ps
              module sr_latch_tb;
  reg S,R;
  wire Q,NQ;
  sr_latch sr_1 (
  .S(S),
  .R(R),
  .Q(Q),
  .NQ(NQ)
  );
   initial
        begin
             assign S = 1;
             assign R = 0;

        #20      assign S = 0;
              assign R = 0;

         #20     assign S = 0;
             assign R = 1;

         #20      assign S = 0;
              assign R = 0;
        #20       assign S = 1;
              assign R = 0;

        #20      assign S = 0;
             assign R = 0;

       #20       assign S = 0;
             assign R = 1;

       #20       assign S = 0;
             assign R = 0;
            $finish;
                  end
endmodule
```

5.2.3 Bitcell

```
`timescale 1 ns / 1 ps
             module bitcell ( select ,RW ,input_bit ,output_bit);
 input  select,RW,input_bit;
 output output_bit;
 wire   n_select,n_input_bit,Q,NQ,R,S;

  not (n_select,select);
  not (n_input_bit,input_bit);

  nor(R,n_select,RW,input_bit);
  nor(S,n_select,RW,n_input_bit);

  sr_latch latch1(
    .R  (R),
    .S  (S),
     .Q  (Q),
    .NQ (NQ)
  );
  cmos (output_bit,Q, select, n_select);
endmodule
```

5.2.4 Bitcell Testbench

```
`timescale 1 ns / 1 ps
module bitcell_tb;
  reg  select,RW,input_bit;
  wire output_bit;
```

```
bitcell bitcell_DUT(
.select(select),
.RW(RW),
.input_bit(input_bit),
.output_bit(output_bit)
);

initial
        begin
                   assign select    = 0;
                   assign RW        = 0;
                   assign input_bit = 0;
         #20
                   assign select    = 0;
                   assign RW        = 0;
                   assign input_bit = 1;
         #20
                   assign select    = 0;
                   assign RW        = 1;
                   assign input_bit = 0;
             #20
                   assign select    = 0;
                   assign RW        = 1;
                   assign input_bit = 1;
         #20
                   assign select    = 1;
                   assign RW        = 0;
                   assign input_bit = 0;
         #20
                   assign select    = 1;
                   assign RW        = 0;
                   assign input_bit = 1;
         #20
                   assign select    = 1;
                   assign RW        = 1;
                   assign input_bit = 0;
         #20
                   assign select    = 1;
                   assign RW        = 1;
                   assign input_bit = 1;
         #20
                   assign select    = 1;
                   assign RW        = 0;
                   assign input_bit = 0;
         #20
                   assign select    = 1;
                   assign RW        = 1;
```

```
                assign input_bit = 1;
         #20
                $finish;
        end
endmodule
```

5.2.5 Word

```
`timescale 1 ns / 1 ps
module word(RW, select, i, o);
  input  RW, select;
  input  [7:0] i;
  output [7:0] o;

  bitcell b0(.select(select), .RW(RW), .input_bit(i[0]),
.output_bit(o[0]));
  bitcell b1(.select(select), .RW(RW), .input_bit(i[1]),
.output_bit(o[1]));
  bitcell b2(.select(select), .RW(RW), .input_bit(i[2]),
.output_bit(o[2]));
  bitcell b3(.select(select), .RW(RW), .input_bit(i[3]),
.output_bit(o[3]));
  bitcell b4(.select(select), .RW(RW), .input_bit(i[4]),
.output_bit(o[4]));
  bitcell b5(.select(select), .RW(RW), .input_bit(i[5]),
.output_bit(o[5]));
  bitcell b6(.select(select), .RW(RW), .input_bit(i[6]),
.output_bit(o[6]));
  bitcell b7(.select(select), .RW(RW), .input_bit(i[7]),
.output_bit(o[7]));
endmodule
```

5.2.6 Word Testbench

```
`timescale 1 ns / 1 ps
module word_tb;
  reg  RW;
  reg  [7:0] i;
  reg  select;
  wire [7:0] o;

  word word_DUT(.RW(RW), .select(select), .i(i), .o(o));

  initial
        begin
```

```
              assign RW     = 0;
              assign select = 0;
               assign i      = 9;
       #20
              assign RW     = 0;
              assign select = 1;
               assign i      = 4;
       #20
              assign RW     = 1;
              assign select = 1;
               assign i      = 3;
       #20
              assign RW     = 1;
              assign select = 0;
               assign i      = 2;
       #20
              assign RW     = 0;
              assign select = 0;
               assign i      = 5;
       #20
              assign RW     = 0;
              assign select = 1;
               assign i      = 8;
       #20
              assign RW     = 0;
              assign select = 0;
               assign i      = 4;
       #20
              assign RW     = 0;
              assign select = 0;
               assign i      = 4;
       #20
              $finish;
       end
endmodule
```

5.2.7 Memory

```
`timescale 1 ns / 1 ps
module mem(RW, addr0,addr1,addr2, i0,i1,i2,i3,i4,i5,i6,i7,
o0,o1,o2,o3,o4,o5,o6,o7);
  input  RW,addr0,addr1,addr2, i0,i1,i2,i3,i4,i5,i6,i7;
  output o0,o1,o2,o3,o4,o5,o6,o7;
  wire   Naddr0,Naddr1,Naddr2,s0,s1,s2,s3,s4,s5,s6,s7;
  wire [7:0] o, i;
  wire RWd;

  assign i[0] = i0;
  assign i[1] = i1;
  assign i[2] = i2;
  assign i[3] = i3;
  assign i[4] = i4;
  assign i[5] = i5;
  assign i[6] = i6;
  assign i[7] = i7;

  assign o0 = o[0];
  assign o1 = o[1];
  assign o2 = o[2];
  assign o3 = o[3];
  assign o4 = o[4];
  assign o5 = o[5];
  assign o6 = o[6];
  assign o7 = o[7];

  // Selects 6*5 + 8*3  = 54 transistors   ----------
  nor (s0,addr2,addr1,addr0);     // address bits adr0' * adr1' *
adr1' -> (adr2 + adr1 + adr0)'

  not (Naddr0,addr0);
  nor (s1,addr2,addr1,Naddr0);    // address bits adr0' * adr1' * adr1
->  (adr2 + adr1 + adr0')'

  not (Naddr1,addr1);
  nor (s2,addr2,Naddr1,addr0);    // address bits adr0' * adr1 * adr1'
->  (adr2 + adr1' + adr0)'

  nor (s3,addr2,Naddr1,Naddr0);   // address bits adr0' * adr1 * adr1
->  (adr2 + adr1' + adr0')'

  not (Naddr2,addr2);
  nor (s4,Naddr2,addr1,addr0);    // address bits adr0 * adr1' * adr1'
->  (adr2' + adr1 + adr0)'

  nor (s5,Naddr2,addr1,Naddr0);   // address bits adr0 * adr1' * adr1
->  (adr2' + adr1 + adr0')'

  nor (s6,Naddr2,Naddr1,addr0);   // address bits adr0 * adr1 * adr1'
->  (adr2' + adr1' + adr0)'

  nor (s7,Naddr2,Naddr1,Naddr0);  // address bits adr0 * adr1 * adr1 -
>  (adr2' + adr1' + adr0')'


  word w0 (.RW(RW),.select(s0), .i(i), .o(o));
  word w1 (.RW(RW),.select(s1), .i(i), .o(o));
  word w2 (.RW(RW),.select(s2), .i(i), .o(o));
  word w3 (.RW(RW),.select(s3), .i(i), .o(o));
  word w4 (.RW(RW),.select(s4), .i(i), .o(o));
  word w5 (.RW(RW),.select(s5), .i(i), .o(o));
  word w6 (.RW(RW),.select(s6), .i(i), .o(o));
  word w7 (.RW(RW),.select(s7), .i(i), .o(o));
endmodule
```

5.2.8 Memory Testbench

```
`timescale 1 ns / 1 ps
module mem_tb;
  reg  RW;
  reg  [7:0] i;
  reg  [2:0] addr;
  wire [7:0] o;

  mem mem_DUT(.RW(RW), .addr0(addr[0]), .addr1(addr[1]),.
addr2(addr[2]),

.i0(i[0]),.i1(i[1]),.i2(i[2]),.i3(i[3]),.i4(i[4]),.i5(i[5]),.i6(i[6]),
.i7(i[7]),

.o0(o[0]),.o1(o[1]),.o2(o[2]),.o3(o[3]),.o4(o[4]),.o5(o[5]),.o6(o[6]),
.o7(o[7]));

  initial
       begin
              // Write ADR 7 Value 4
              assign RW    = 0;
              assign addr  = 0;
               assign i     = 12;
        #20
              // Write ADR 7 Value 4
              assign RW    = 0;
              assign addr  = 1;
               assign i     = 13;
        #20
              // Write ADR 7 Value 4
              assign RW    = 0;
              assign addr  = 2;
               assign i     = 14;
        #20
              // Write ADR 7 Value 4
              assign RW    = 0;
              assign addr  = 3;
```

```
          assign i     = 15;
#20
         // Write ADR 7 Value 4
         assign RW    = 0;
         assign addr  = 4;
          assign i     = 16;
#20
         // Write ADR 7 Value 4
         assign RW    = 0;
         assign addr  = 5;
          assign i     = 17;
#20
         // Write ADR 7 Value 4
         assign RW    = 0;
         assign addr  = 6;
          assign i     = 18;
#20
         // Write ADR 7 Value 4
         assign RW    = 0;
         assign addr  = 7;
          assign i     = 19;
#20

         // Write ADR 7 Value 4
         assign RW    = 1;
         assign addr  = 0;
          assign i     = 0;
#20
         // Write ADR 7 Value 4
         assign RW    = 1;
         assign addr  = 1;
          assign i     = 0;
#20
         // Write ADR 7 Value 4
         assign RW    = 1;
         assign addr  = 2;
          assign i     = 0;
#20
         // Write ADR 7 Value 4
         assign RW    = 1;
         assign addr  = 3;
          assign i     = 0;
#20
         // Write ADR 7 Value 4
         assign RW    = 1;
         assign addr  = 4;
          assign i     = 0;
#20
```

```
              // Write ADR 7 Value 4
              assign RW    = 1;
              assign addr  = 5;
               assign i     = 0;
       #20
              // Write ADR 7 Value 4
              assign RW    = 1;
              assign addr  = 6;
               assign i     = 0;
       #20
              // Write ADR 7 Value 4
              assign RW    = 1;
              assign addr  = 7;
               assign i     = 0;
       #20

               $finish;
       end
endmodule
```

5.2.9 D Flip-flop

```
`timescale 1 ns / 1 ps
module d_flip_flop(D, CLK ,Q, NQ);

input  D, CLK;
output Q, NQ;
wire   S2,R2,Q1, NQ1,NCLK,ND;

  // A
  not  (ND,D);
  nor (S1,D,CLK);
  nor (R1,CLK,ND);

  nor (Q1,S1,NQ1);
  nor (NQ1,R1,Q1);

  // B
  not (NCLK,CLK);
  nor (S2,Q1,NCLK);
  nor (R2,NCLK,NQ1);

  nor (Q,S2,NQ);
  nor (NQ,R2,Q);
endmodule
```

5.2.10 D Flip-flop Testbench

```
`timescale 1 ns / 1 ps
module d_flip_flop_tb;
  reg  D,CLK = 0;
  wire Q,NQ;

  d_flip_flop f_1 (
    .D(D),
    .CLK(CLK),
    .Q(Q),
    .NQ(NQ)
  );

  always
  begin
    CLK = 0; #10;
    CLK = 1; #10;
  end

  initial
       begin
             assign D = 1;

        #9.9 assign D = 0;

           #20 assign D = 0;

           #20 assign D = 0;

        #20 assign D = 1;

        #20 assign D = 1;

           #20 assign D = 1;

           #20 assign D = 0;

           #20
           $finish;
       end
endmodule
```

5.2.11 FSM

```
`timescale 1 ns / 1 ps
```

```
module fsm(i0,i1,i2,i3,i4,i5,i6,i7,adr0,adr1,adr2,R,W,CLK,
o0,o1,o2,o3,o4,o5,o6,o7);

 input  i0,i1,i2,i3,i4,i5,i6,i7,adr0,adr1,adr2,CLK,R,W;
 output o0,o1,o2,o3,o4,o5,o6,o7;
 wire   Q1,Q2,NQ1,NQ2,RW, D1,D2, Rn,Wn;

 d_flip_flop A(.D (D1),
               .CLK(CLK),
               .Q (Q1),
               .NQ (NQ1));

 d_flip_flop B(.D (D2),
               .CLK(CLK),
               .Q (Q2),
               .NQ (NQ2));

 not(Rn, R);
 not(Wn, W);

 nor(D1, Rn, Q1, Q2);
 nor(D2, Wn, R, Q1, Q2);


 mem mem_1(.RW(NQ2),.addr0(adr0),.addr1(adr1),.addr2(adr2),

.i0(i0),.i1(i1),.i2(i2),.i3(i3),.i4(i4),.i5(i5),.i6(i6),.i7(i7),

.o0(o0),.o1(o1),.o2(o2),.o3(o3),.o4(o4),.o5(o5),.o6(o6),.o7(o7));
endmodule
```

5.2.10 FSM Testbench

```
`timescale 1 ns / 1 ps
module fsm_tb;
  reg  R,W,CLK = 0;
  wire i0,i1,i2,i3,i4,i5,i6,i7,adr0,adr1,adr2,
o0,o1,o2,o3,o4,o5,o6,o7;
  reg [7:0] i;
  wire [7:0] o;
  reg [2:0] addr;

  fsm fsm_1 (
      .i0(i0), .i1(i1), .i2(i2), .i3(i3), .i4(i4), .i5(i5), .i6(i6),
.i7(i7),
      .adr0(adr0), .adr1(adr1), .adr2(adr2), .R(R), .W(W), .CLK(CLK),
      .o0(o0), .o1(o1), .o2(o2), .o3(o3), .o4(o4), .o5(o5), .o6(o6),
.o7(o7)
  );
```

```
assign adr0 = addr[0];
assign adr1 = addr[1];
assign adr2 = addr[2];

assign i0 = i[0];
assign i1 = i[1];
assign i2 = i[2];
assign i3 = i[3];
assign i4 = i[4];
assign i5 = i[5];
assign i6 = i[6];
assign i7 = i[7];

assign o[0] = o0;
assign o[1] = o1;
assign o[2] = o2;
assign o[3] = o3;
assign o[4] = o4;
assign o[5] = o5;
assign o[6] = o6;
assign o[7] = o7;

always
begin
  CLK = 0; #10;
  CLK = 1; #10;
end

initial
     begin
           assign addr = 0;
           assign i = 0;
           assign R = 0;
           assign W = 0;

    #29.9  assign addr = 0;
           assign i = 73;
           assign R = 0;
           assign W = 1;

    #40 assign addr = 1;
           assign i = 97;
           assign R = 0;
           assign W = 1;

    #40 assign addr = 0;
           assign i = 0;
```

```
             assign R = 1;
             assign W = 0;

      #40 assign addr = 1;
             assign i = 0;
             assign R = 1;
             assign W = 0;

      #40
           $finish;
        end
endmodule
```