The project is about designing a small memory array that should contain 8 words, each having 8 bits. Low power implementation is among the design goals. Both Verilog and Spice simulations are required to complete the tasks. The gpdk 90 nm technology is to be used. Design your memory having knowledge relevant to TFE4152 in mind.

The work should end up in a written report, with all material delivered in a single pdf-file. Under a project related folder on BlackBoard you find excerpts from the book «*Digital Design – With an introduction to Verilog HDL*», 5th ed., by M. Morris Mano andMichael D. Ciletti, which contain information that may be useful. This includes, but isn't necessary limited to: chapter 2: «Boolean Algebra and Logic Gates», chapter 3: Gate-level Minimization», and chapter 7: «Memory and Programmable Logic».

The overall goal is to implement a low power memory, having it's top level as depicted in Figure 1. To limit the scope, while still providing room for demonstrating relevant knowledge, it should be an 8x8 memory, which means that it should contain a total of 64 bits (/bitcells). Priority should be to design the memory for low static leakage power, but with an emphasis on the chosen bitcell.

Your memory unit should adhere to the block diagram in Figure 1. Each word in the memory should have a particular address, starting from 0 up to 2^k-1. An internal decoder uses the address to select the n–bit word specified, whether a word should be read from, or written to, the memory. When the binary «rw» input is logic 1, a particular word may be read from memory, depending on the address. rw = 0 indicates a write operation. For the memory unit, no additional input- or output- signals shouldbe used. (This also means that no clock signal is used for the memory itself, while such is likely to be included in systems who could use it as a component.)

Each 1-bit memory element should have input and output signals as shown rightmost in Figure 1. The bitcell («BC») is ready for a read- or write operation only when the binary value select = 1, where «rw» decides between a read- or write operation. The arrows indicate the directions of the binary signals. So, read operations make use of the «outp» signal, whereas write operations get information through the «inp» connection.

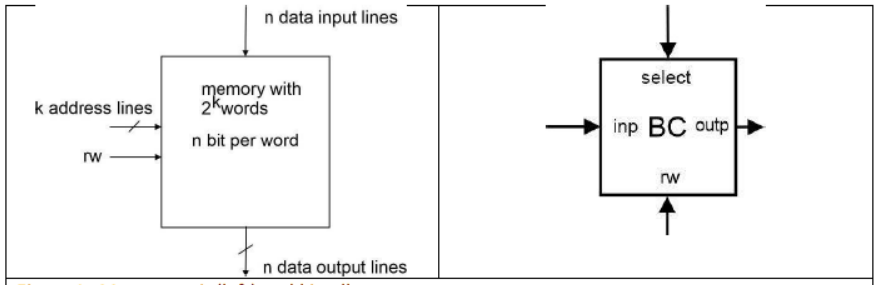


Figure 1. Memory unit (left) and bitcell.

There are 3 main design tasks for this project:

The first task is to design a 1-bit memory/bitcell, depicted to the right in Figure 1, using Boolean gates only, aiming for low leakage power / low static power consumption. The 2nd main task is to construct an 8x8 bit memory built upon the use of the 1-bit memory from the first main task.

A start could be to design a 1-bit memory cell using Boolean gates, aiming primarely for low leakage power / low static power consumption, to be embedded in the 8x8 bit memory. The 1-bit cell as well as the 8x8 bit memory should be built from Boolean gates and interconnect only, and demonstrated by using the Verilog option in Active HDL. At some point in time, properties of the bitcell should be demonstrated through AIMSpice and the 90 nm gpdk parameters.

The 3rd main task is to design a low power Finite State Machine ("FSM") using structural level Verilog, that is able to read information from, and write information to, the memory (illustrated leftmost in Figure 1).

Further specifications will be given below.

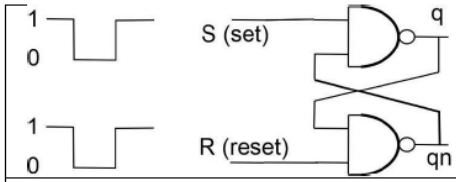
You are adviced to design your memory in a hierarchical fashion and include simulations demonstrating the functionalities of any subcomponents being parts of the memory.

Use the following naming convention and order of signals, as presented below (i0 for example, means the rightmost output bit from the «top» row in your memory:

module <name> (i0 ,i1, i2, i3, i4, i5, i6, i7 ,adr2, adr1, adr0 ,rw ,o0 ,o1, o2, o3, o4, o5, o6, o7);

For your memory, use *only structural level code*, which shows the entire memory as connected components. Examples of such components could include for example gates, latches, flip-flops, multiplexers and so on. Also, no logic gate is allowed to have a fan-in above 4.

A possible starting point for the bitcells is to let it include an SR-latch shown in Figure 2, in some way. Please note that the memory cell shown in Figure 7.5 in the book by Mano and Ciletti, is not making sense, due to what are probably typos.



S R	q qn	
1 0	0 1	
1 1	0 1	(after S=1, R=0)
0 1	1 0	
1 1	1 0	(after S=0, R=1)
0 0	1 1	(forbidden)

Figure 2. SR-latch based on 2-input NAND gates. See page 194 in the book by Mano & Celetti.

Aim at demonstrating the writing of bit patterns to the 8 different lines of the memory, followed by reading them out. Include *at least* the simple test of writing the pattern «01010101» to one of the words in your memory, and a following read operation of the same. This can be simulated in Active HDL.

For the Spice simulations of the bitcells, your simulations should include *at least* demonstrations of the following: the main functionalities for the 5 process corners. Also simulate and present figures for leakage power, for the TT and FF corners. For simplicity, we limit any gate lengths to a maximum of 300 nm, and any gate widths to 1500 nm. The upper limit for any supply voltage is 1.0 V.

For the FSM, use structural level Verilog, that writes the ASCII equivalent of the initial of the first name (or the first part) of one of the members of your group to the first address of your memory unit, and afterwards reads out the same pattern. A general FSM is shown in Figure 3. Suppose that the 8 bit pattern is ready in a register, ready for the FSM. Write and Read operations should happen in different states, and there should also be a separate Sleep state, everything partially controlled by a clock signal. Complete the FSM,illustrated in Figure 4, but keep in mind that all 3 states shown (Sleep, Read, Write) should be included. The input signal to the FSM named "R" (= "1"), should cause information to become present at the output lines of the memory from Figure 1, and the "W" signal should cause information

on the input lines in Figure 1 to be stored at a particular location indicated by the address. Make sensible decisions when designing your FSM and try and make an implementation that aims at low power, with an emphasis towards low static power contribution.

ASCII to binary conversion can be made using the following link.
https://www.rapidtables.com/convert/number/ascii-to-binary.html

For example, the letter 'a' has the binary value 01100001, and 's' has the binary value of 01110011.

The completeness, conciseness and quality of your solutions are important factors for the grading of the final report. Remember to justify and explain your design choices. The report should be handed in as a *single* pdf document (no .zip-files or references to repositorys, for example*) ideally containing all information needed to be able to reproduce your experiments*. Make schematics for all parts of your design, indicating every buildingblock and interconnect («wire»).

Deadline for your final report: Thursday 24th of November, 12:00.

Prior to the final report, you should deliver a report where you create a testbench for your chosen bitcell, where you have simulated it's functionality on structural level using only Boolean gates and interconnect using Verilog and Active HDL. In addition you should demonstrate the functionality of your bitcell using AIMSpice. Demonstrate the functionality of the bitcell for the TT, FF, SS, SF and FS corners, for 0, 27 and 70 degrees C. Delivery date for this part: Wednesday 26th of October, 12:00.

Additional information with respect to the project should be expected, through exercise hours and lectures, as well as in the project folder on BlackBoard as well as a forum on BlackBoard.

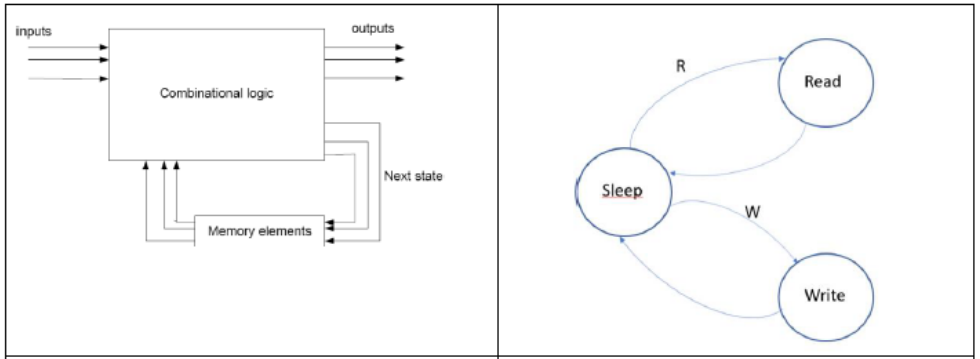


Figure 3. FSM. The clock signal is not shown.

Figure 4. State diagram for the incomplete FSM.

Good luck!

SA, NTNU, Trondheim, September 29th, 2022.