POLITECNICO DI TORINO

MASTER's Degree in ELECTRONIC ENGINEERING

MASTER's Degree Thesis

LARGE ARRAY COMPILER FOR AUTOMATIC SRAM GENERATION AND MODELING

Supervisors

Prof. DANILO DEMARCHI

Dr. TIMOTHY CONSTANDINOU

Candidate

GIUSEPPE FEROTTI

APRIL 2021

Acknowledgements

I want to dedicate some words to thank all the person that allowed this project to be carried out. In particular, I would like to thank my supervisors, Prof. Danilo Demarchi and Dr. Timothy Constandinou, for giving me the precious opportunity to work with the Next Generation Neural Interfaces (NGNI) Lab of Imperial College London. I want to thank also Dr. Peilong Feng and Mr. Andrea Misfud, who shared their knowledge and helped me during this demanding project.

Voglio dedicare questo lavoro a tutte le persone che mi sono state vicine durante questo lungo percorso. Innanzitutto ai miei genitori, che mi hanno supportato sotto ogni punto di vista, hanno condiviso con me traguardi e fallimenti e hanno sopportato la lontananza dei loro figli. A mia sorella Sara, che mi ha fatto da guida durante questo percorso, aiutandomi con scelte difficili, e che mi ha sopportato durante i periodi di stress.

Summary

To reduce time-to-market associated costs and not stall the rest of the system's development, ICs designers frequently employ memory compilers to produce the desired memory with different dimensions and configurations rapidly. Memory design methodology in the academic environment is often insufficient: most process design kits (PDKs) do not include a memory compiler. Commercial tools are also not possible for researchers due to university funding limitations, especially for modern technology nodes. Besides, the customization of memory and cell dimensions and specific peripherals is constrained by these industrial options. Moreover, its internal function and the generated memory layout is commonly inaccessible. Although open-source compilers are reported in the literature, they lack bit-cell customization, do not support the most recent cell layouts, and are not integrated in standard IC design tools. Most memory compilers provide only two possible cell dimensions, one optimized for density and one optimized for performance. Since many different size arrays can be produced (mainly constrained by system data storage need, area, power, and performance), it is practically impossible to have a perfectly tailored bit-cell for the specific system requirements with modern compilers.

This project intends to overcome this obstacle by designing a simple, modular and flexible SRAM memory compiler that uses a 6T-SRAM cell in 180 nm TSMC technology with customizable transistor widths: in this way, the system designer can select the desired transistor sizes in the cell, optimized for the system requirements and size of the array. This bottom-up approach permits to fulfil designer's specific needs without renouncing flexibility, transparency and integration with most used IC design tools in industry and academy.

Several reasons led to the choice of SRAM for this project. SRAM is vastly used in every-day electronic devices: CPUs, MCUs, ASICs, caches, and many more. Its speed, energy consumption, and reliability are often preferred characteristics in specific applications, especially when high-density memories, such as DRAM, are not needed. An added benefit of SRAM is that it is manufactured using a standard technology process, so it does not require additional masks and process steps, reducing the cost (other than different design rules). Those characteristics of

SRAM cannot be achieved by other standard memories such as DRAM and Flash. On-chip CPU SRAM memory can occupy up to 50% of the area: consequently, the design of SRAM cell and array impacts heavily on system performance, power efficiency and cost. Moreover, the static nature of SRAM memory makes timing definition a lot easier than DRAM: due to its dynamic essence (hence the need to be cyclically refreshed), several timing parameters should be respected.

This thesis starts by analysing the most used SRAM circuits in literature. 6T-SRAM cell is compared with 4T2R-SRAM one: the first is commonly prefered in the industry due to its superior stability, performance, power consumption, and simple process. The attention is focused on cell read stability and write-ability, which are conflicting requirements. Cell stability is also a primary concern in deep-submicron technology nodes: in-die transistor threshold voltage variations (that can be caused, among others, by oxide thickness and random dopants fluctuations) can strongly affect cell stability. Static Noise Margin (SNM) is one of the possible metrics to measure stability (easily obtained by SPICE DC sweep simulations), which is greatly influenced by cell transistors dimensions. SNM during the hold, read and write operations, read/write time and power/energy consumption are analysed varying cell transistor sizes and an optimal trade-off among these parameters have been found. A study on different cell layouts reported in the literature is also carried on: the two most common layout, tall cell and wide cell (also known as thin cell), are compared. Due to its lithographically friendly pattern, the fact that all the transistors have the same orientation (hence improving reproducibility and threshold voltage matching) and reduced cell height (so a decreased bit-line length/cell), thin cell layout exhibits better performances and lower power consumptions; so it has been chosen for the compiler development.

To simulate an array properly, the essential column peripheral circuits are analysed and designed. Bit-line precharge circuits are needed to have bit-lines fixed to a known voltage level before starting reading the cell. Due to its simplicity and less susceptibility to threshold variations, a static PMOS precharge circuit has been preferred to the dynamic or NMOS counterpart. The sense amplifier is another essential circuit: it amplifies the small differential voltage drop across bit-lines that occurs during a read operation, speeding up data output delay. A latched differential sense amplifier is used for this compiler because of its superior performances. Finally, the write driver circuit is analysed: it is needed to lower the voltage of one bit-line to ground during a write operation. One possible write driver circuit is designed: minimum transistor count and smallest area influenced the choice.

Cadence Virtuoso suite is one of the primary tool used in the industry for designing ICs. Since the developed compiler's main characteristic is flexibility, it is designed to be perfectly integrated with the aforementioned software. In pursuance of this, Cadence proprietary SKILL scripting language is used to develop

the compiler software. SKILL scripts perform cell, peripherals and array schematic cell-view creation and cell layout cell-view creation. By means of a bash script then, DRC, LVS and parasitic extraction of the wanted SRAM cell are carried out using Calibre software. Once the extracted cell-view is available, a test-bench is automatically created, instantiating SRAM array, column peripherals and voltage generators. Test-bench timing parameters can be customized by the user, thanks to which minimum setup/hold times, pulse widths and word-line period can be easily found. Then, a complete write/read SPICE simulation of the test-bench is performed using an OCEAN script (Cadence SKILL extension for simulation with Spectre), and the results are stored. Thanks to this approach, the designer can access simulation results easily and perform optimization of the cell, peripheral circuits sizing, array dimensions and timing parameters. The user can also decide to use or not the extracted cell-view during the simulation: this can be helpful to have quick estimations, very useful in preliminary design stages and to know how the designed array impacts system performance and power consumption. Although for large arrays, where delay and consumption due to bit-lines and word-lines wiring can be predominant, the use of the extracted cell-view could be necessary to have reliable simulation results.

Lastly, compiler performances are analysed in generating and simulating different array sizes. Compilation time is not affected by array dimension, since the software must perform DRC, LVS, and, particularly, parasitic extraction (i.e. the most computationally intensive) only for the single cell layout and not for the entire array. Simulation time strongly depends on array dimension: considering that each cell comprises six transistors, even small arrays can contain up to tens of thousands of transistors. For this reason and due to limited server capabilities, only small and medium sized arrays (up to 4 kbit) have been simulated, for which optimal timing parameters have been found. Moreover, read/write delays, peak currents, absorbed energies, and static currents are analysed for the different array sizes and compared for Typical-NMOS/Typical-PMOS (TT) and Slow-NMOS/Fast-PMOS process corners. The last is a worst-case process corner for this particular SRAM design, in which a faster static PMOS precharge transistor can slow down bit-lines discharge carried out by slower NMOS transistors during read and write operations.

Table of Contents

List of Tables

List of Figures

Listings

Chapter 1

Introduction and Motivations

SRAM is vastly used in every-day electronic devices: CPUs, MCUs, ASICs, caches, and many more. Its speed, energy consumption, and reliability are often desired characteristics in certain applications, especially when high-density memories, such as DRAM, are not needed. ITRS (and then IRDS) predicted SRAM to use more than 90% of high-performance SoC area in next years [1]. Since its ubiquity, it is crucial to integrate SRAM arrays in electronic systems easily. Moreover, designing and manually laying out even a small array can be prone to errors and very time-consuming, besides the fact that it requires specialized know-how.

In order to reduce time-to-market related costs and not to slow down the development of the rest of the system, ICs engineers often use memory compilers to generate the desired memory block. In fact, the repetitive architecture of memories allows relatively simple design automation that can produce different dimensions and configurations rapidly, even if it is difficult to accomplish with different technology nodes. In the academic environment, there is a lack of memory design methodology: most process design kits (PDKs) does not come with an included memory compiler. When a memory compiler is publicly available and without cost, it really only supports a non-fabricable process technology. Commercial options can also be not feasible for researchers due to university funding limitations. In addition, the customization of memory dimensions and specific peripherals is constrained by these industrial options, and the internal function/layout is generally confidential. Numerous leader industries and foundries have indeed supplied their clients with memory compilers. Once a non-disclosure agreement (NDA) is signed, memory compilers typically let users access only simulation data and pin placement. Most of the memory compilers, which do not permit the user to customize the cell and peripherals designs, are constrained by licences and typically involve a very high

price; hence for most university research programs, this makes them practically inaccessible [2][3][4]. Although an open-source compiler exists in literature [2], it lacks bit-cell customization and does not support the most recent cell layouts. Most memory compilers provide only two possible cell dimensions, one optimized for density and one optimized for performance. Since many different size arrays can be produced (mainly constrained by system data storage need, area, power, and performance), it is practically impossible to have a perfectly tailored bit-cell for the specific system requirements with modern compilers.

This project aims to handle some of these problems: a simple, modular and flexible memory compiler is designed, by means of which the user can customize bit-cell transistors' widths in order to optimize the array for power, area, and performance, a desirable feature in academic research. This bottom-up approach permits to fulfil designer's specific needs without renouncing flexibility, transparency and integration with IC design existing tools. The tool produces the desired bit-cell layout, carries out DRC and LVS, extracts parasitics, and performs SPICE simulation of the desired array in order to evaluate power consumption and performance. Although it does not generate a complete array layout, it can be beneficial to have quick and reliable estimations. Moreover, thanks to its modularity and flexibility, it can be functional during SRAM architecture design exploration.

The thesis is organized as it follows. In Chapter 2, memory circuits overview is reported, from SRAM state-of-the-art cell and layout to peripheral circuits. In Chapter 3, SRAM memory circuits are designed, explaining the stability, timing, and energy consumption trade-offs in details. Chapter 4 provides a detailed overview of the compiler design, its structure and its usage. In Chapter 5, compiler performances and simulation results are described, where three different array sizes are compared in terms of performances and consumption. Chapter 6 concludes the work, summarizing all the technical efforts made for the development of this project and possible future works.

Chapter 2

Memory Circuits Overview

2.1 Memory Classification

In the design of CMOS circuits, memory components represent essential parts and are crucial for many devices. They are being used for a vast spectrum of systems with varying requirements. While all memory is used to store and retrieve information, depending on how the stored data is accessed, they can be categorized into three main groups: random access memory (RAM), serial access memory, and content addressable memory (CAM), as described in Figure 2.1.

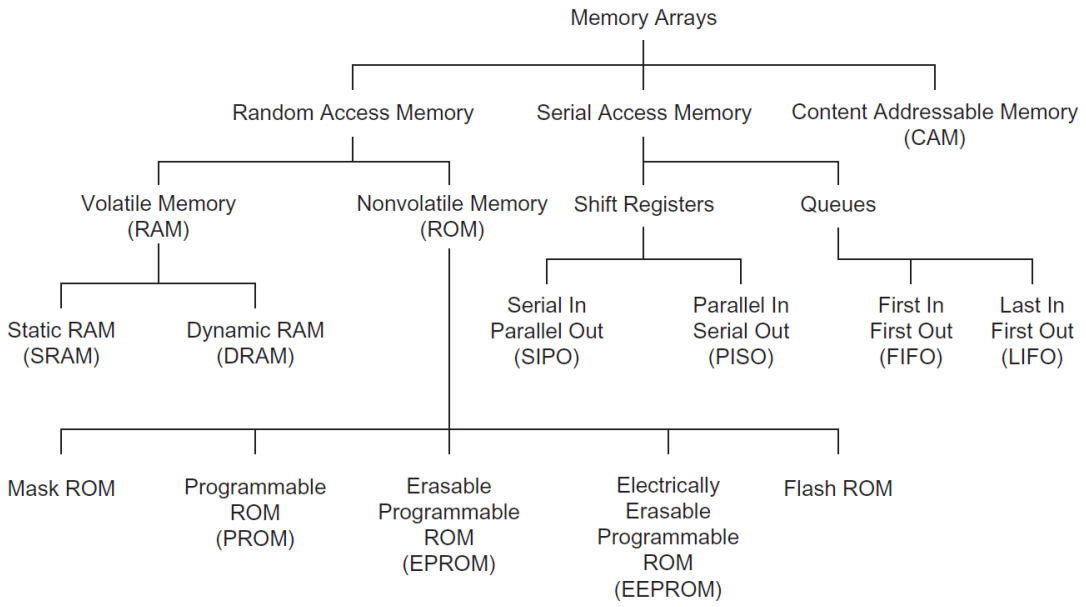


Figure 2.1: Memory classification depending on how data are accessed [5]

While data stored in a RAM are accessed through an address that corresponds to a physical memory location, serial access memory is read and written sequentially; thus, there is no need for an address. Instead, CAM identifies which address(es) stores the datum/data equal to the given key; this feature is precious in server switches for MAC address checking/routing [5]. RAMs can be further classified into volatile and nonvolatile memories (RAM/ROM distinction is misleading; this denomination is still used for historical reasons). Volatile memories can maintain information as long as power is supplied, while nonvolatile memory can retain data even without power almost endlessly. Given that nonvolatile memories are much slower than volatile ones, but they can achieve much higher densities (at a lower cost), they are more suited for secondary storage elements rather than system main memories.

Since this discrepancy in access times, cost per bit, and density, Personal Computers (PC) architecture often implements the well-known memory hierarchy (Figure 2.2), in which fast, small and costly memories (such as register files, caches and main memory) are inside or near the CPU, while slow, large and cheaper memories (such as hard drives or solid-state disks) are distant from the processor. The use of a memory hierarchy is essential in order to fill the gap between processing unit and memories cycle times while guaranteeing reasonable performances, power consumption, and costs. In embedded systems, where the main concern is power consumption and costs, and there is no need for large memories, very small secondary storage is used and lower levels of memory hierarchy is employed; DRAM is used as main memory when performances are power are not the main concern [6], otherwise small SRAM (to prevent cost increase) is preferred.

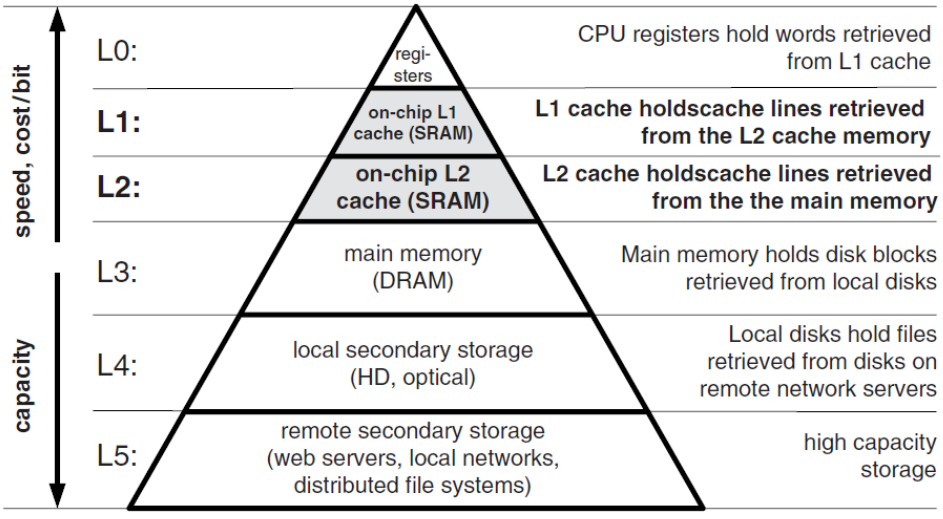


Figure 2.2: Example of memory hierarchy of a personal computer [1]

2.1.1 SRAM and DRAM: a comparison

Before describing SRAM in details, it is natural to compare SRAM ad DRAM, given their similarities and uses in integrated circuits. As pictured in Figure 2.3, these two memories show a significant difference in cell topology.

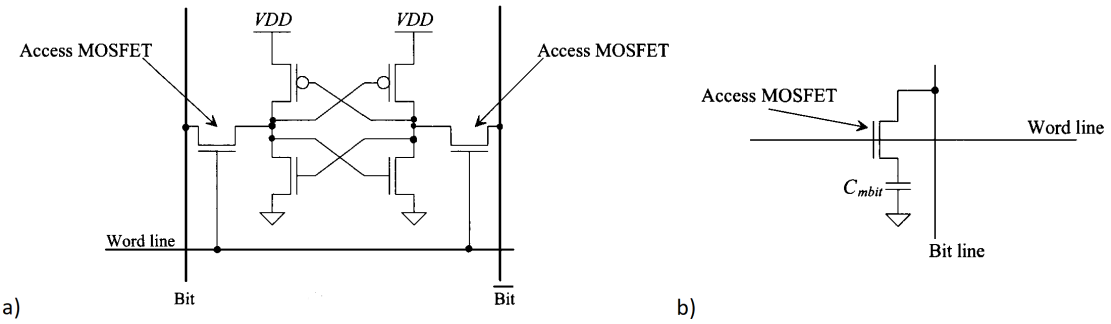


Figure 2.3: 6T-SRAM cell (a) and 1T1C-DRAM cell (b) (adapted from [7])

While most of SRAMs use six transistors (two n-type MOSFETs for accessing cell data nodes, two n-type and two p-type MOSFETs which form two cross-coupled inverters), DRAM cell uses only one n-type access transistor and a capacitor which stores the information. This makes DRAM a much denser and cheaper memory with respect to SRAM. In SRAM the datum is maintained by means of two cross-coupled inverters (also known as latch): thanks to the positive feedback loop the information is not lost as long as power is supplied. On the contrary, DRAM uses the charge stored in a capacitor to represent information; since capacitors and access transistors have leakage currents, the charge disappears with time, and so the voltage across the capacitor tends to degrade. Even reading a DRAM cell makes the charge to vanish, in fact the operation is also called disrupting reading. Due to these reasons, a DRAM cell needs to be periodically refreshed in order not to lose its datum. Refreshing means to re-write data in the cells through special circuits (amplifier and switch) every certain time (usually in the order of milliseconds) or right after a read operation in order to not corrupt the information [5][7][8][9]. The fact that DRAM needs to be periodically refreshed represents a significant drawback: during refreshing, the cell cannot be accessed, and several clock cycles may be needed for this. Furthermore, some power is dissipated during a refreshing operation; thus, DRAM is not well suited for very low-power applications. Another determinant factor is the storing capacitance value: it determines not only reliability, access time and power consumption but also leakage current; since refreshing time (and so clock cycle) depends on the latter, it is challenging to integrate a DRAM array in a system with strict time requirements; this makes DRAM array design more system dependent. On the contrary, a relatively large

SRAM array can easily reach access times comparable to logic clock cycles; in fact, they are commonly used as caches and register files. Even though the SRAM cell area is larger than DRAM, speed and power benefits surpass cost and area disadvantages in these applications [10][8].

A modern-day high-density DRAM cell is depicted in Figure 2.4. It uses a polysilicon plug (or trench) capacitor: it exploits the verticality of very high aspect ratio holes to reach large surfaces, hence large capacitances, and an Oxide-Nitride-Oxide dielectric film. The poly-plug is directly connected to the access transistor drain and acts as a capacitor terminal, and the other is the heavily p-doped substrate. High aspect ratio trenches can be obtained employing very costly specialized equipment (such as Deep Reactive Ion Etching tools), and it requires high quality and uniform deposition of the dielectric. Due to these reasons, the manufacturing process needs to be carefully tailored and can be unavailable with more traditional CMOS processes [5][7]. Additional equipment, lithographic steps and masks can make the DRAM process very costly, especially for low-volume production. Lower cost DRAM cells are available (such as buried capacitor cell) but have a larger area and are not suited for embedded DRAM. The trench capacitor's main advantages are that the capacitance value can be increased without enlarging the cell area, and that high-temperature process steps are performed before logic fabrication; this makes it suitable for embedded DRAM application [10]. These advanced process steps carry special Design Rules, which are very dependent on technology and they need to be respected during cell layout for the compiler development.

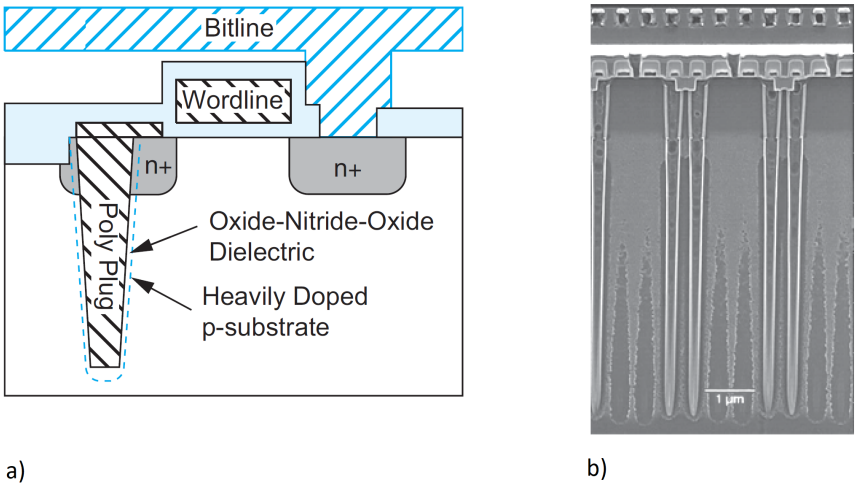


Figure 2.4: DRAM trench capacitor cell: Drawing (a) and Cross-section SEM view (b) (adapted from [5])

Furthermore, DRAM needs additional signals to be properly controlled, especially for precharge and refreshing, which are critical for a correct operation. This complicates timing requirements: in DRAM, tens of timing parameters have to be respected, particularly for an asynchronous design [11].

Due to the above considerations, SRAM is the best of the two memory categories for developing a memory compiler. The two types of memory have their strengths and limitations; both are commonly employed in most computing systems and are necessary hardware components. Having defined similarities and differences between SRAM and DRAM, and stressed the motivations of the choice of SRAM to develop the memory compiler, now the discussion proceeds with an accurate description of SRAM cell, peripherals and architecture.

2.1.2 SRAM Cells

Understanding the functionality of SRAM cells is essential for a more thorough description addressed later in this section. The circuit topology of a 6T-SRAM cell is shown in Figure 2.5. It is made of six transistors: two NMOS (Q5, Q6)

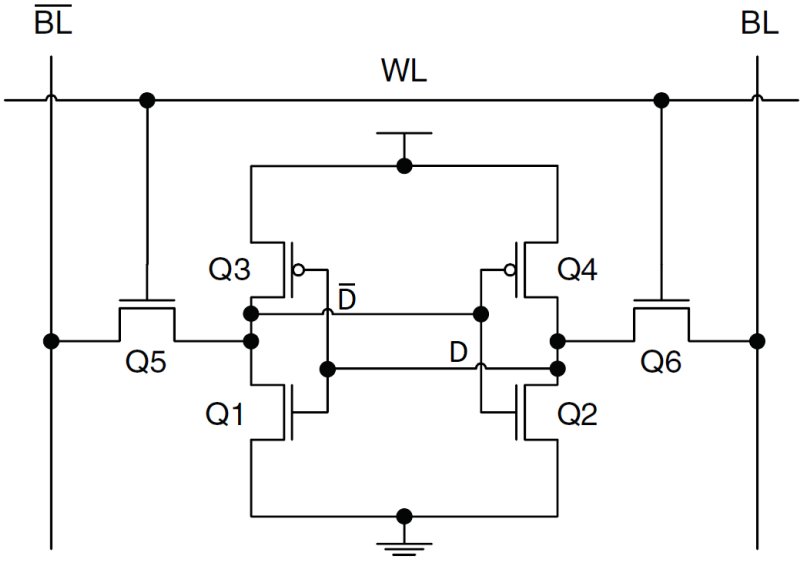


Figure 2.5: 6T-SRAM cell (adapted from [1])

are the access transistors, while two NMOS (Q1, Q2), called pull-down or driver transistors, and two PMOS (Q3, Q4), called pull-up or load transistors, form the bistable latch (so two cross-coupled inverters), responsible for maintaining the information through its positive feedback loop. Without the feedback, any stored information would be lost with time due to leakages, as it happens with

DRAM. Through the access transistors, internal latch data voltages (i.e. the stored datum and its complemented value) are transferred to the bit-lines (BL and $\overline{BL}$) during a read operation. On a strictly theoretical level, there is no need for two bit-lines and hence two access transistors, but having BL and $\overline{BL}$ permits to perform a differential sensing, so speeding up the read operation and guaranteeing an excellent common-noise rejection, at the expense of an increased cell area. Moreover, having two sides of the cell that can be accessed improves cell stability during the read/write operation. The cell is accessed by rising the word-line signal (connected to access transistors' gates) to V_{DD}, so letting a current flow from bit-lines to internal data nodes or vice-versa. Before any read/write operation, bit-lines are precharged to a known voltage, usually V_{DD}. During a read operation, supposing to have D equal to V_{DD} and $\overline{D}$ equal to 0 V, BL remains to its precharged value, and $\overline{BL}$ is discharged through the series of access and pull-down transistors (Q5 and Q1 in this example). When the involved bit-line is discharged to a sufficiently low voltage level, the small differential voltage across bit-lines is amplified to the full-swing output level. During a write operation, one of the two bit-lines is fully discharged by a write driver circuit. Assuming that the stored datum is a logic '1' and that a logic '0' is to be written, the internal datum node is discharged through access transistor Q6; once the node is discharged to a sufficient voltage level, the inverter made of Q1-Q2 starts switching, and the positive feedback loop forces the other inverter Q3-Q4 to switch to the opposite value. Now that the read and write operations are briefly described, it is clear why the bit-lines need to be precharged to a high voltage value: both read and write relies on transferring a logic '0' to/from bit-lines, and the access transistors are NMOS, that can pass a strong logic '0' but a weak logic '1'. If the bit-lines were precharged to a logic '0', then the read/write operation would rely on charging bit-lines/internal nodes to logic '1'; hence read/write delays would be higher due to the use of NMOS access transistor. PMOS transistors are not used for accessing the cell since they would require a higher area due to degraded holes mobility and the presence of additional n-wells in the cell layout. As shown later, the sizing of access, pull-down, and pull-up transistors is fundamental to allow correct and reliable read/write operations. Since bit-lines are already precharged by external circuitry, pull-up transistors' only task is to provide low-to-high transition during switching and to maintain internal nodes to V_{DD} during data hold [1][8].

Another well-known SRAM cell topology is the 4T2R one, depicted in Figure 2.6, also called polysilicon resistor load SRAM cell. It employs only four NMOS transistors, two for accessing the cell and two for pull-down of the inverters, and two high-resistive polysilicon loads in place of pull-up transistors. The immediate drawback is that ratioed inverters present smaller gains around voltage transfer characteristics switching point; this causes lower noise margins and increases the time needed to switch from the metastable point. The clear benefit with respect to

6T-cell is the area, around 30% smaller [1]. The load resistor's primary purpose is to counterbalance the leakage currents of pull-down transistors during data hold, i.e. maintaining the correct datum. The upper bound on the resistance value is fixed by the necessity to have a pull-up current at least two orders of magnitude higher than the pull-down leakage current to prevent data loss [8] and by minimum low-to-high switching times [1]. The lower bound is given by minimum noise margins and maximum static power consumption. Unfortunately, this cell presents a much higher static current (due to the presence of a finite load resistor); up to three orders of magnitudes higher than 6T-cell [8]. Due to these conflicting requirements, designing a 4T2R-SRAM cell is becoming more difficult, especially with deep sub-micron technologies. Technological steps, such as doping by implantation and annealing of polysilicon resistors, cause significant resistance values variability and do not let resistor area to scale down as transistors one. Moreover, the transition from constant-voltage scaling to constant-field scaling produced unsustainable trade-offs for power dissipation and cell stability. Another issue is the need to introduce additional process steps to produce load resistors, which are not present in a standard CMOS process; this makes 4T2R cell incompatible with embedded memory applications, such as caches in CPUs or SoCs. Due to these motivations, the 6T-SRAM cell is the preferred choice in standard deep sub-micron scaled systems [1] and it is chosen for the development of this memory compiler.

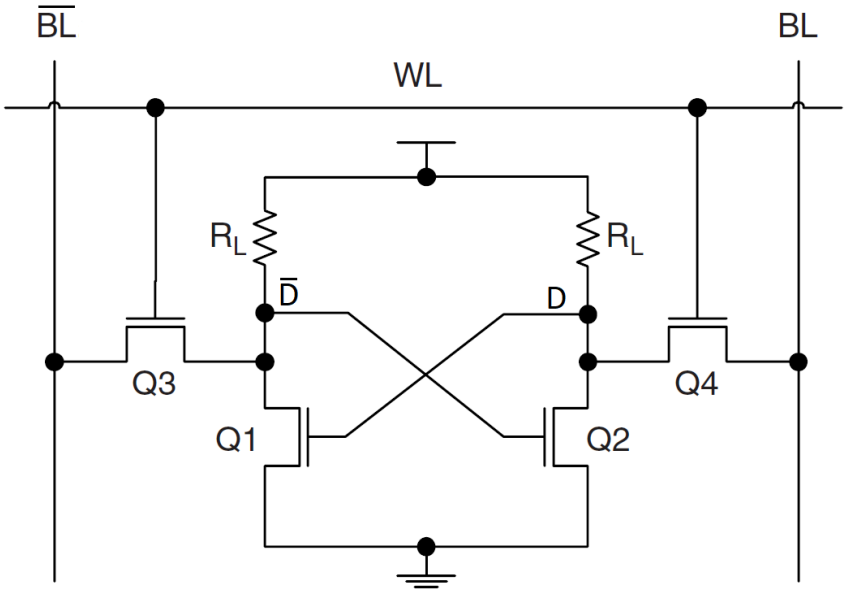


Figure 2.6: 4T2R-SRAM cell (adapted from [1])

2.2 SRAM Operations and Stability

Attention is now focused on a thorough analysis of the 6T-SRAM cell during the read and write operation. Mathematical models based on simple equivalent circuits are derived and discussed. Definition of cell stability and one of its possible metrics, i.e. Static Noise Margin (SNM), is also analyzed since it is a significant concern in deep submicron technology nodes.

2.2.1 Read

The read operation is examined. Figure 2.7 shows a simplified 6T-SRAM cell circuit in which D is equal to a logic '1' and $\overline{D}$ is equal to a logic '0'. Transistors

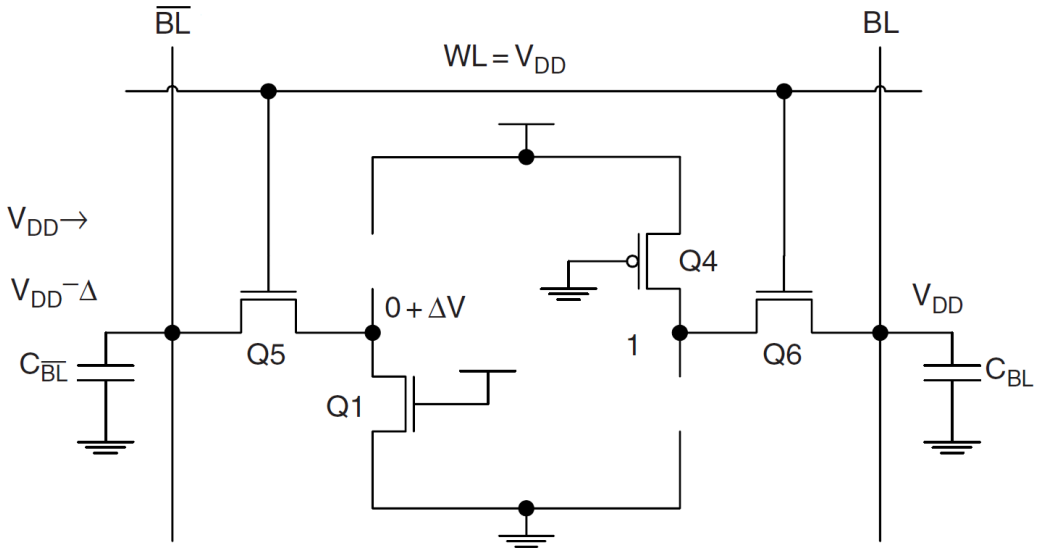


Figure 2.7: Simplified 6T-SRAM cell circuit during read operation (adapted from [1])

Q2 and Q3, which are in the off state, are idealized with open circuits. BL and $\overline{BL}$ parasitic capacitances, which are the effects of all the cells connected to the same columns and bit-lines wiring capacitances, are modelled by lumped capacitances C_{BL} and $C_{\overline{BL}}$. Since both bit-lines are precharged to V_{DD} before starting a read operation, only $\overline{BL}$ is discharged while BL remains at V_{DD}; hence we focus our attention only on the left side of the circuit. When the word-line is asserted high, $\overline{BL}$ starts discharging through the series of Q5 and Q1. Q1 and Q5 form a voltage divider whose input is initially at V_{DD}, so its voltage output cannot remain at 0 V, but it will rise to a certain value ΔV. The problem is that this voltage acts as an input to the second inverter Q3-Q4: if this voltage rise passes the inverter threshold,

the cell starts switching and the datum is lost, so causing a disruptive read. To have some safety margin and be sure not to overwrite cell datum, the voltage ΔV should be lower than NMOS threshold voltage $V_{th,n}$: to prevent this from happening, Q5 equivalent resistance should be higher than Q1 one [8]. Assuming a simple quadratic MOSFET model and ignoring body effect and short-channel effects, DC currents of Q1/Q5 can be derived and must be equal (ignoring off transistor contribution)[8]:

$$\begin{aligned} K_{n,Q5}\left(\left(V_{DD}-\Delta V-V_{th,n}\right)V_{DSAT,n}-\frac{V_{DSAT,n}^2}{2}\right)= \\ =K_{n,Q1}\left(\left(V_{DD}-V_{th,n}\right)\Delta V-\frac{\Delta V^2}{2}\right) \end{aligned} \tag{2.1}$$

where Q5 is assumed to be at the edge of saturation (and $V_{DSAT,n}$ is given by velocity saturation model in short-channel devices) and Q1 in triode region (since its low $V_{DS}=\Delta V$). Equation (2.1) can be rearranged to express ΔV as a function of all the other parameters [8][1]:

$$\Delta V=\frac{V_{DSAT,n}+CR\left(V_{DD}-V_{th,n}\right)-\sqrt{V_{DSAT,n}^2\left(1+CR\right)+CR^2\left(V_{DD}-V_{th,n}\right)^2}}{CR} \tag{2.2}$$

where CR is the cell ratio and is equal to:

$$CR=\frac{W_1/L_1}{W_5/L_5} \tag{2.3}$$

Equation (2.2) is plotted in Figure 2.8, considering TSMC 180 nm parameters extracted from its PDK documentation. The red line represents the NMOS threshold voltage $V_{th,n}$, approximately equal to 0.4 V. As can be observed, the CR needs to be almost greater than one to ensure a ΔV lower than $V_{th,n}$. A larger CR makes the read current, sunk by pull-down transistors, higher, providing a faster bit-line discharge rate and a lower read delay. As shown later in this section, higher cell ratios give the cell also more stability during a read operation [12][13]. The main disadvantage is a larger cell, so reducing bit density [1][8]. The above analysis holds true also for the opposite case (D ='0' and $\overline{D}$ ='1') due to cell symmetry.

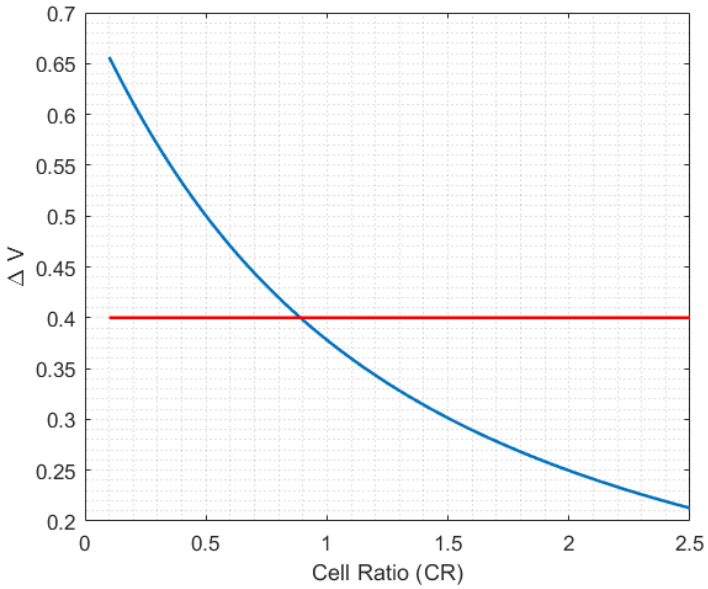


Figure 2.8: ΔV (blue line) as a function of Cell Ratio; red line depicts $V_{th,n}$

2.2.2 Write

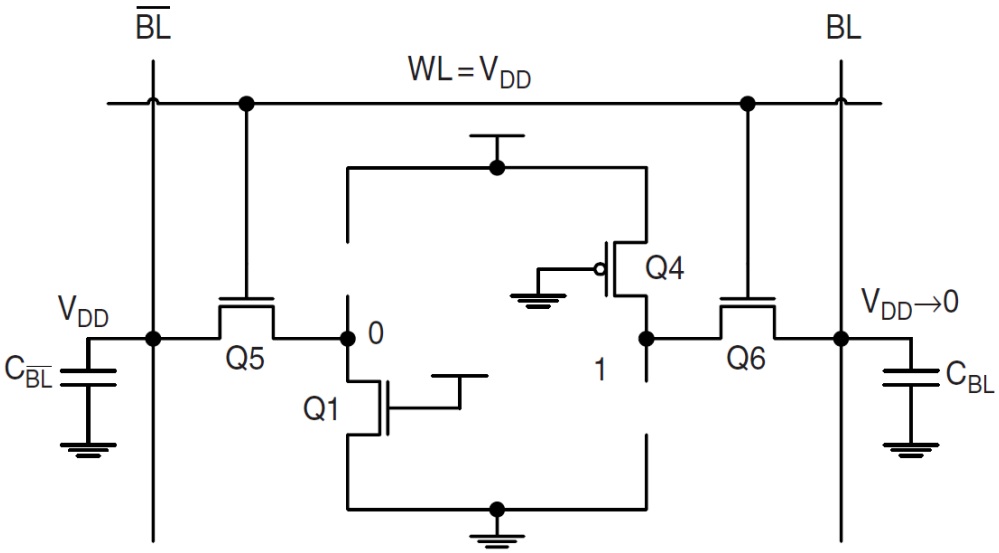


Figure 2.9: Simplified 6T-SRAM cell circuit during write operation (adapted from [1])

The write operation is examined through the simplified circuit shown in Figure 2.9. As before, D is equal to a logic '1', $\overline{D}$ is equal to a logic '0' and transistors Q2-Q3 can be omitted. Supposing that a logic '0' is to be written into the cell, BL is discharged to ground by an external write driver circuit through the access transistor Q6, once the word-line is asserted. Before the effective toggling starts, D and $\overline{D}$ nodes (i.e. Q1 and Q4 gates, respectively) remain at the same voltages. In theory, once the bistable begins switching, this condition is no more valid; we can ignore this for the purpose of a simplified analysis. The first aspect to notice is that in this example the writing operation starts at D node (i.e. the node storing a logic '1'); this is because $\overline{D}$ node voltage cannot be raised high enough due to non-disruptive reading requirement (low ΔV) and its resulting sizing restriction (weak Q5 and strong Q1)[1]. So the attention is focused on the right side of the circuit: once the toggling starts, the feedback mechanism will force the other side of the cell to flip. As already stressed in Section 2.1.2, pull-up transistors only task is to provide low-to-high transition during switching and maintain internal nodes to V_{DD} during data hold. The cell can be overwritten if the internal node voltage V_D is pulled sufficiently down (below NMOS threshold $V_{th,n}$). The problem is that transistor Q4 tries to keep V_D to V_{DD}: to prevent this from happening, transistor Q6 equivalent resistance should be lower than Q4 one. Assuming a simple quadratic MOSFET model and ignoring body effect and short-channel effects, DC currents of Q4/Q6 can be derived and must be equal (ignoring off transistor contribution)[8]:

$$K_{n,Q6}\left(\left(V_{DD}-V_{th,n}\right)V_D-\frac{V_D^2}{2}\right)=K_{p,Q4}\left(\left(V_{DD}-|V_{th,p}|\right)V_{DSAT,p}-\frac{V_{DSAT,p}^2}{2}\right) \tag{2.4}$$

where Q4 is assumed to be at the edge of saturation (and $V_{DSAT,p}$ is given by velocity saturation model in short-channel devices) and Q6 in triode region (since its low $V_{DS}=V_D$). Equation (2.4) can be rearranged to express V_D as a function of all the other parameters [8][1]:

$$V_D=V_{DD}-V_{th,n}-\sqrt{\left(V_{DD}-V_{th,n}\right)^2-2\frac{\mu_p}{\mu_n}PR\left(V_{DD}-|V_{th,p}|\right)V_{DSAT,p}-\frac{V_{DSAT,p}^2}{2}} \tag{2.5}$$

where PR is the pull-up ratio and is equal to:

$$PR=\frac{W_4/L_4}{W_6/L_6} \tag{2.6}$$

Equation (2.5) is plotted in Figure 2.10, considering TSMC 180 nm parameters extracted from its Process Design Kit (PDK) documentation. The red line represents the NMOS threshold voltage $V_{th,n}$, approximately equal to 0.4 V. As can be

observed, the PR needs to be almost lower than 3.5 to ensure a V_D lower than $V_{th,n}$. This is a very conservative analysis since the cell can also start switching when V_D goes below $V_{DD} - |V_{th,p}|$, hence higher pull-up ratios are possible, but it can make the cell very sensible to process variations. Normally, the pull-up ratio would be set to one with minimal size access and pull-up transistors to guarantee low cell area. $PR = 1$ ensures write-ability due to mobility differences between access and pull-up transistor, but it could be a problem for slow NMOS/fast PMOS process corners or a V_{DD} higher than the nominal one. On the contrary, increasing PR (with constant CR) centres latch inverters switching point, improving read and hold stability, but can degrade the write time to the point where the cell becomes not writable. As can be noticed, write-ability and read stability are opposing design constraints [1][8]. The above analysis holds true also for the opposite case (D ='0' and $\overline{D}$ ='1') due to cell symmetry.

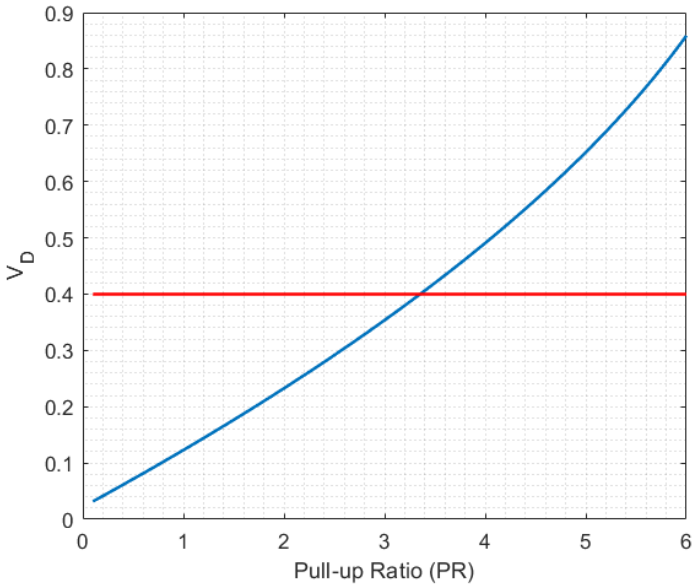


Figure 2.10: V_D (blue line) as a function of Pull-Up Ratio; red line depicts $V_{th,n}$

2.2.3 Cell Stability and SNM Analysis

Current SRAMs aim to maximize the number of bits while preserving low energy consumption, high speed and reliability. These goals involve continuous scaling of the transistors' sizes; constant field scaling also forced a reduction of supply voltage to prevent high power consumption and reliability issues (such as gate oxide breaking) [1]. Supply voltage and transistor dimensions scaling impose several challenges for the CMOS manufacturing process; this is particularly true for large SRAM arrays, where random dopant in-die and within-die fluctuations cause large threshold voltage variability [14]. Short-channel effects, polysilicon minimum dimensions, and line edge roughness also have a significant impact on threshold voltage variations, even among very close transistors [15]: this can lead, among others, to data stability issues and asymmetric cell behaviour. SRAM stability is greatly degraded by all the above factors and presents concerning worsening with technology scaling [14]. Static Noise Margin (SNM) is one of the well-known metrics for measuring stability. «Noise Margin (NM) is the maximum spurious signal that can be accepted by the device when used in a system while still maintaining the correct operation»[1]. When the noise is applied for enough time for the device to be affected, it is called static (ideally is applied for an infinite time). If a finite noise impulse is fed to the system, the noise is called dynamic, and the used metric is Dynamic Noise Margin (DNM); when the noise pulse width becomes large, DNM decreases asymptotically to SNM. DNM is always greater than SNM, i.e. considering SNM represents a worst-case analysis [16].

To evaluate SNM, latch butterfly curves are commonly used; they are obtained from inverter transfer curves: since two back-to-back inverters form a latch, one inverter presents the standard transfer curve, while the other a mirrored transfer curve (since its output is the input of the other inverter and vice-versa) [17]. A very convenient method for deriving SNM from butterfly curves consists of measuring the full square that can be inscribed in the two lobes of the butterfly curve: since the cell, in reality, is never perfectly symmetrical, the side of the smaller of the two squares is the worst-case SNM [17]. The main advantages of this method are that a simple analytical expression linking SNM to SRAM cell parameters can be derived and that DC SPICE simulations can easily measure it. One drawback is that it cannot be measured experimentally because one should have access to internal cell nodes. Other metrics for measuring stability, such as the N-curve [18], are more suited for chip testing, but they do not have a relatively simple analytical relation with cell parameters [12]; hence SNM is the preferred metric in this project.

The method for deriving SNM consists of inserting voltage noise sources in the latch feedback loop: as shown in Figure 2.11, these noise sources must have opposing polarities for describing a worst-case scenario. Practically speaking, SNM is the minimum noise voltage required to move inverters transfer curves away from each other to the point where they do not have a stable point [5][12].

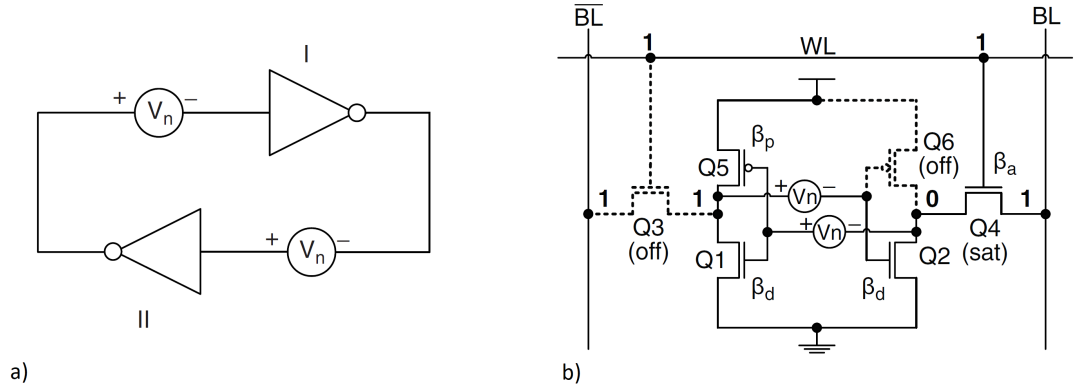


Figure 2.11: Bistable latch with opposing polarities noise sources (a) and equivalent SRAM cell (b) (adapted from [1])

A way to measure SNM with DC simulations involves considering latch butterfly curve, normally plotted in $x - y$ coordinates (i.e. $V_{in} - V_{out}$), and plotting it in a counterclockwise 45° rotated $u - v$ system, as depicted in Figure 2.12 [1][17][19].

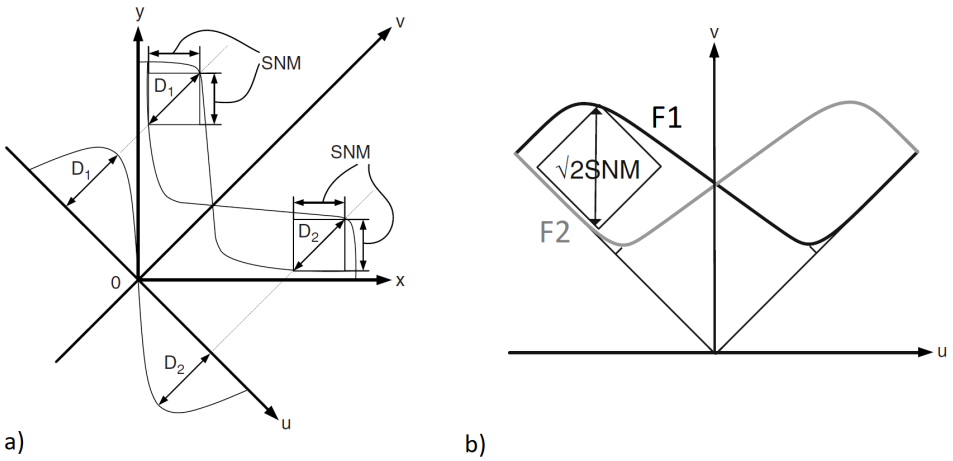


Figure 2.12: Butterfly curve (a) and its 45° rotated plot (b) measuring inscribed square diagonals (F1 and F2 are inverters curves in the new coordinates) (adapted from [1], [19])

Supposing $y = F_1(x)$ and $y = F_2'(x)$ are the normal and mirrored inverters transfer curves respectively, we can apply a coordinates transformation in the new $u - v$ system:

$$x = cos(45°)u + sin(45°)v = \frac{1}{\sqrt{2}}u + \frac{1}{\sqrt{2}}v \tag{2.7}$$

$$y = -sin(45°)u + cos(45°)v = -\frac{1}{\sqrt{2}}u + \frac{1}{\sqrt{2}}v \tag{2.8}$$

substituting Equations (2.7) and (2.8) in $y = F_1(x)$ yields to:

$$v = u + \sqrt{2}F_1\left(\frac{1}{\sqrt{2}}u + \frac{1}{\sqrt{2}}v\right) \tag{2.9}$$

For the second inverter F_2 the procedure is similar but with swapped $x - y$ axis:

$$x = -\frac{1}{\sqrt{2}}u + \frac{1}{\sqrt{2}}v \tag{2.10}$$

$$y = \frac{1}{\sqrt{2}}u + \frac{1}{\sqrt{2}}v \tag{2.11}$$

substituting Equations (2.10) and (2.11) in $y = F_2(x)$ yields to:

$$v = -u + \sqrt{2}F_2\left(-\frac{1}{\sqrt{2}}u + \frac{1}{\sqrt{2}}v\right) \tag{2.12}$$

Equations (2.9) and (2.12) are expressions of v as a function of u for both inverters constituting the latch (plotted in Figure 2.12b). They can be easily obtained by circuital implementations employing «voltage-dependent voltage sources in a feedback loop» [17], as shown in Figure 2.13.

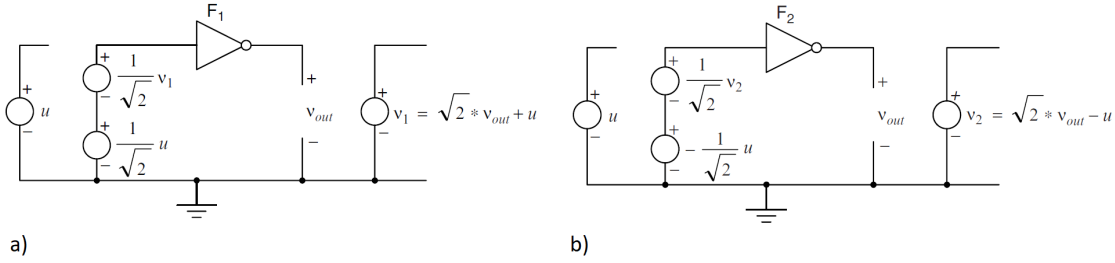


Figure 2.13: Circuital implementations of Equations (2.9) (a) and (2.12) (b) (adapted from [1])

The solution of Equations (2.9) and (2.12) are v_1 and v_2 in Figure 2.13 (a) and (b), respectively. Once these are obtained, the difference $v_1 - v_2$ is calculated (plotted

diagonally in Figure 2.12a, similar to a sine wave). The absolute values of the global maximum and minimum of this curve are the diagonals of the maximum squares inscribed in the two lobes of the butterfly curve. The SNM is the minimum side of the two squares, calculated as $\frac{D_{min}}{\sqrt{2}}$.

Seevink et al. [17] presented also an analytical derivation of SNM based on the circuit showed in Figure 2.13b. Here the complete derivation is omitted for conciseness; the resulted formula is:

$$SNM = V_{th,n} - \left(\frac{1}{k+1}\right) \cdot \cdot \left(\frac{V_{DD} - \frac{2r+1}{r+1} V_{th,n}}{1 + \frac{r}{k(r+1)}} - \frac{V_{DD} - 2V_{th,n}}{1 + k\frac{r}{q} + \sqrt{\frac{r}{q}\left(1 + 2k + \frac{r}{q}k^2\right)}} \right) \tag{2.13}$$

where:

$$\begin{aligned} r &= \frac{\beta_{driver}}{\beta_{access}} \quad and \quad q = \frac{\beta_{pull-up}}{\beta_{access}}, \\ k &= \left(\frac{r}{r+1}\right)\left(\sqrt{\frac{r+1}{r+1-V_s^2/V_r^2}} - 1\right), \\ V_r &= V_s - \left(\frac{r}{r+1}\right) V_{th,n}\ , \\ V_s &= V_{DD} - V_{th,n} \end{aligned} \tag{2.14}$$

From the above equations the most important aspect is that SNM is independent on absolute values of β, but it rather depends only on their ratios. That is why technology scaling and its increase in transistors' β values does not improve SNM. Besides, V_{DD} scaling can make SNM worst from node to node. Particular values of r and q can be found that make SNM independent on V_{DD} due to the opposite signs of the coefficients in Equation (2.13). Changing q and r can make SNM dependency on V_{DD} positive or negative. In general, in order to maximize SNM, r and $\frac{q}{r} = \beta_p/\beta_d$ should be maximized. The drawbacks of doing this are an increased cell area and a poor write-ability due to increased pull-up ratio (as seen in Section 2.2.2). Finally, the direct dependence on threshold voltage makes SNM dependent on temperature (as temperature increases, $V_{th,n}$ and SNM decrease). [1][17].

As can be noticed, only SNM during read operation has been evaluated. Similar considerations can be carried out for hold operation (i.e. when word-line is not asserted), but the main differences lie in butterfly curves appearance: as shown in

Figure 2.14a, when the cell is accessed, the zero-level of internal nodes is no more at ground (as described in Section 2.2.1); this makes the SNM of a read-accessed cell worse than a not-accessed one [20]. During a write operation, the cell acts as a monostable circuit, i.e. it presents a single stable point (corresponding to the datum is being written) (Figure 2.14b) [18]; a write SNM can be still derived through simulation and reflects the write-ability of the cell [18][21].

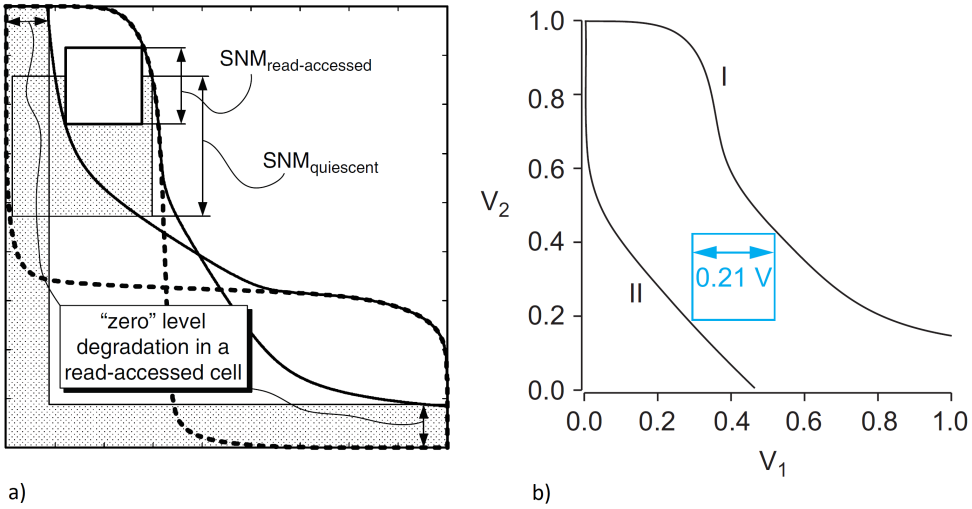


Figure 2.14: Read SNM vs. Hold SNM (a) and Write SNM (b) (reprinted from [1][5])

2.3 SRAM Cell Layout

Many possible 6T-SRAM cell layouts are reported in literature. Figure 2.15 shows different layouts: this categorization divides the layout on the base of inverters placement and access transistors orientation [22][23]. Type-1a cell, also called "Tall Cell", was commonly used in industry until 90 nm technology node. From 90 nm on, a more lithographically friendly layout (type 4), called "Thin Cell", has been adopted [24]. Type 5 cell, called "Ultra-Thin Cell", has been recently introduced for sub-22 nm nodes [25]. The latter showed a larger area and some delay penalty down to 32 nm process node [23]. Since this project employs 180 nm node, the analysis is now focused on the two most used layouts for this technology, i.e. tall and thin cell.

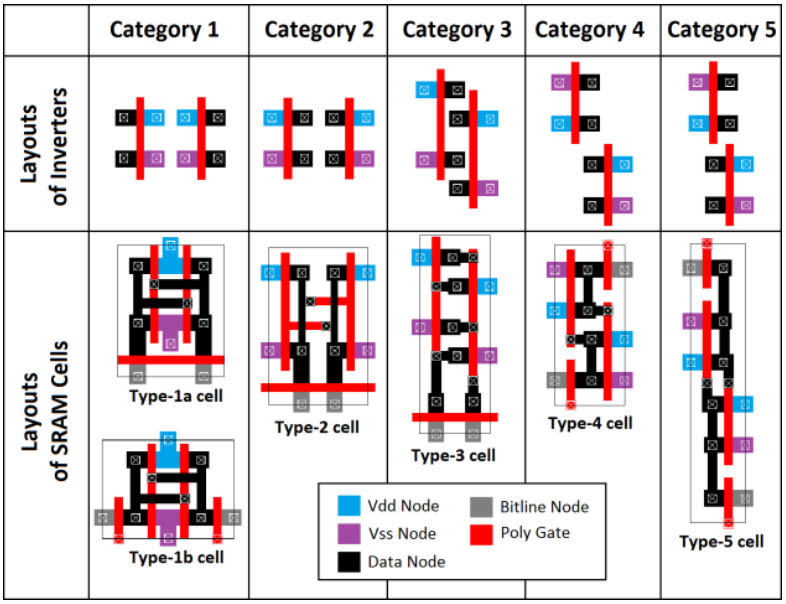


Figure 2.15: Possible 6T-SRAM cell layouts depending on inverters and access transistor layout (reprinted from [23])

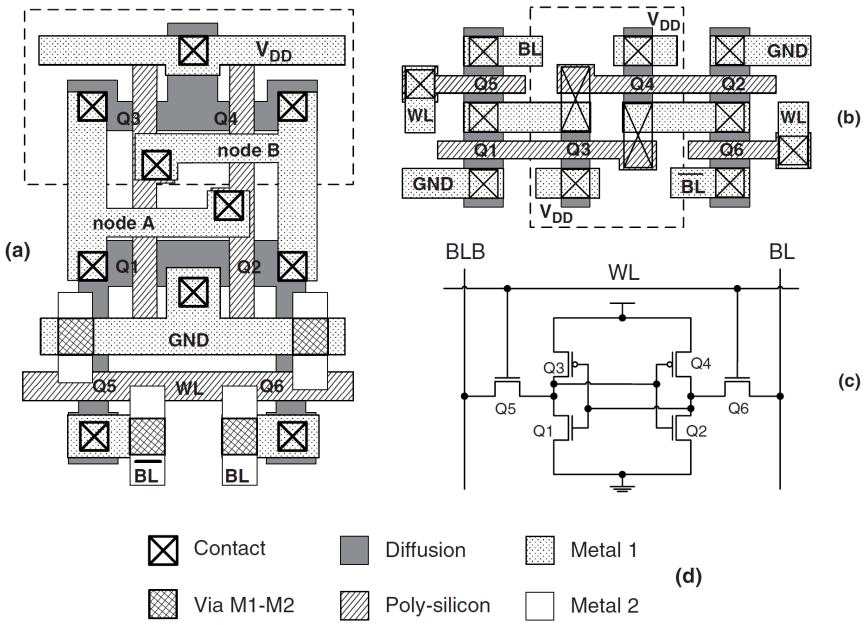


Figure 2.16: Tall Cell (a) and Thin Cell (b) comparison; 6T-SRAM schematic (c) and layout layers legend (d) are also reported (reprinted from [1])

2.3.1 Tall vs. Thin Cell

Figure 2.16 shows a comparison between tall and thin cells. It is evident that the tall cell presents a much higher number of corners of diffusion and M1 layers. Since lithography, the technique used to transfer pattern using UV-light sources, lenses and masks to the wafer, did not scale as much as transistors [1], the reproducibility of complex patterns became a problem; this led to a reduction of pattern quality and process yield [5]. On the other hand, the thin cell was introduced to mitigate these problems: its slender layout, without corners in M1 and diffusion layers, led to a major yield enhancement. Furthermore, all transistors in the thin cell have the same orientation: this is very important for threshold voltage and width/length matching. In fact, process variation (such as dopants or oxide thickness gradients) and optical effects in corners can make matching really challenging [1]. Another advantage is the shorter cell bit-line: this reduces the wiring capacitance and can reduce read/write delays. Besides, wide cell has word-lines routed through polysilicon, while thin cell uses metal layers which have a much lower sheet resistance so reducing word-line signal propagation delay [1][24]. A comparison of all the types of cell in Figure 2.15 has been carried out for 65, 45 and 32 nm process nodes: thin cell always shows comparable or lower read/write delays, minimum cell area, and comparable power consumption [23]. The fact that all transistors have the same orientation is also beneficial for designing a customizable cell: changing any transistors' width will produce only a change in cell width, thus greatly simplifying custom layout design. For the above reasons, the thin cell has been chosen for the development of this project.

2.4 SRAM Architecture

SRAM array is only a part of the complete memory system. Even if this project focuses on the array, it is important to have an overview of the complete memory architecture. Figure 2.17 depicts a typical SRAM architecture block diagram [1]. Usually the number of memory locations is much higher than the number of bits of a word; this would make inappropriate to build a memory array with rows equal to memory locations and columns equal to word width, because it would produce a very "long" and "thin" array. Therefore, this would cause very long bit-lines (thus worsening delays and power consumption) and a non-squared circuit footprint. To prevent this, a 2D-structure is always employed: address bits, which are used to access the wanted memory location (usually a word with a certain number of bits), are split into two parts: the most significant bits (X in Figure 2.17) addresses a row through a row decoder, while the least significant bits (Y in Figure 2.17) selects the wanted word among all the columns through a column decoder/multiplexer. Consequently, memory arrays with aspect ratios close to one

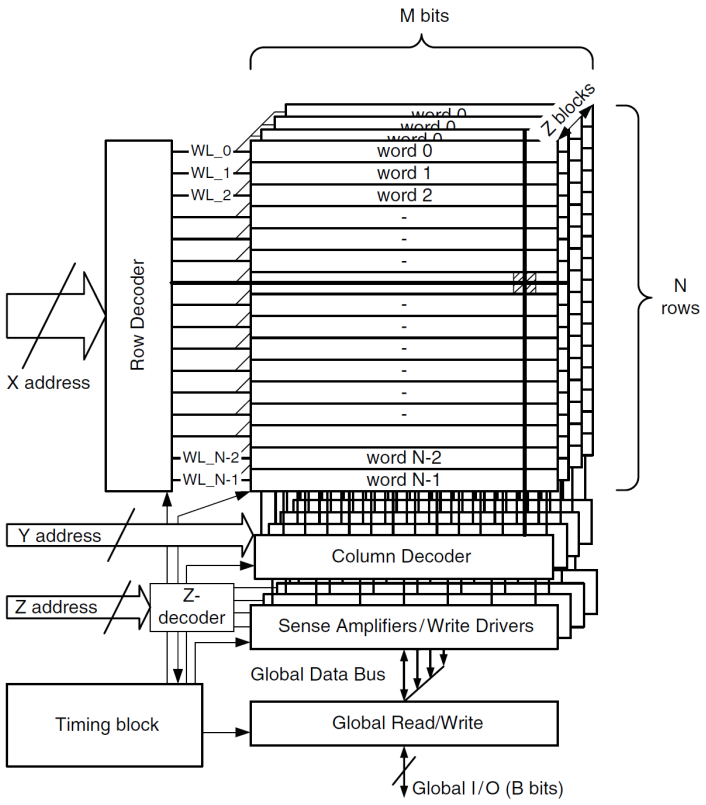


Figure 2.17: SRAM architecture block diagram (reprinted from [1])

can be employed. Column MUXs let sense amplifiers and write drivers circuits to be shared among multiple columns (usually a power of 2), i.e. reducing area and power. Memory array can be further divided into blocks, accessed by address most significant bits (Z in Figure 2.17) through a block decoder. Memory partitioning can be very advantageous: firstly, power and delay can be drastically reduced due to diminished capacitance (one could reduce also V_{DD} for a further quadratic power reduction but with constant delay) [26]; secondly, CPUs often accesses data in a page manner, hence each page can be easily associated to each block [1]. The main disadvantage is the area overhead introduced by block decoder, additional column peripherals, and a more complex control logic. Clearly this technique is convenient only with large memories, otherwise any added circuitry would increase static power consumption with a negligible dynamic power minimization. I/O buffers are often needed to drive large I/O data busses connected to the rest of the system. Timing block is responsible for correct operation during access, precharge, read and write operations. Modern-day stand-alone SRAMs are often asynchronous and self-timed. Asynchronous SRAMs leverage the burden of a clock distribution system, reducing

routing complexity and power related to clock switching capacitance. Moreover, deep sub-micron nodes lead to great variability in delays; taking into account all possible variations would lead to a very relaxed clock frequency to ensure correct operations in all process corners. A self-timed SRAM, and, in particular, one employing replica dummy loop technique [27], shows more robustness against these variations. In embedded SRAMs, where the memory has to be used as a cache, synchronous pipelined arrays are needed to match CPUs cycle time at the cost of increased latency [1].

The analysis now proceeds with column peripheral circuits, needed to ensure a correct operation and test the produced memory array.

2.4.1 Precharge Circuit

As already discussed, these circuits are needed to precharge bit-lines to a known high voltage. Precharge circuits can be categorized depending on transistors (PMOS or NMOS) and dynamic or static behaviour (Figure 2.18). NMOS transistors precharge bit-lines to $V_{DD} - V_{th,n}$; this makes them unsuitable for deep sub-micron nodes since threshold voltage variability can make bit-lines precharged to different voltage levels. On the contrary, PMOS transistors precharge bit-lines to full V_{DD}. Dynamic circuits, controlled by a precharghe signal (ϕ or $\overline{\phi}$ in Figure 2.18), permits to divide timing phases into precharging and read/write operation. On the other hand, static circuits constantly try to charge bit-lines even if the cell is being read or written; this imposes a constraint on how much strong (i.e. low resistive) precharge transistors can be, because they have a contention with pull-down cell transistors during a read operation [5]. Instead, stronger precharge transistors

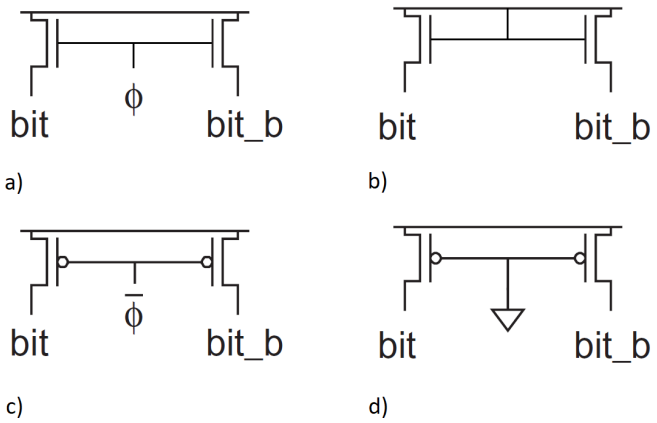


Figure 2.18: Possible precharge circuits: NMOS dynamic (a), NMOS static (b), PMOS dynamic (c) and PMOS static (d) (adapted from [5])

provide a higher current and precharge speed, essential especially after a write operation where bit-lines are completely discharged. Precharge recovery time (i.e. the time needed to fully charge bit-lines after being discharged) greatly influences operating frequency. Equalizing pass-transistors connected among bit-lines are often added to ensure equal precharge voltages before a read operation is started [1]. For this project, a PMOS static precharge circuit has been chosen for its simplicity, since there is no need for additional control signals.

2.4.2 Sense Amplifier

Sensing is maybe the most critical operation in a memory system. When big arrays are involved, the higher bit-lines capacitance would make a large-signal sensing too slow and power hungry [5]. In order to speed-up sensing and not to completely discharge bit-lines, the small differential voltage is amplified to full-swing output levels. Sense amplifier's design constraints can include: delay, gain, power consumption, minimum sensed differential voltage, minimum common-mode rejection ratio, voltage/current offsets and layout area [1][8][9]. Figure 2.19 shows the basic sense amplifier used in integrated circuits, i.e. a current mirror load with differential pair amplifier. When a sufficient differential voltage is present at its inputs, Sense Amplifier Enable signal (SAE) is raised in order to properly bias the amplifier and the amplifier senses which voltage signal is decreasing; the output inverter is needed to bring internal differential pair output signal, with limited output swing, to full-swing.

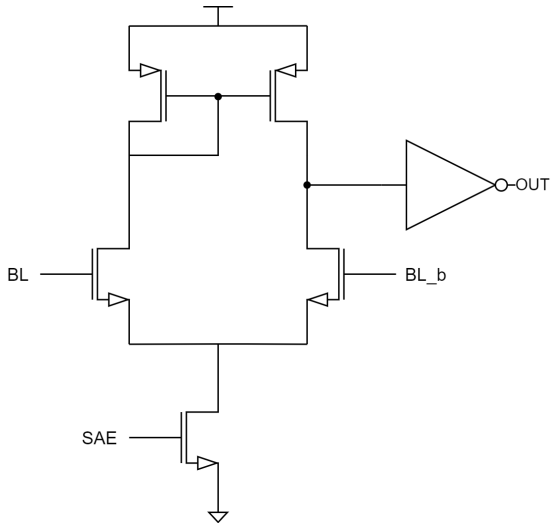


Figure 2.19: Current mirror load differential sense amplifier (adapted from [5])

The gain of this amplifier is:

$$A = -g_{m,n} \left(r_{o,n} \parallel r_{o,p} \right) \tag{2.15}$$

where $g_{m,n}$ is the NMOS transconductance and $r_{o,n}$ and $r_{o,p}$ are the NMOS and PMOS equivalent output resistances, respectively. Due to gain-bandwidth product limitations, the gain of the amplifier is not set to a very high value (usually around ten), since it is more important to have a quick estimation. To increase the gain one could increase bias current (paying attention to not reducing output resistances or increasing power consumption too much) or widen differential pair transistors. To further increase gain without renouncing to speed and to have better output swing, two-stage differential sense amplifier can be used (Figure 2.20). The clear drawback is an increased area and power consumption, while the sensitivity (minimum differential input voltage, typically around 0.5 V) of the amplifier is enhanced [7][8]. SAE signal is used to bias sense amplifier only when needed, so reducing static power consumption.

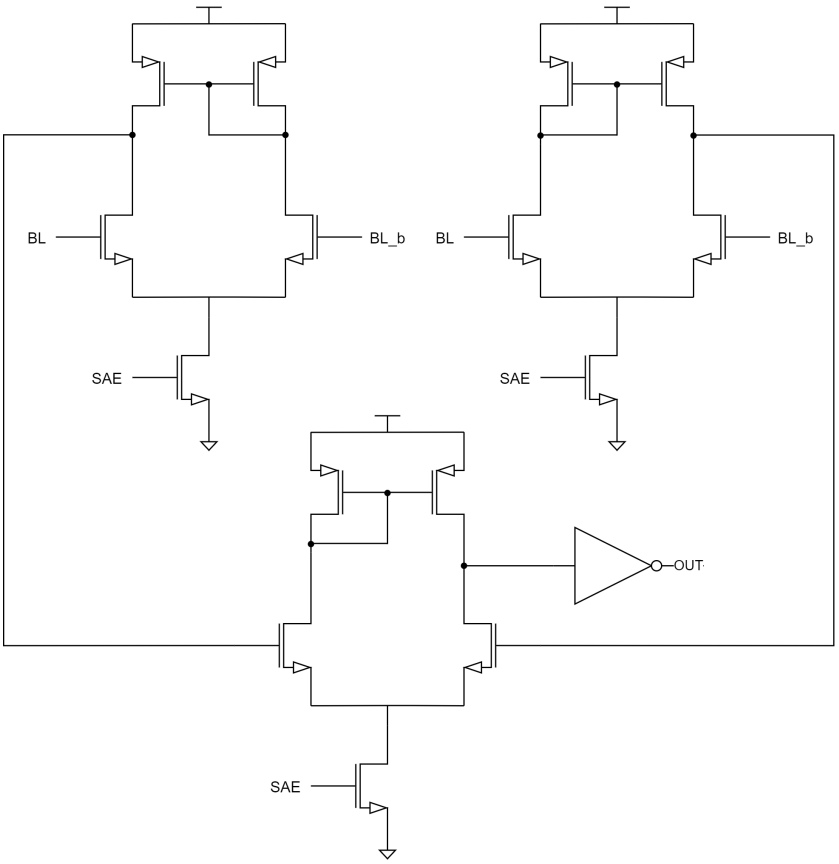


Figure 2.20: Two-stage differential sense amplifier (adapted from [5])

A totally different approach is used in latched sense amplifiers (Figure 2.21). Since a latch, when in metastable state (i.e. inputs with equal voltages), presents a very high gain, it can be exploited to perform differential sensing with very small voltage differences [28][7]. When one of the two output starts going low, the latch feedback mechanism enhances discharge rate and speeds-up sensing. When SAE is low, precharged bit-lines are connected to latch inputs. When a read operation is started and a sufficient differential voltage is created, SAE signal is asserted high; this causes two effects: NMOS pull down transistor is on and lets a current flow to ground, secondly, PMOS pass-transistors isolates bit-lines from sense amplifier outputs. The second effect is crucial, because, when the latch performs sensing, one of the two outputs is completely discharged; without isolation PMOS transistors, also the high-capacitive bit-lines would be discharged wasting an incredible amount of power and time for each read operation. The situation is diametrically opposite in DRAMs, where bit-lines needs to be discharged to ensure data refresh after a disruptive read operation [8]. In order to increase speed, $^{W}/_{L}$ ratios and $C_{GS,n}/C_{GS,p}$ of latch transistors should be maximized and minimized, respectively. For the NMOS pull-down transistor, a higher $^{W}/_{L}$ ratio lowers its drain voltage before and after sensing, which again reduces delays but highers the amount of current, that in turns leads to increased power consumption [9][7]. This topology has been chosen for this project because of its superior speed, reduced power consumption, and its simpler design. In order to have memory of previous sensing and drive output capacitance, a Set-Reset (SR) output latch is added, as shown in Figure 2.22 [7].

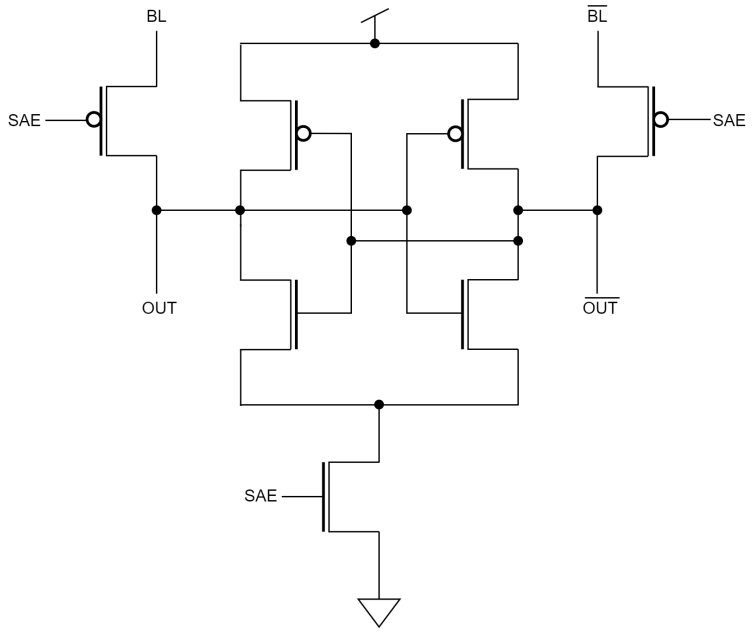


Figure 2.21: Latched differential sense amplifier (adapted from [28])

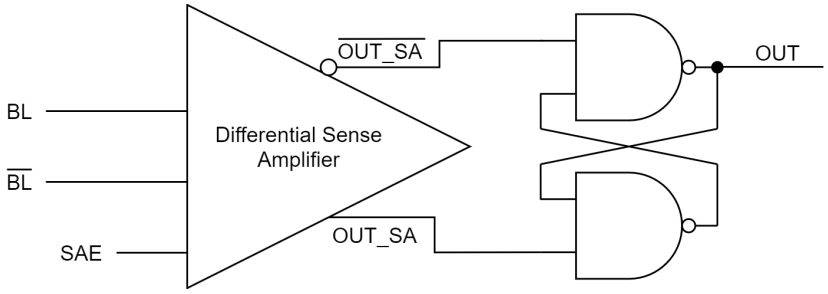


Figure 2.22: Adding a latch to sense amplifier outputs (adapted from [7])

2.4.3 Write Driver

The write driver circuit has the task to rapidly discharge one of the bit-lines during a write operation and is normally activated by a write enable signal (WE). Figure 2.23 shows possible implementations of this circuit [1]. Figure 2.23a shows a circuit where bit-lines are actively driven to V_{DD} or ground thanks to inverters (1 and 2); the data are transferred to bit-lines through transmission gates (TG1 and TG2), activated by WE signals. The use of transmission gates is mandatory since here both '1' and '0' have to be driven correctly. In Figure 2.23b WE signal controls upper pass-transistors (Q3 and Q4); when WE is low, the write driver outputs are in high-impedance, thus disconnected from bit-lines. When WE signal is high, upper transistors are on and one of the two bit-lines is grounded by a lower pass-transistor (Q1 or Q2), enabled by input datum through inverters 1 and 2. The circuit in Figure 2.23c exploits two AND gates, enabled by WE signal and input datum through inverters (1 and 2), to turn on one of the two pass-transistors (Q1 and Q2) and ground one bit-line. Both circuits (b) and (c) do not need transmission gates, since only one bit-line is set to ground during a write operation, while the other remains at its precharged value. The circuit in Figure 2.23b has been chosen for the development of this project due to its lower transistors count.

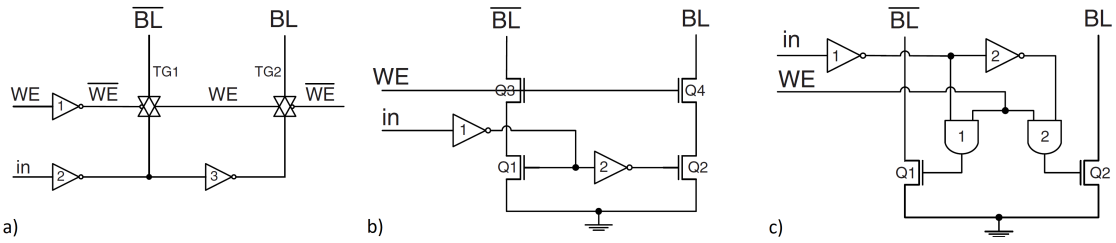


Figure 2.23: Possible write driver circuits (reprinted from [1])

Chapter 3

SRAM Circuits Design

The thesis now focuses on SRAM cell and peripherals design. Even though more attention is drawn to a careful cell design since the project proposes designing a compiler with a customizable bit-cell, peripheral circuits are nevertheless essential to properly test the produced array schematic; hence, these circuits are also analyzed and designed. In the first sections, an overview of the used Electronic Design Automation (EDA) tools and programming languages is provided.

3.1 Software Framework

Many EDA tools are available in the industry, and each one is specialized for some type or part of the design flow. Cadence Virtuoso is widely considered the leader in full custom IC digital and analogue design. Calibre, by Mentor Graphics, is generally used for physical verification (e.g. DRC, LVS, LPE and others). Other tools are more suited for digital synthesis (such as Synopsys) or radio-frequency design (such as PatchWave from Keysight). In this project, Cadence Virtuoso suite and Calibre are used for schematic entry and simulation, layout and verification, and post-layout simulation. SKILL and OCEAN, two Cadence proprietary scripting languages, are used to design the memory compiler and perform simulations of the produced test-bench.

3.1.1 Cadence Virtuoso and Calibre

All Cadence design files are arranged as libraries, cells and views. The library is the highest hierarchy level, with each library usually including many designs (arranged by the user). A cell is a design "tree" containing several views: schematic, symbol, layout and calibre view (a schematic view that contains all parasitic elements).

For a schematic generation, Virtuoso Schematic Editor XL is used (Figure 3.1). The schematic is the first stage in development with full-custom or semi-custom IC design flows right after the choice of circuit topology and its required components. The schematic is a device-level design that shows how the components are related and their properties, such as capacitance or resistance of passive components, and width/length of transistors. The tool lets the user set variables as components properties so that parametric analysis can be performed to find the parameters' optimal set. In the schematic, the user defines input/output pins, ideal connections among component, and nets. Once the schematic of the cell is designed, a symbol representing it can be created so that the cell can be instantiated hierarchically in other cells or test-benches.

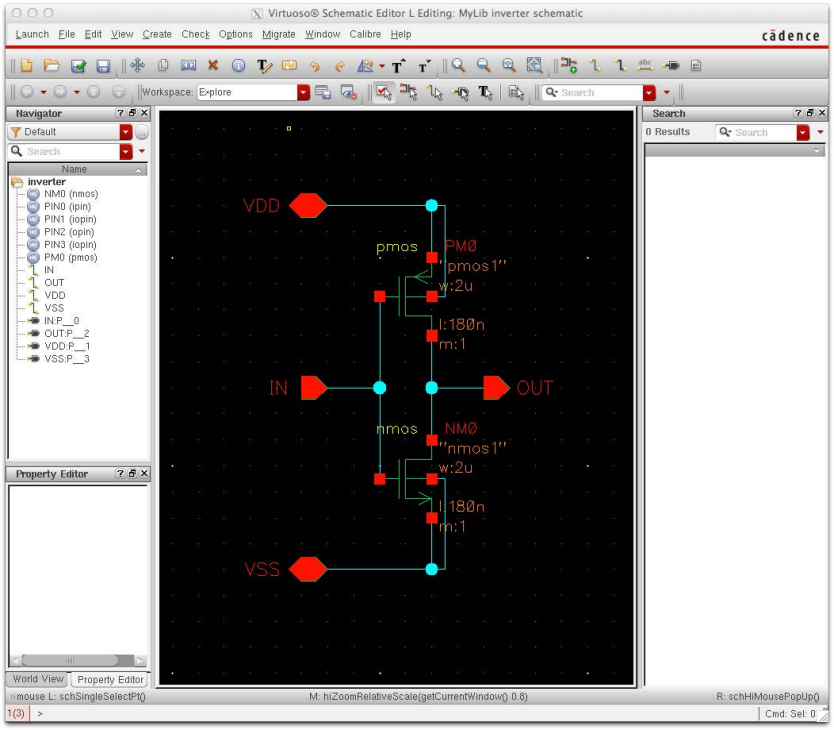


Figure 3.1: Virtuoso Schematic Editor XL with an inverter design example

A further critical aspect of IC design flow is the circuits' simulation to guarantee that they operate correctly and have optimized performance. To validate the circuits designed in this project, Virtuoso ADE (Analog Design Environment) XL tool is used (Figure 3.2). It can rely on two different simulators: Spectre, Cadence equivalent of SPICE, and Ultrasim, mixed-signal SPICE based transistor-level simulator (it can detect digital and analogue portions of the system and apply different models to speed-up simulation time). In this project, only Spectre simulator is used. It can perform various analysis (transient, AC, DC, noise,

Periodic Steady-State and others), depending on designer needs. ADE XL lets the user decide types of simulation, perform parametric analysis by sweeping parameters, set specifications (e.g. delay, energy and stability) to be met, run multiple simulations of different test-benches with the same parameters, and run corner simulations for the process, voltage and temperature (PVT) variations.

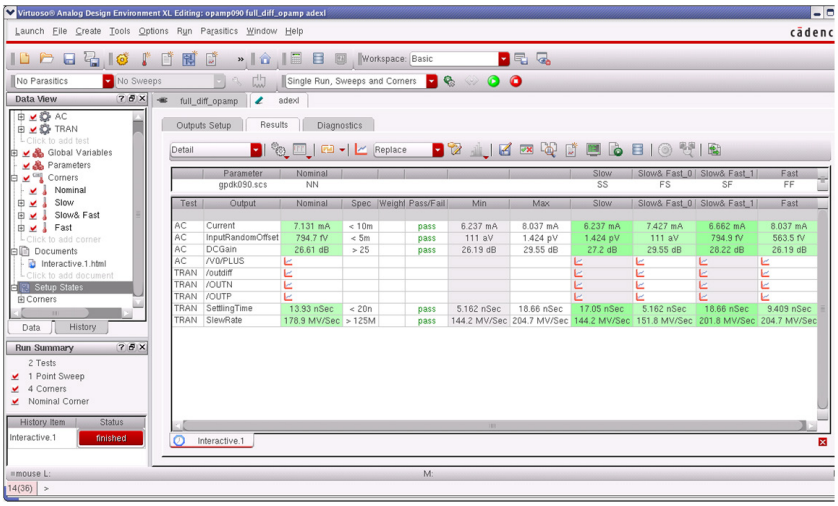


Figure 3.2: Virtuoso ADE XL with multiple corner simulation example

Another tool used in this project is Virtuoso Layout XL (Figure 3.3). It allows the user to design physically the circuit schematic, instantiating transistors given by PDKs, drawing geometries of different layout layers (mainly implantation, n-well, oxide, polysilicon, contacts, and metal), creating pins on different layers (useful for automatic routing). With the same tool, also parametrized cells (PCells) can be designed. A PCell allows the user to stretch geometries in all the wanted layers according to specific parameters [29]. Since this work aims to have customizable SRAM bit-cells, PCell is the perfect tool to implement them.

During and after the layout design, it is essential to check if it reflects the connections and the properties of the corresponding schematic and if it respects all the physical rules, such as minimum width, minimum spacing and minimum density, that make the design manufacturable. Calibre (Figure 3.4) is used to perform layout vs schematic checking (LVS) and design rules checking (DRC). After the layout design is completed and checked, Calibre can also perform parasitic extraction (PEX) to estimate parasitic resistances and capacitances of the used layers, thanks to which a much more accurate post-layout simulation can be run. This accuracy is particularly needed for large memory arrays, in which long bit-lines and word-lines have a significant impact on delay and power consumption. DRC, LVS and PEX can be also performed in a batch mode (so without opening a GUI), particularly useful for an automated memory compiler.

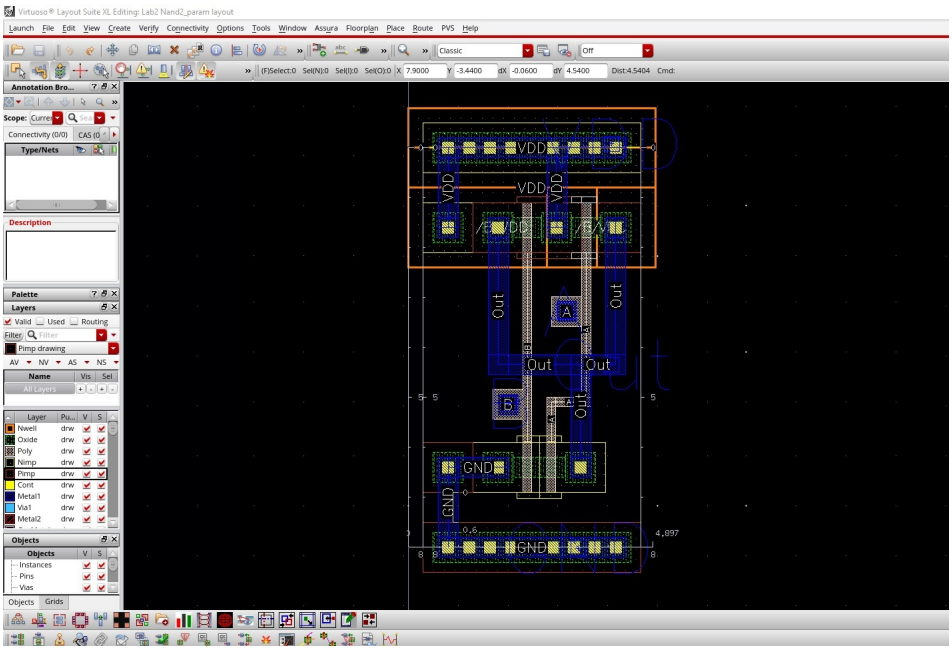


Figure 3.3: Virtuoso Layout XL with a NAND design example

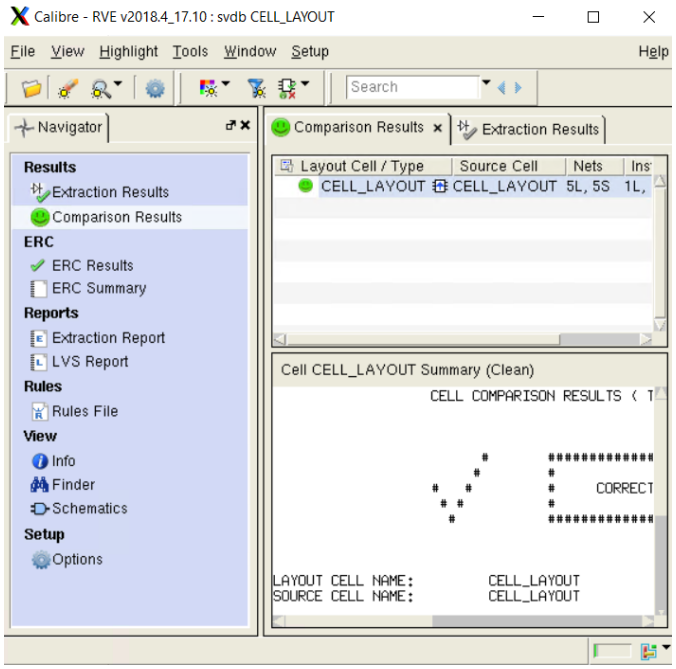


Figure 3.4: Calibre successful LVS screenshot

3.2 SRAM Cell Design

Even though this work aims to design a compiler with a customizable cell, it is crucial to understand how transistors' sizing for this particular technology node influences stability, performances, and energy consumption to ensure a good starting point for the user.

DC simulations employing the circuits described in Section 2.2.3 are used to evaluate stability. Figure 3.5 shows the test-bench for evaluating SNM while a logic '0' is being written in the cell (i.e. WSNM); this is proved by bit-lines voltage generators ($BL = 0\,\text{V}$ and $\overline{BL} = V_{DD} = 1.8\,\text{V}$). A write '1' SNM could be evaluated by merely exchanging BL and $\overline{BL}$ voltage generators, but this is not needed since the cell is symmetrical. Hold and read SNM (HSNM and RSNM) are evaluated by leaving bit-lines floating with $WL = 0\,\text{V}$ and $1.8\,\text{V}$, respectively. HSNM and WSNM test-bench schematics are not reported for conciseness since all those circuits are similar. The common aspect of all those circuits is the implementation of Equations (2.9) and (2.12): by sweeping the DC source at the node "u" (lower-left corner in Figure 3.5) from $-\sqrt{2}V_{DD}$ to $+\sqrt{2}V_{DD}$, latch inverters voltage transfer curves (V_1 and V_2) in the rotated $u - v$ coordinates system are obtained. By taking the difference of these curves and evaluating its global maxima and minima, SNMs can be easily measured.

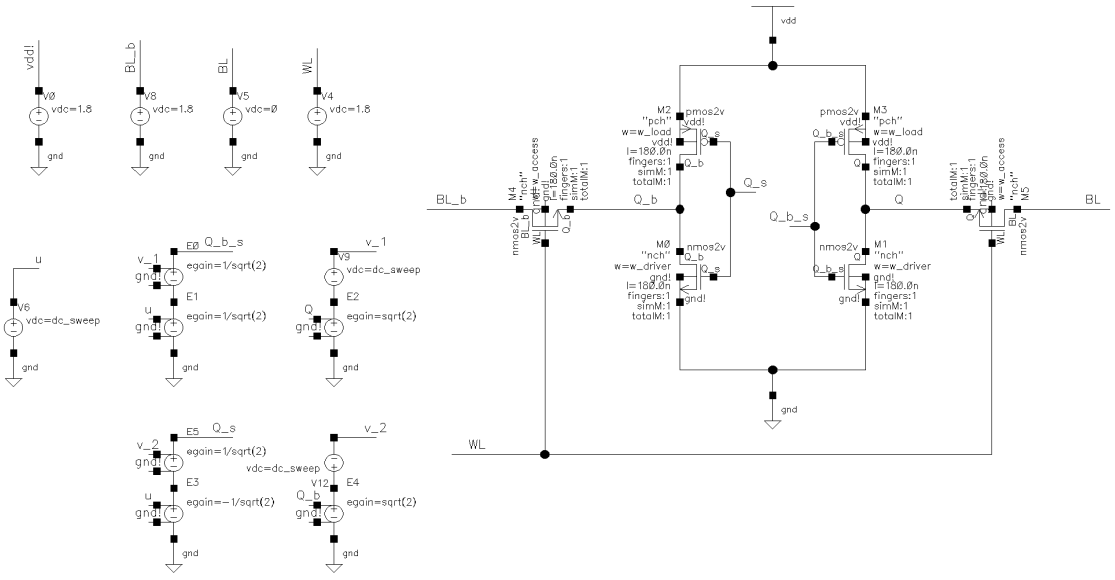


Figure 3.5: Test-bench schematic for evaluating cell WSNM

To measure read/write delays and energy consumption, a transient simulation is needed. Figure 3.6 shows the test-bench schematic for four consecutive read/write cycles (write '1', read, write '0' and read). In the upper-left part of the schematic, two ideal switches, controlled by the Read signal, are used. When Read='0', the switches are closed, and bit-line voltage generators write the cell by discharging one of the bit-lines; after a reasonable amount of time, i.e. after the cell is written, the discharged bit-line is charged to V_{DD}, emulating precharge circuit action. When Read='1', the switches are open, and the bit-lines are floating: one bit-line is discharged depending on the previously written datum. During both read and write cycles, word-line is accordingly asserted. Read/write delays are measured considering 50% voltage variations of word-line, bit-lines (for read) and cell internal nodes $Q/\overline{Q}$ (for write). A 0 V test dc voltage generator is inserted between the cell and ground to measure the cell's absorbed current directly. From the current measurement and knowing initial and final time instants of read/write cycles, the absorbed energy is evaluated as:

$$E = \int_{t_i}^{t_f} V_{DD} I(t)\, dt \tag{3.1}$$

The focus is on energy consumption rather than power because for battery-powered systems is important to limit energy drawn to extend battery life. Peak power is considered for heat removal issues or its impact on power delivery networks [30].

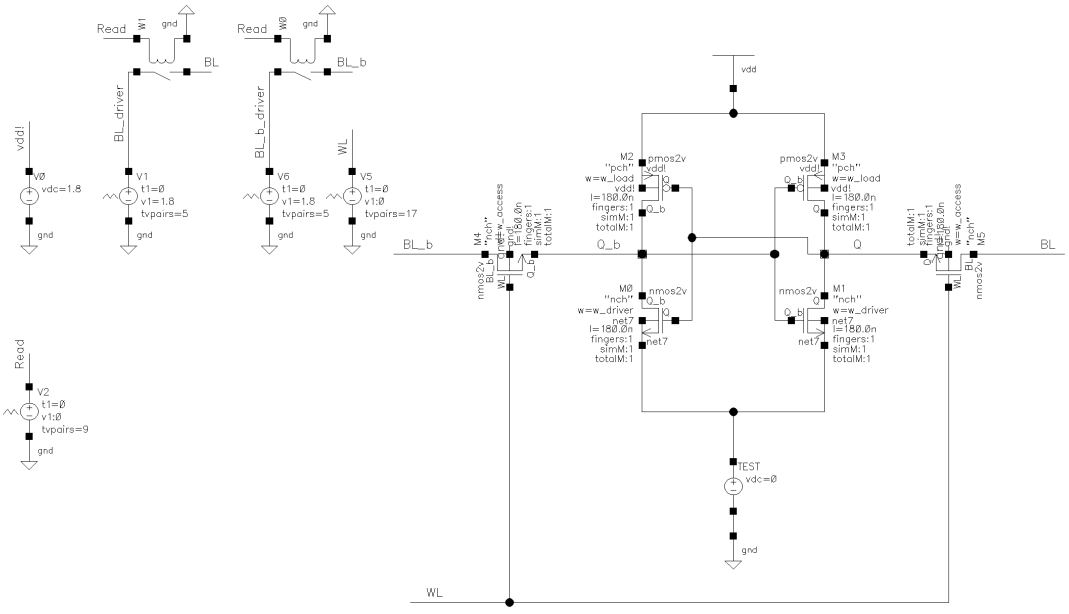


Figure 3.6: Test-bench schematic for evaluating cell delays and consumption

Figure 3.7 shows an example of transient simulation waveforms. The analysis focuses on sweeping *CR* and *PR* to experience how those parameters influence stability, performance and energy consumption. For example, in Figure 3.7, with a constant *CR*, two very different *PRs* are chosen (i.e. 0.5 and 4) to highlight their influence on write-ability: when *PR* is high, the cell becomes unwritable. As already discussed, cell read stability and write-ability are conflicting design requirements and this is proved by simulations in Section 3.2.1.

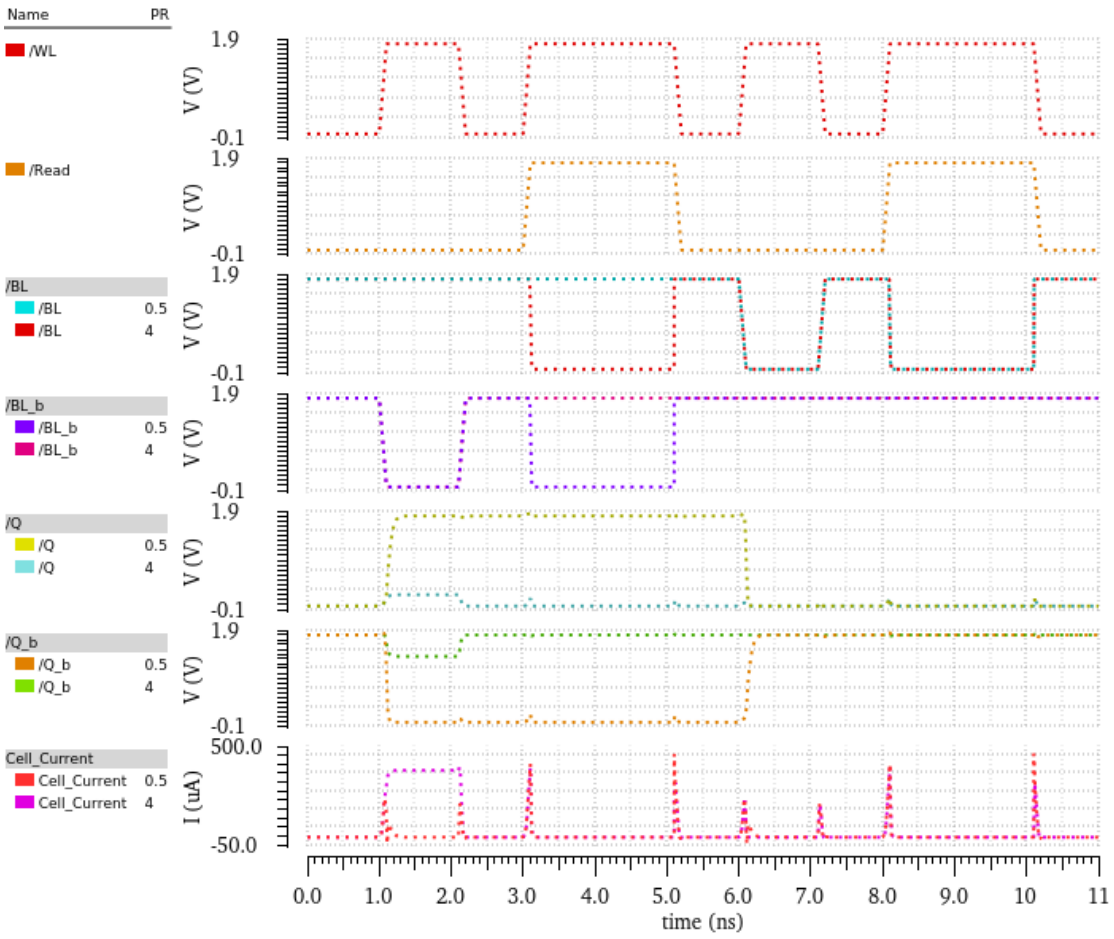


Figure 3.7: Example of transient simulation waveforms of circuit in Figure 3.6 for constant *CR* and two values of *PR* (notice how for *PR*=4 the write-ability is compromised)

3.2.1 Stability, Read/Write Delay and Energy Constraints

Since modern-day technological issues, already discussed before, stability is a significant concern with deep-submicron technologies. Hence SNM is the most critical parameter to find an optimal design. Other parameters, such as cell area, read delay/energy and write delay/energy, are considered in second place. Many different *CRs* and *PRs* have been explored, with minimum length (180 nm) for all the cell transistors. For clarity and conciseness, only simulations results for a chosen constant *CR*/variable *PR* and a constant *PR*/variable *CR* are reported.

Constant *CR* and swept *PR*

For this analysis, $W_{access} = 0.84\,\mu m$ and $W_{driver} = 1.68\ \mu m$ (i.e. $CR = 2$) have been chosen and *PR* has been swept from 0.5 to 4. Since *PR* significantly affects write-ability, WSNM is firstly analyzed (Figure 3.8). As can be noted, as *PR* increases, WSNM dramatically decreases. When *PR* reaches 3.35, the two inverters transfer curves touch in other points (evidenced in Figure 3.8 with a thick red line); this makes the cell unwritable since the circuit acts as a bistable and the WSNM is zero. The cell exhibits a maximum WSNM for approximately $PR = 0.9$.

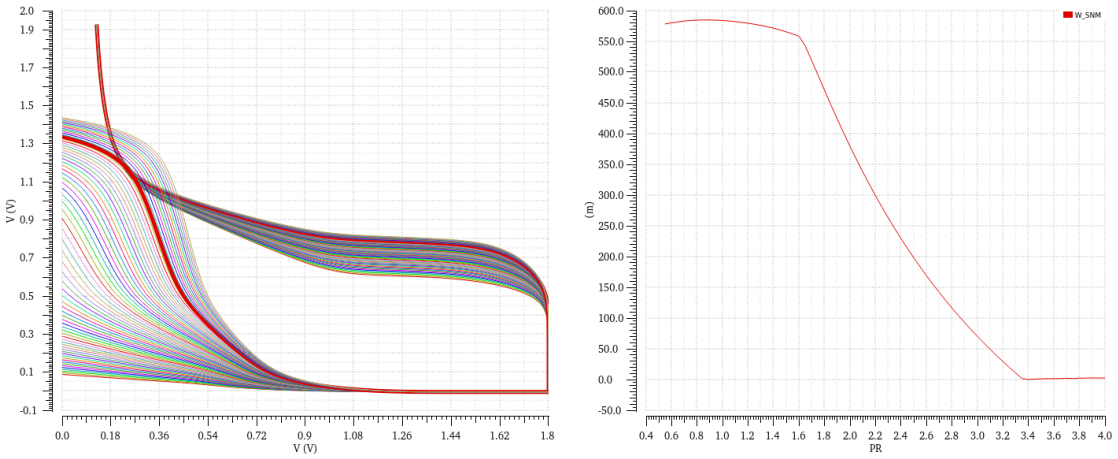


Figure 3.8: WSNM as a function of *PR* for constant *CR* (right) and corresponding butterfly curves (left)

For what concerns HSNM and RSNM, they are shown in Figure 3.9. As expected, RSNM is always lower than HSNM, since the internal nodes degraded zero voltage levels during a read operation. On the contrary, as *PR* increases, both HSNM and RSNM increases. This is explained by the fact that increasing *PR* means to have pull-up transistors beta factors matched with pull-down transistors' ones; this makes the inverters transfer curves centred around $V_{DD}/2$ and increases SNMs. Therefore read stability and write-ability have opposite trends with increasing *PR*, as theoretically described in Section 2.2.3.

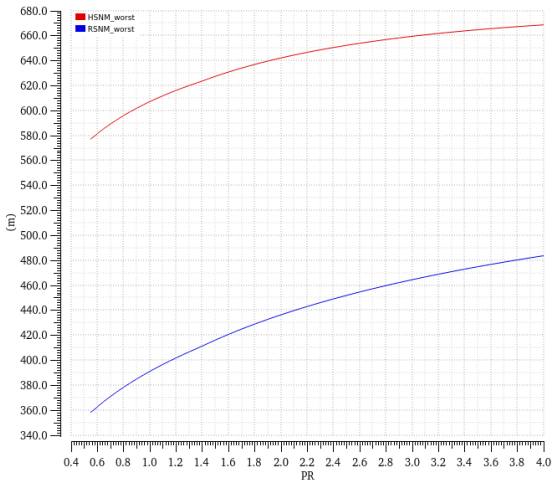
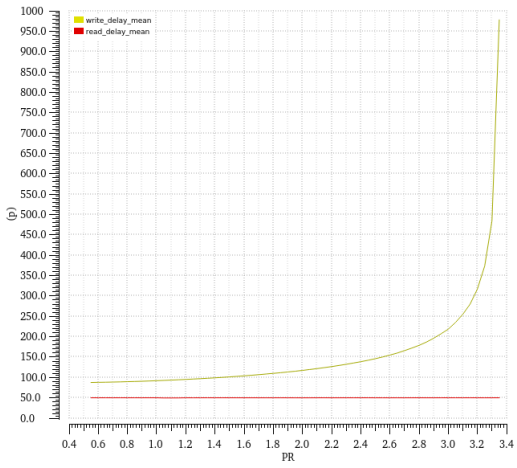
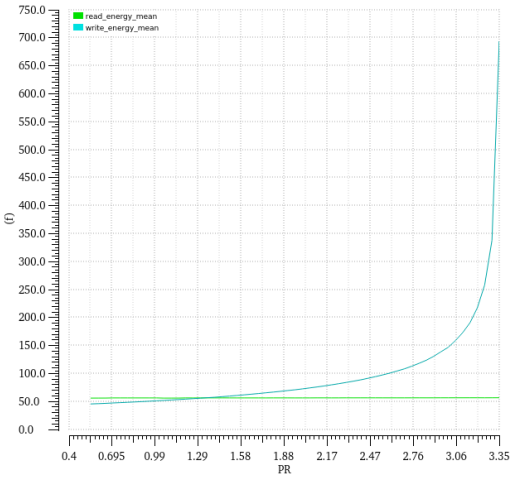


Figure 3.9: HSNM (red line) and RSNM (blue line) as a function of *PR* for constant *CR*

Figure 3.10 shows read/write delays and absorbed energies as functions of *PR*. As can be noticed, both read delay and energy are not affected by *PR* variations; this is because, during a read operation, pull-up transistors do not play any role, and only driver transistors will discharge one bit-line. On the other hand, write delay and energy are significantly affected by pull-up transistors' width.



(a) Read delay (red line) and write delay (yellow line) as a function of PR



(b) Read energy (green line) and write energy (cyan line) as a function of PR

Figure 3.10: Read/write delay and energy as functions of *PR* for constant *CR*

As PR increases, write delay and energy increase exponentially up to the point where the cell becomes unwritable (i.e. for $PR = 3.35$, where the curves show a vertical asymptote). Write delay increases because, as the pull-up transistor becomes stronger than the access-transistor, the datum node is pulled down from V_{DD} by a lower voltage difference, so increasing switching time. Due to this increase of write delay, switching currents must be maintained for a much longer time. Moreover, since inverters input fall-time increases, also short-circuit power increases [31][30]. These two factors contribute to a strong increase in write energies.

Having made all the aforementioned considerations, it is convenient to have PR as low as possible to limit WSNM degradation, area and write delay/energy without degrading HSNM and especially RSNM too much.

Constant PR and swept CR

For this analysis, $W_{access} = 0.84\,\mu m$ and $W_{pull-up} = 0.42\,\mu m$ (i.e. $PR = 0.5$) have been chosen and CR has been swept from approximately 0.2 to 4. Since CR significantly affects read stability, RSNM is firstly analyzed (Figure 3.11). As can be noted, as CR decreases, RSNM dramatically decreases. When CR reaches approximately 0.95, the two inverters transfer curves touch in other points (evidenced in Figure 3.11 with a thick red line); this makes the cell unstable since the butterfly curve shows more metastable points. Any measured RSNM for $CR < 0.95$ has not to be taken into account since the simulator is measuring squares inside small lobes between unstable points, so read stability has to be considered compromised.

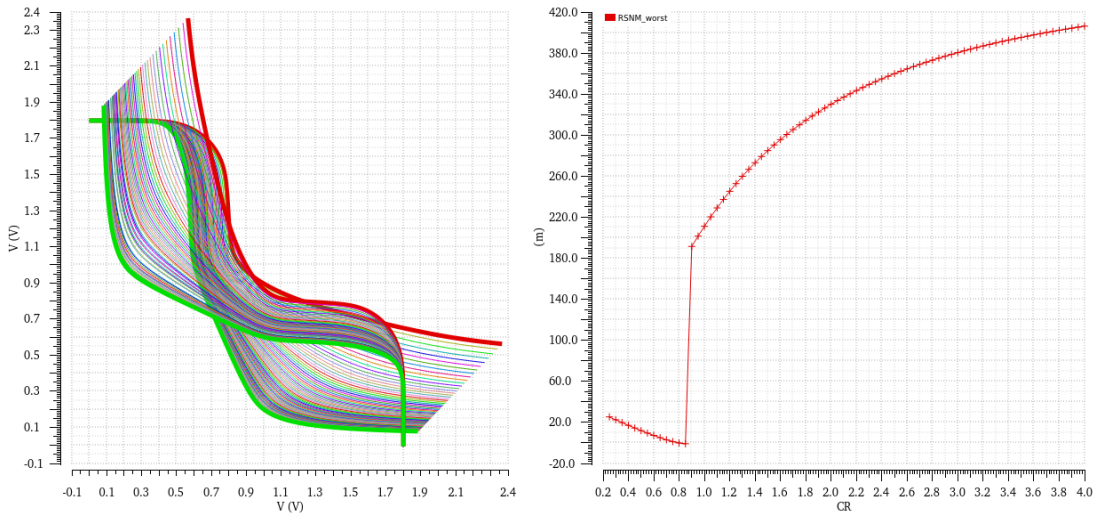
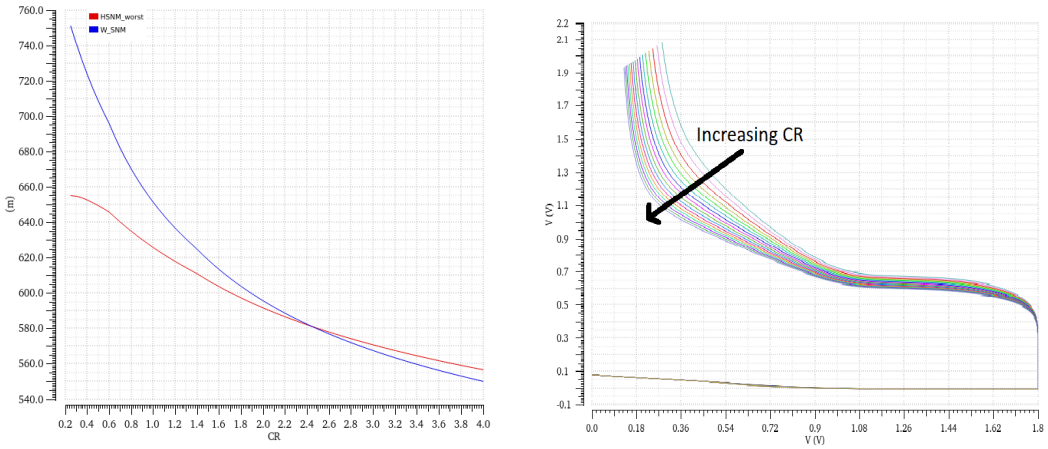


Figure 3.11: RSNM as a function of CR for constant PR (right) and corresponding butterfly curves (left); red thick curve represents an unstable cell while the green thick curve ($CR = 4$) the most stable during read

For what concerns HSNM and WSNM, they are shown in Figure 3.12a. On the contrary, as *CR* increases, both HSNM and WSNM decrease. A first straightforward explanation is that increasing *CR* means to have pull-up transistors much weaker than pull-down transistors; this makes the inverters transfer curves less centred around $V_{DD}/2$ and decreases SNMs. For WSNM there is another factor contributing its decrease: an increasing CR makes the zero level voltage degradation less effective; while this is beneficial for RSNM, during a write operation the butterfly curve tends to be more "closed", thus further reducing WSNM (Figure 3.12b).



(a) HSNM (red line) and WSNM (blue line) **(b)** Butterfly diagrams during write

Figure 3.12: HSNM and WSNM as a function of *CR* for constant *PR*

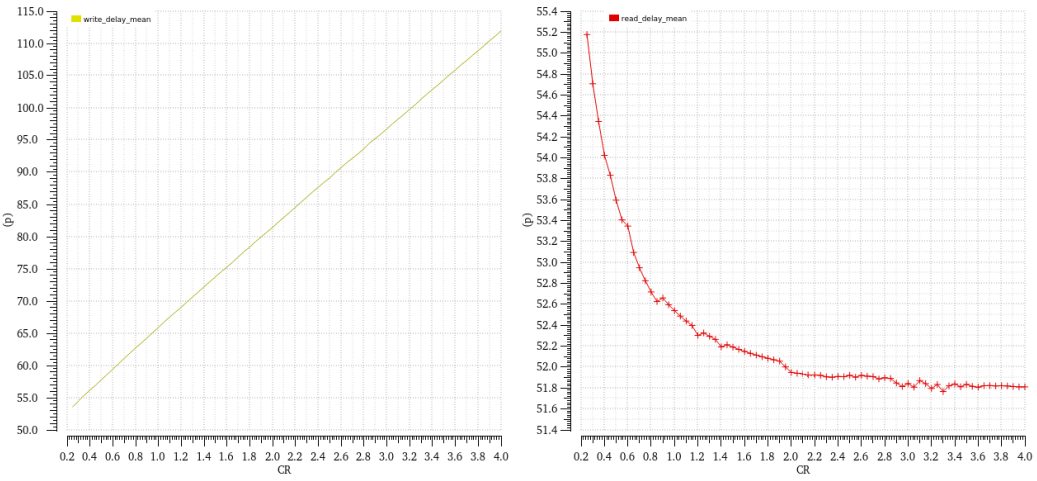


Figure 3.13: Read delay (red line) and write delay (yellow line) as functions of *CR* for constant *PR*

Figure 3.13 shows read/write delays as functions of *CR*. As expected, read delays shows a hyperbolic decrease with increasing CR. In fact, the bit-line discharge time constant, assuming a constant read current and a bit-line voltage drop ΔV from its precharged value, can be approximated as [1]:

$$\tau_{BL} \approx C_{BL}\frac{\Delta V}{I_{read}} \approx C_{BL}\frac{\Delta V}{\frac{\mu_n C_{ox}}{2}\frac{W_{driver}}{L_{driver}}\left(V_{DD}-V_{th,n}\right)^2} \tag{3.2}$$

Hence an increased *CR* (i.e. increased W_{driver}) makes the discharge time constant smaller. On the contrary, write delay shows a positive linear relation with *CR*. Recalling that at the beginning of a write operation, the driver transistor is off, its drain voltage is charged to V_{DD}, and it must be discharged through access transistor before the switching can occur (see Section 2.2.2), the driver transistor act as a capacitive load for the bit-line driver. Hence any increase in its width will produce a linear increase of drain capacitance, which in turn makes propagation delay to rise linearly ($t_p = ln(2)RC$).

Figure 3.14 shows the increasing trend of read/write energies with increasing *CR*. Read energy increases mainly due to an increase of read current (sunk by driver transistor). Write energy increment is due to dynamic power increase during switching because of the increased internal node capacitance and switching current.

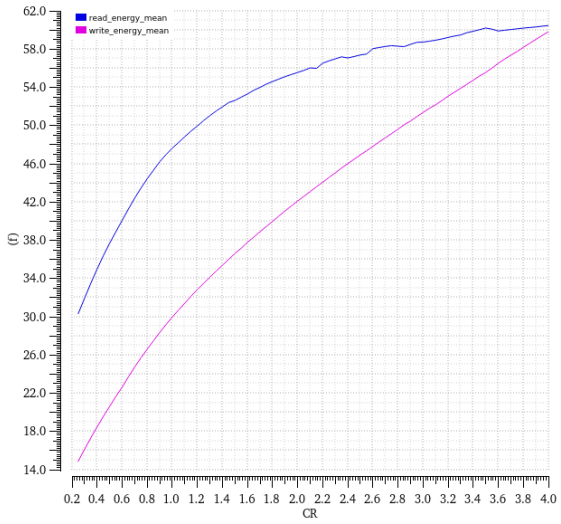


Figure 3.14: Read energy (blue line) and write energy (purple line) as functions of CR

From the simulations above, *CR* should be maximized mainly due to RSNM concerns, without degrading HSNM/WSNM and increasing write delay/energy and read energy too much.

After many simulations in which W_{driver}, W_{access} and $W_{pull-up}$ have been swept, an optimal solution has been found. It is the product of a design trade-off among read stability, write-ability, cell area and energy consumption. The designed cell transistors' widths are summarized in Table 3.1, while its measured key parameters in Table 3.2. Transistors' length has been kept to its minimum (180 nm) to ensure best performances, while $CR \approx 1.45$ and $PR \approx 0.95$ guaranteed a good compromise of the above simulated parameters. One thing to keep in mind is that these simulations have been carried out for a single cell, where during a read operation, bit-lines are completely discharged; this usually does not happen with multiple rows, where the large bit-line capacitance prevent this from happening. Instead, sense amplifiers are used to sense small differential voltage drop among bit-lines: this, of course, influences single bit-cell read delay and energy, which are overestimated in this analysis. Nevertheless, their trends and relationships with CR and PR remain the same. On the contrary, during a write operation, bit-lines are completely discharged; thus, its delay/energy estimation is to be considered accurate.

Transistor parameter	Driver	Access	Pull-up
Length	180 nm		
Width	0.64 µm	0.44 µm	0.42 µm

Table 3.1: Transistors parameters of the designed cell

Measured parameter	Hold	Read	Write
SNM	631 mV	294 mV	633 mV
Delay	-	52 ps	75 ps
Peak power	-	359 µW	197 µW
Absorbed energy	-	27 fJ	27 fJ

Table 3.2: Measured parameters of the designed cell

3.2.2 Cell Layout Design

SRAM cell layout can be very challenging since the very first aim is to design the smallest cell possible while still guaranteeing manufacturability. Virtuoso Layout XL is used with TSMC 180 nm PDK. The aim is to produce a customizable cell layout so that the user can produce a cell with a wanted area, stability, performance and power consumption. Cadence Virtuoso PCell allows to construct layout with customizable geometries, specified by user input [29][32]. Figure 3.15 shows a screenshot of the designed PCell, with cell transistors' widths specified by the user. Once the PCell is designed through its GUI, its SKILL code can be dumped; in this way, the design is portable and can be imported by SKILL commands. The thin cell layout, already discussed in Section 2.3.1, really simplified the PCell creation: since all transistors have the same orientation, a change in transistor width produces only a change in cell width, while the height remains constant; other layers and instances, such as metal, polysilicon and vias, are stretched or moved according to the user-specified parameters.

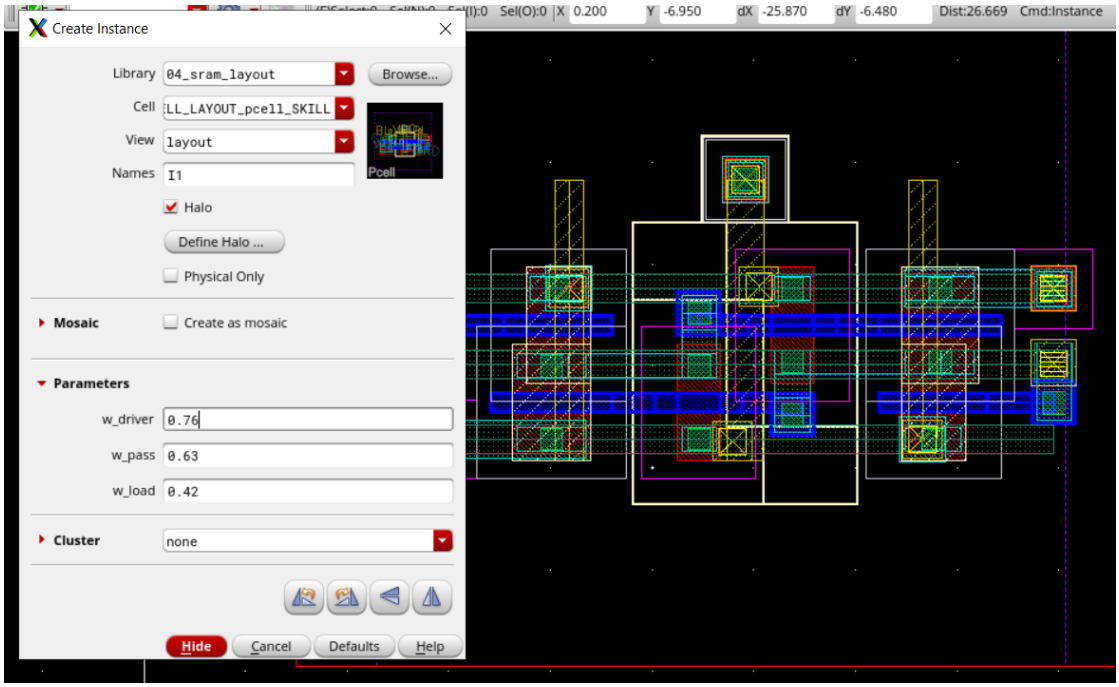


Figure 3.15: Screenshot of SRAM bit PCell instantiation with user-defined transistors' widths in Virtuoso Layout XL

The designed cell of Table 3.1 is shown in Figure 3.16. Metal 1 and polysilicon layers are used to connect pull-up PMOS transistors with NMOS driver ones. Driver transistors share drain contacts with access pass-transistor to minimize cell area. Access transistors' sources are connected through M1-M2 vias to bit-lines, vertically routed by metal 2 layer. Word-line is routed by horizontal metal 3 layer and is connected to access transistors through Poly-M1 via and M1-M3 stacked vias. V_{DD} line is routed vertically by metal 2 and is connected to pull-up transistors trough M1-M2 vias, and to n-well through p-tap on top of the cell. Ground line is routed horizontally by metal 3 and is connected to substrate by n-taps placed on cell sides. N-taps are connected to pull-down transistors horizontally by metal 1. Cell parameters can be summarized as:

- Width: 5.82 µm
- Height: 1.48 µm
- Area: 8.61 μm^2
- Bit density: 116.14 $kbit/cm^2$

Optimal read stability, write-ability and performances counterbalance area penalty, noticing that high-density cell in 180 nm technology has an area of 7.38 μm^2 [33]. The user can nevertheless lower transistors' widths to have a more compact and less power-hungry cell, at the expense of stability and performance. The minimum width available by this technology is 220 nm. During the layout phase, however, it has been noticed that transistors change aspect ratios up to 420 nm to maintain diffusion area constant, constrained by design rules. Hence to have a customizable cell with a constant height, the minimum transistor width is set to 420 nm.

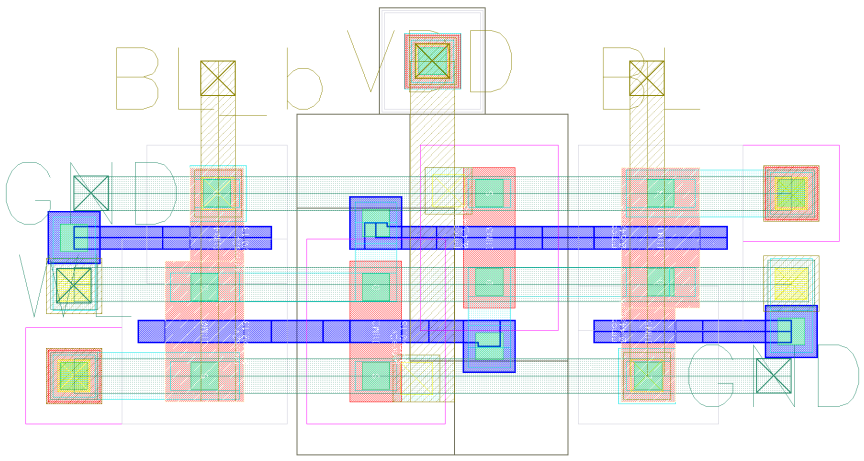


Figure 3.16: 6T-SRAM cell layout with transistors' widths defined in Table 3.1

Bit-cell width and height is measured from vias and transistor contacts centre lines because, when an array is built, each cell can share vias, and transistors are abutted; this drastically increases memory density. An example of an array made of 4 rows and 2 columns is shown in Figure 3.17: adjacent cells are vertically and horizontally flipped to let transistors share their source contacts; moreover, p-taps can be repeated every 4 rows, so that pull-up transistors nwell is also shared vertically among the column cells.

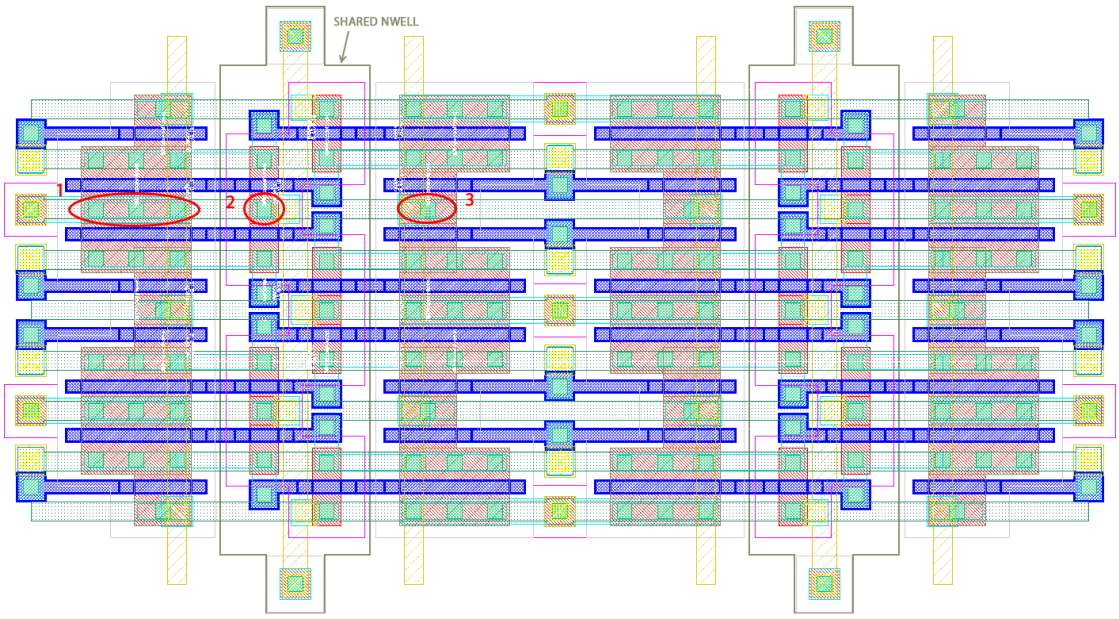


Figure 3.17: 4x2 SRAM array layout; notice that access transistors (red ellipse 1), load transistors (red ellipse 2) and, access transistors (red ellipse 3) share source contacts, and load transistors shares vertical nwell.

The description proceeds with the design of precharge circuit, sense amplifier and write driver. Peripheral circuits performances greatly rely on array dimensions. A lumped capacitance has been added to the bit-lines to speed up simulations and optimize these circuits for a given array dimension. For 180 nm technology, bit-line capacitance is approximately 0.5 fF/bit [22]. Hence, to model a relatively large array of about 1000 rows, a bit-line capacitance of 500 fF has been considered during simulations for an optimal design.

3.3 Precharge Circuit Design

Figure 3.18 shows the precharge circuit schematic. As already discussed in Section 2.4.1, it consists of two PMOS transistors with their gates tied to ground; this makes it a static circuit, which constantly tries to pull up bit-lines to V_{DD}. This imposes a constraint on how much precharge transistors can be strong (i.e. low resistive): the stronger they are, bit-lines will be discharged to a higher minimum voltage level and they will be charged again rapidly to V_{DD}. In the opposite case (i.e. weak precharge transistors), the discharge minimum voltage level will be higher, while they will be charged slowly to V_{DD}. If the minimum bit-line voltage level is too high, sense amplifier could be not able to sense the differential voltage at its input. Instead, if the precharge rise time is too slow, more time is needed before another read/write operation can be started, thus limiting operating frequency. Moreover, during a read operation, cell internal node voltage rise will depend on how strong precharge transistors are; so it is important to check if this level can corrupt information making the cell to switch. Figure 3.19 shows internal node voltage, minimum bit-line voltage and bit-line rise time as functions of precharge transistors width W_{prch} for two bit-line capacitance values. Cell internal node voltage does not depend on bit-line capacitance, since it is a DC voltage. It mainly depends on precharge, access and driver transistor widths. As expected, it increases as W_{prch} increases, but it never reaches $V_{th,n}$ (approximately 0.4 V). So any W_{prch} value does not corrupt cell datum. On the other hand, as W_{prch} increases, minimum bit-line voltage increases and its rise time decreases. $W_{prch} = 1\,\mu m$ has been chosen for a good trade-off between these two. Transistors length has been set to its minimum (180 nm) for high speed.

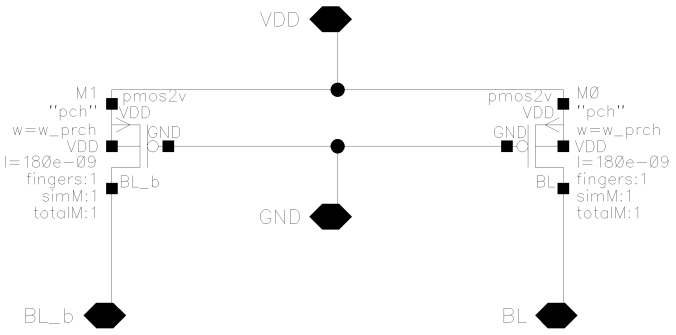


Figure 3.18: Precharge circuit schematic

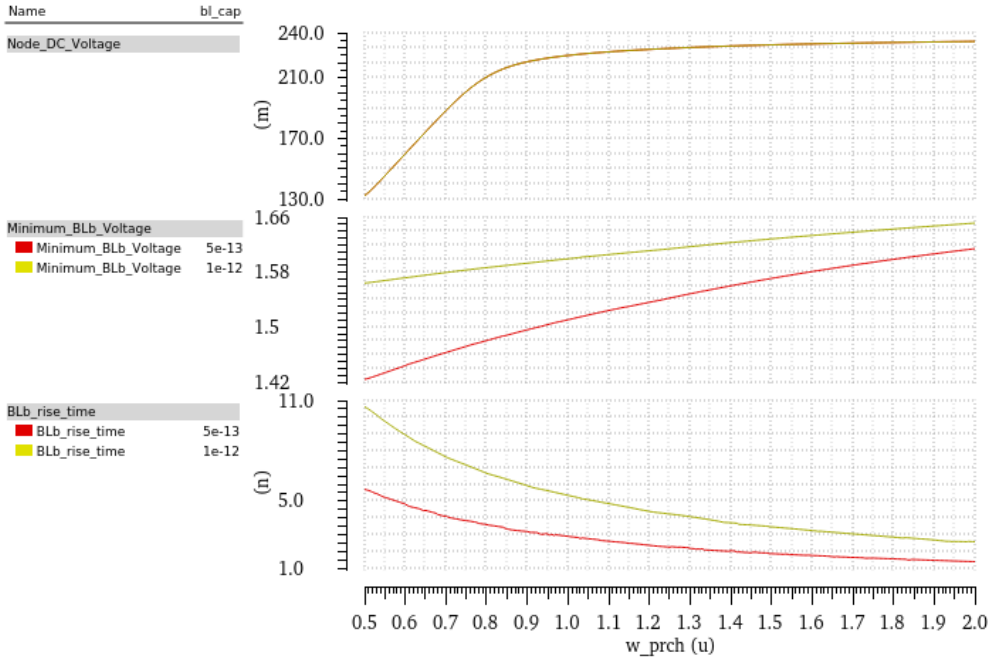


Figure 3.19: Internal node voltage, minimum bit-line voltage and bit-line rise time as functions of W_{prch} for two bit-line capacitance values: 500 fF (red) and 1 pF (yellow)

3.4 Sense Amplifier Design

Sense amplifier is one of the most critical peripheral circuits. Minimum bit-line voltage level greatly influences sense amplifier behaviour. Even if latched sense amplifier can reach very low sensitivities in the order of tens of mV, mismatches can considerably increase this parameter [34]. On the other hand, if bit-lines are discharged to a very low voltage level, sensing would be faster and more reliable but at the expense of an increased read energy due to a higher voltage swing. As discussed before, W_{prch} determines bit-line voltage swing. For a bit-line capacitance of 500 fF and a W_{prch} of 1 µm, bit-line minimum voltage is about 1.4 V, a good trade-off considering sensitivity variation and read energy. Figure 3.20 shows latched sense amplifier schematic. As already discussed in Section 2.4.2, latch transistors' widths and SAE transistor width determines sensing speed. Isolation transistors can be sized with lower widths, since their only task is to precharge latch outputs before a read operation is started, and to isolate them from bit-lines during sensing. An SR ouptut latch is implemented with two cross-coupled NAND gates: they must be sized according to latch output capacitances. The optimal

sizing of all these transistors, for a bit-line capacitance of 500 fF, is reported in Table 3.3. Transistors' length has been set to its minimum (180 nm) for maximum speed.

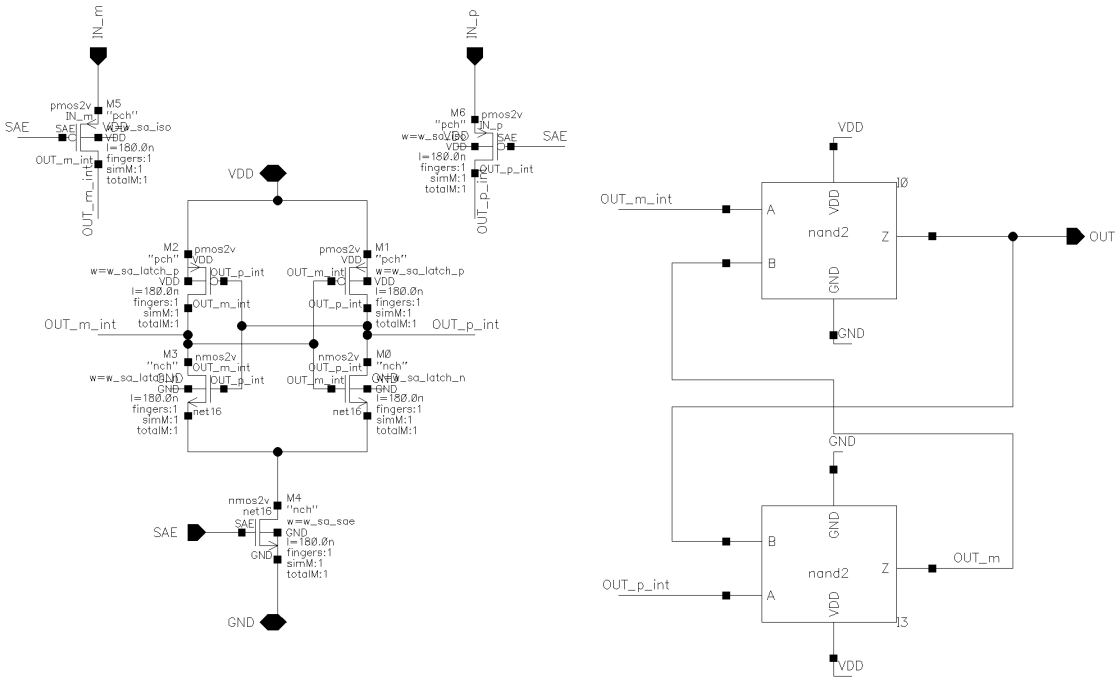


Figure 3.20: Sense amplifier circuit schematic

$W_{latch,n}$	$W_{latch,p}$	W_{iso}	W_{SAE}	$W_{nand,n}$	$W_{nand,p}$
5 µm	10 µm	0.5 µm	10 µm	6 µm	12 µm

Table 3.3: Sense amplifier transistors' widths

An example of transient waveforms is shown in Figure 3.21: when a sufficient differential voltage is produced, SAE pulse is issued and sense amplifier output switches accordingly. The simulation has been run for different bit-line capacitances (up to 2 pF) and the circuit still functions correctly. Figure 3.22 shows sense delays (i.e. 50% delay between SAE signal and sense amplifier output) as functions of bit-line capacitance. As expected, the delay needed to sense a logic '0' is always greater than logic '1': this is because NAND gates have been used for the output latch, which have a slower high-to-low transition due to series of NMOS transistors. The relationship of the delay with bit-line capacitance is not linear as usually happens because isolation transistors isolate sense amplifier input/output from bit-line, but a higher capacitance produces lower voltage swing reducing speed.

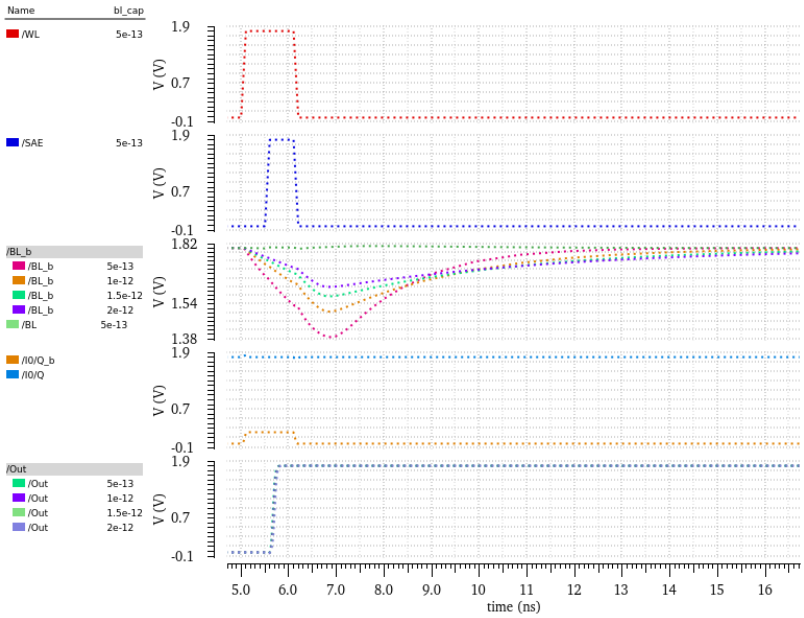


Figure 3.21: Sense amplifier transient simulation example

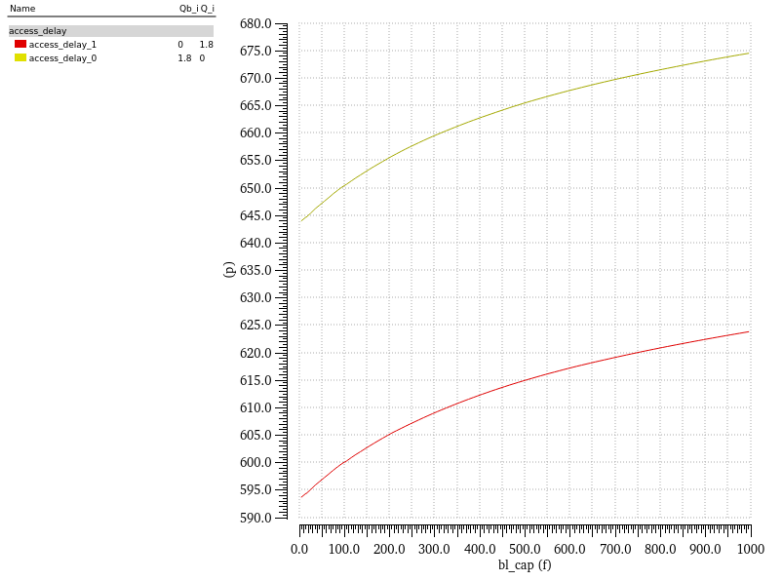


Figure 3.22: Sense delays (red line for logic '1', yellow line for logic '0') as functions of bit-line capacitance

3.5 Write Driver Design

Write driver circuit schematic is shown in Figure 3.23. Write speed depends on the capability of this circuit to pull-down bit-lines rapidly. High and low pass transistors must be stronger than precharge transistors, which constantly tries to pull up bit-lines to V_{DD}. Lower pass transistors could be sized with a higher width than higher pass transistors for optimal delay. Inverters must be sized accordingly to drive pass transistors gates: inverter 2 should be stronger than inverter 1 for optimal delay distribution. Table 3.4 shows optimal write driver transistors' widths for a bit-line capacitance of 500 fF.

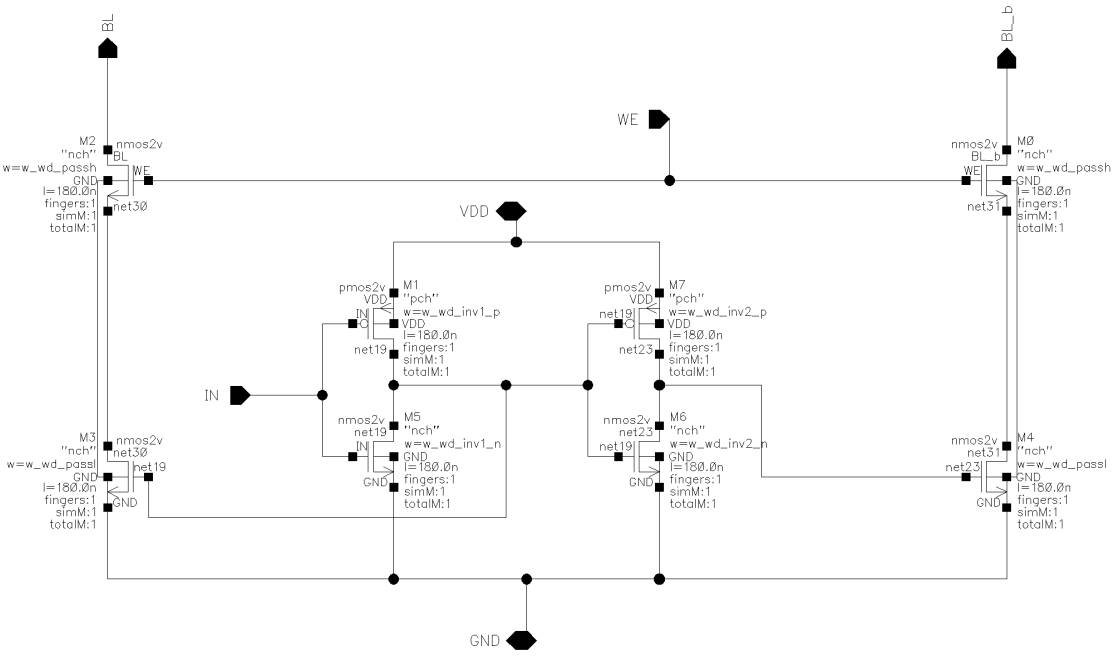


Figure 3.23: Write driver circuit schematic

$W_{pass,l}$	$W_{pass,h}$	$W_{inv_1,n}$	$W_{inv_1,p}$	$W_{inv_2,n}$	$W_{inv_2,p}$
10 µm	6 µm	4 µm	8 µm	6 µm	12 µm

Table 3.4: Write driver transistors' widths

An example of transient simulation is shown in Figure 3.24. Word-line, write-enable and input datum pulses are issued for a sufficient amount of time to guarantee cell writing. Increasing bit-line capacitance causes an increase of bit-line rise time after a write operation, where bit-lines are completely discharged. As already underline, this influences operating frequency. Write delay (i.e. 50% delay between

input datum and rising internal node) as function of bit-line capacitance is shown in Figure 3.25: its linear dependence is expected since during writing bit-lines are directly driven by driver pass transistors. Write '1' delay is always higher than '0' since both the inverters are triggered by input low-to-high transition.

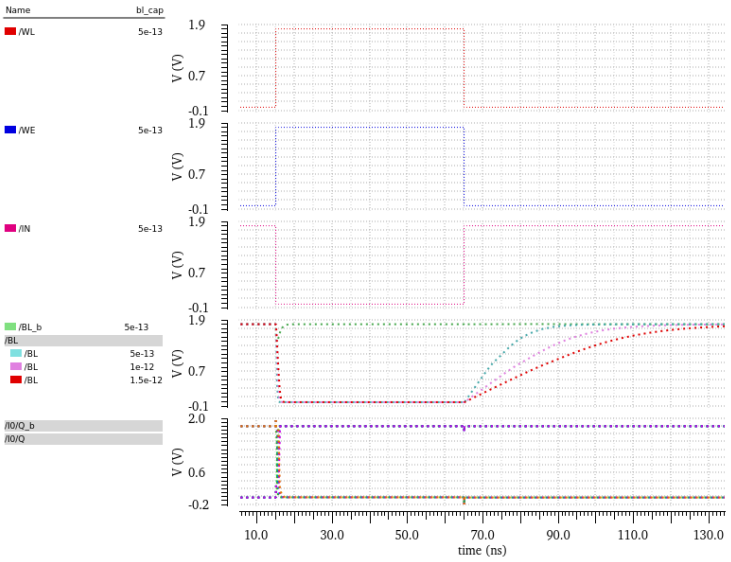


Figure 3.24: Write driver transient simulation example

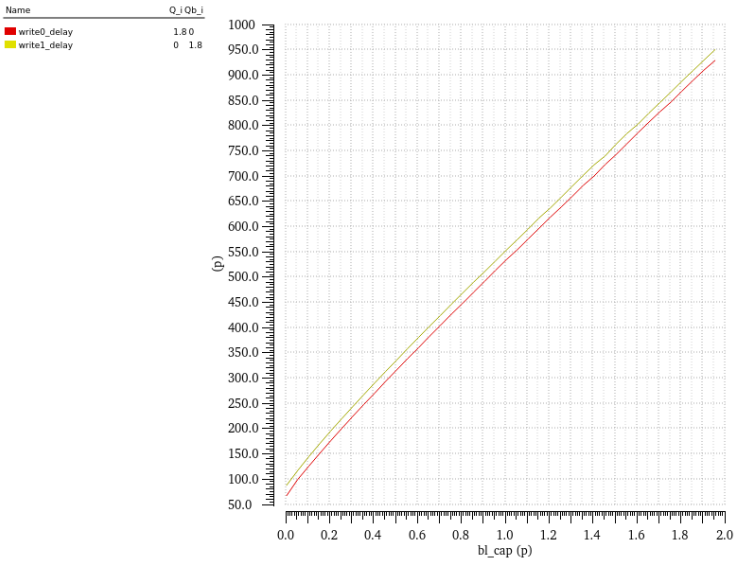


Figure 3.25: Write delays (red line for logic '0', yellow line for logic '1') as functions of bit-line capacitance

Chapter 4

Compiler Design

The exposition now proceeds with the description of the compiler design. Firstly, SKILL scripting language is briefly described in its essential functions and purposes. Then, the compiler structure is depicted, and its primary functions are thoroughly explained. Finally, an example of compiler use is showed and discussed.

4.1 SKILL Scripting Language

The SKILL scripting language was developed by Cadence and is based on the LISP scripting language. SKILL has some features and a syntax borrowed from C language, but it remains a scripting, interpreted, and high-level language. SKILL lets quickly and easily personalize existing CAD software while also assisting in developing new functions. SKILL includes Application Programming Interface (API) functions for having access to each Cadence tool [35]. SKILL scripts are of the ".il" type and are loaded and executed in the Command Interpreter Window (CIW), which is the Graphical User Interface (GUI) of Cadence software. CIW is the actual GUI of the SKILL interpreter. Cadence can also be executed in batch mode, and SKILL scripts can be run from the terminal. SKILL makes use extensively of two basic data types: atoms and lists. An atom is a pure object that can represent a number, a string or a boolean. A list is an ordered collection of data objects: each element can be of any data type, including other lists. Lists are commonly used to represent $x - y$ coordinates of objects, so they are essential in layout automation. Manipulating and accessing lists' elements is the main activity in SKILL scripting. SKILL variables are dynamic, which means that its type (e.g. integer, float or string) is defined by the assigned value and has not to be declared. SKILL control structures are very similar to C ones: "if", "while", and "for" are the most basic with a C-like syntax, while "foreach" lets the user execute an expression for each element of a list.

Procedures can be also declared, specifying its arguments and possible local variables. Table 4.1 shows most peculiar SKILL basic functions and their description.

SKILL function	**Description**
list(x y ...) or *'(x y ...)*	Creates a list with *x, y, ...* elements. *list(x y ...)* function evaluates its arguments, while *'(x y ...)* does not.
car(list(1 2 3)) *=> 1*	Accesses the first element of a list[1].
cdr(list(1 2 3)) *=> (2 3)*	Accesses all the elements of a list except the first[1].
cadr(list(1 2 3)) *=> 2*	Accesses second element of a list[1]; equivalent to *car(cdr(list(1 2 3)))*.
foreach(var_x list_y *expression_1* *expression_2* *...)*	Assigns *var_x* to an element of *list_y* and *expression_1*, *expression_2*, ... are evaluated; this is repeated for each element of *list_y*.
procedure(func_name(*arg_1 arg_2 ...)* *let(x (y val_i) ...)* *expression_1* *expression_2* *...)*	Defines *func_name* procedure with *arg_1*, *arg_2*, ... arguments. The *let* statement defines *x*, *y*, ... local variables, with *y* initialized with *val_i* as initial value. The procedure body is made of *expression_1*, *expression_2*, ... which are evaluated every function call.

Table 4.1: SKILL most peculiar basic functions

Most of the Cadence tools adopt the Design Framework II (DFII) database: they interpret every schematic or layout instance (e.g. cell views, pins, nets, lines, layout rectangles) as software objects, which can be created, accessed and modified by database (db) functions [36]. Each db function returns a database object identifier (or ID): using the access operator "~>", each ID property (e.g. library name, cell name, terminals, layout layer) can be accessed and modified. Table 4.2 shows most used SKILL db functions in layout and schematic generation. The Open Command Environment for Analysis XL (OCEAN XL) allows the user to configure, simulate, and interpret circuit data. OCEAN XL is a text-based scripting language that can be launched from a UNIX shell or the CIW. It is an extension of SKILL for running simulation within Virtuoso ADE XL, using Spectre or other simulators [37]. OCEAN XL most used commands are reported in Table 4.3.

[1]The symbol "=>" is used to indicate the returned value of the function call.

SKILL function	Description
dbOpenCellViewByType()	Opens a cell view of different view types (e.g. schematic, symbol or layout) in read, write or append mode.
dbCreateNet()	Creates a net, needed for a pin.
dbCreatePin()	Creates a pin corresponding to a specified net.
dbCreateTerm()	Creates a terminal for pins, specifying whether it is input, output or input/output.
dbCreateInst()	Creates an instance in the cell view; useful for instantiating lower hierarchy cell views.
dbCreateRect()	Draws a rectangle in the wanted layout layer.
dbFlattenInst()	Flattens instance down to wanted hierarchy levels; useful after Pcell instantiation for preserving nets' names (used later during LVS).
dbSave()	Saves a cell view.
schCheck()	Checks a schematic cell view; needed to perform simulations with ADE XL.

Table 4.2: SKILL most used db functions

OCEAN XL function	Description
ocnxlTargetCellView()	Specifies test-bench cell view name for simulation.
ocnxlBeginTest()	Specify ADE XL test name.
analysis()	Specifies analysis type (e.g. dc or transient).
envOption('analysisOrder list("dc" "tran" ...) 'switchViewList list("spectre" ... "calibre" ...)	Specifies analysis order and type of views to be used for simulation ("calibre" view is the schematic view with extracted parasitics from layout).
converge()	Specifies nets dc or initial condition voltages; useful for initializing memory content.
ocnxOutputSignal()	Chooses which signal to be saved or plotted.
ocnxlOutputExpr()	Sets output expression, used to make calculations (e.g. delay or energy).
ocnxlSweepVar()	Sets global variable that can be fixed or swept.
temp()	Sets temperature.
ocnxlRun()	Runs simulation(s).

Table 4.3: OCEAN XL most used functions

4.2 Compiler Overview

Before describing compiler structure in details, it is helpful to have an overview of how to use it. Before using the compiler, the user must:

1. Create a working directory in which TSMC 180 *nm* technology must be loaded, and Cadence Virtuoso must be opened;
2. Create a library with "tsmc18" technology attached;
3. Unzip "COMPILER" directory in the working directory;
4. Check *DRC_runset.bak* file: in its first line, "*drcRulesFile:" must be followed by the correct DRC rule file path;
5. Check *PEX_runset.bak* file: in its first line, "*pexRulesFile:" must be followed by the correct PEX rule file path;
6. Check *calibreview.setup.bak* file: in its fifth line, "cellmap_file :" must be followed by the correct calview.cellmap file path;
7. Check *calibre_drc_pex_script.sh.bak* file: the variable "layer_map_path" must be assigned to the correct "tsmc18.layermap" file path.

After all the above procedures, the SRAM compiler main script must be loaded in the Virtuoso CIW by typing the following command:

```
load("./COMPILER/sram_compiler.il")
```

Next, the SRAM compiler main function can be called by following this syntax:

```
sram_compiler_proc(LIB_NAME CELL_NAME TB_NAME ROWS COLS
W_ACCESS W_DRIVER W_LOAD CALIBREVIEW_YES)
```

LIB_NAME must be a string containing the working library name, to which "tsmc18" technology has been attached. *CELL_NAME* and *TB_NAME* must be strings containing memory bit cell view name and test-bench cell view name, respectively. *ROWS* and *COLS* must be two integers representing the number of rows and columns of the array, respectively. *W_ACCESS*, *W_DRIVER*, and *W_LOAD* must be floating-point numbers, representing access, driver, and load transistors' widths, respectively. Finally, if *CALIBRE_VIEW_YES* is a string containing the word *"YES"*, the simulator will use the more accurate extracted calibre view; otherwise, the schematic view will be used, speeding up simulations at the expense of accuracy. An example of compiler use is the following:

```
sram_compiler_proc("SRAM_LIB" "6T_CELL" "TB_ARRAY_1024_256"
1024 256 0.44 0.64 0.42 "YES")
```

Table 4.4 summarizes compiler input arguments with their data type and constraints.

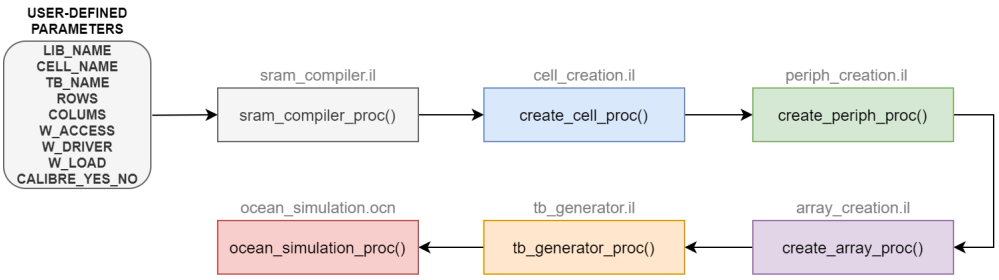
Input argument	Type	Description
LIB_NAME	string	Library name to which "tsmc18" technology must be attached.
CELL_NAME	string	Memory bit cell view name.
TB_NAME	string	Test-bench cell view name.
ROWS	integer	Number of array rows (must be greater than zero).
COLS	integer	Number of array columns (must be greater than zero).
W_ACCESS	float	Access NMOS transistor width expressed in µm; it must be greater or equal than 0.42 µm with increments of 0.01 µm.
W_DRIVER	float	Driver NMOS transistor width expressed in µm; it must be greater or equal than 0.42 µm with increments of 0.01 µm.
W_LOAD	float	Load PMOS transistor width expressed in µm; it must be greater or equal than 0.42 µm with increments of 0.01 µm.
CALIBREVIEW_YES	string	If *"YES"*, it uses the extracted view for simulations; otherwise, it uses the schematic view.

Table 4.4: SRAM compiler input arguments

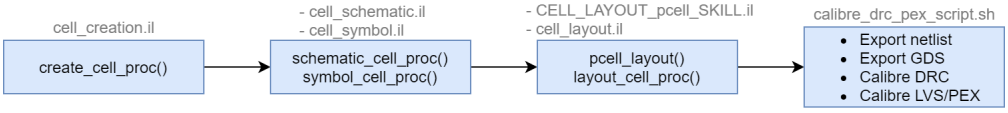
4.3 Compiler Structure

Once the mandatory procedures for using the compiler have been defined, its structure is thoroughly described in this section. The compiler is designed in a modular way: when the user calls the main procedure, five different sub-procedures are executed consecutively. Figure 4.1 shows compiler's main procedure and sub-procedures flowcharts: the rectangles are the used procedures, while the greyed writings near them are the corresponding ".il" files where the procedures are defined. The compilation steps can be summarized in:

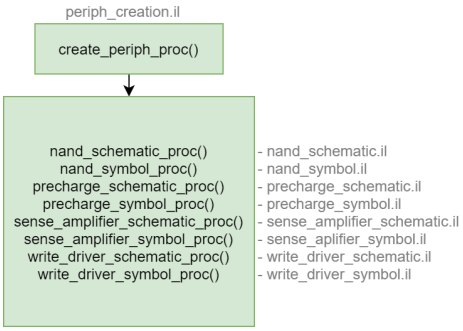
1. Cell creation;
2. Peripherals creation;
3. Array creation;
4. Test-bench generation;
5. Test-bench simulation.



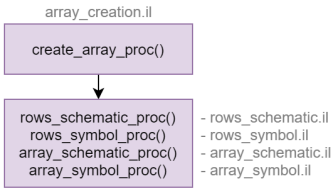
(a) Compiler main flowchart



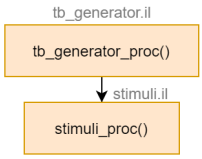
(b) Cell creation flowchart



(c) Peripherals creation flowchart



(d) Array creation flowchart



(e) Test-bench generation flowchart

Figure 4.1: SRAM compiler and sub-procedures flowcharts

Cell Creation

Figure 4.1b shows the flowchart of the procedure *create_cell_proc()*. The first step is the schematic and symbol cell views creation. The 6T-SRAM cell schematic view is created with the user-defined transistors' widths. Afterwards, the designed memory bit PCell (discussed in Section 3.2.2) is loaded in the working library, and it is used by the procedure *layout_cell_proc()*, which creates the layout cell view and instantiates the PCell with the user-defined transistors' widths. The instantiated cell is then flattened to preserve pins and nets names, needed for successful LVS check. After these steps, schematic and layout are exported to perform DRC, LVS, and PEX correctly; to do this, *calibre_drc_pex_script.sh* bash script is executed. In particular, it firstly adds working directory path, library and cell view names to *si.env* file for netlist generation, and calibre runset and setup files for DRC, LVS, and PEX. Then, it exports netilst in Circuit Description Language (CDL) format, layout in Graphic Design System (GDS) format, and it performs DRC, LVS and PEX running Calibre in batch mode, using the following commands:

```
si -batch -command netlist

mv netlist NETLIST_${COMPILER_CELL_VIEW_NAME}

strmout -library $COMPILER_LIB_NAME -strmfile LAYOUT_${
    COMPILER_CELL_VIEW_NAME}.gds -topCell
    $COMPILER_CELL_VIEW_NAME -view layout -logFile strmOut.log -
    layerMap ${layermap_file_path} -case preserve -convertDot
    node

calibre -gui -drc -runset ./COMPILER/DRC_runset -batch > DRC_${
    COMPILER_CELL_VIEW_NAME}_log.txt

calibre -gui -pex -runset ./COMPILER/LVS_PEX_runset -batch >
    LVS_PEX_${COMPILER_CELL_VIEW_NAME}_log.txt
```

Listing 4.1: Bash commands for exporting netlist and layout, and performing DRC, LVS, and PEX

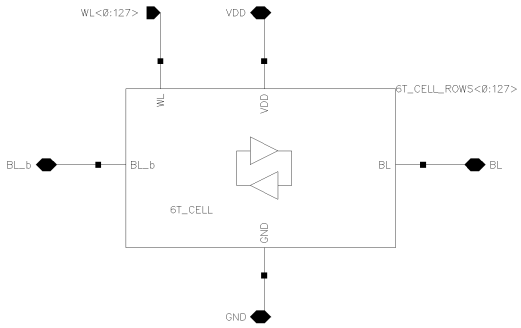
As can be observed, the netlist and layout files are saved in the working directory as "NETLIST_" and "LAYOUT_", respectively, followed by user-defined cell view name. DRC and LVS/PEX logs are also saved in the working directory, while detailed reports are saved in the sub-directories "DRC_RUN" and "PEX_RUN". When all these procedures are completed, the software creates the extracted "calibreview" that can be used later during simulations. Cell creation can be skipped by commenting *create_cell_proc()* in the main script file, if the wanted cell has been already created.

Peripherals creation

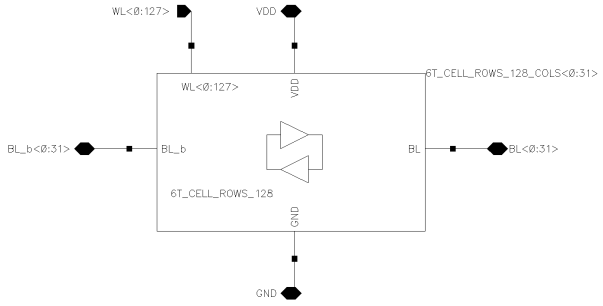
Figure 4.1c shows the flowchart of the procedure *create_periph_proc()*. It executes a series of processes for creating peripherals schematic and symbol cell views, which will be instantiated in the test-bench. All the schematic cell views are defined with variables as transistors' widths: the default dimensions are described in Chapter 3, but their values can be changed for simulations in the script *ocean_simulation.ocn* . If peripherals have been already created, peripheral creation can be skipped by commenting *create_periph_proc()* in the main script file to reduce compilation time.

Array creation

Figure 4.1d shows the flowchart of the procedure *create_array_proc()*. It firstly creates schematic and symbol views for a column of the wanted number of rows by instantiating the memory bit symbol with the vector notation *<0:ROWS-1>* (an example of 128 rows is depicted in Figure 4.2a). Then, it instantiates the single column symbol view for the wanted number of columns with the vector notation *<0:COLS-1>* creating the complete array (an example of 128 rows and 32 columns array is shown in Figure 4.2b), and it creates its symbol view.



(a) Schematic of a single column of 128 rows



(b) Schematic of an array of 128 rows and 32 columns

Figure 4.2: SRAM array creation schematics

Test-bench generation

Figure 4.1e shows the flowchart of the procedure *tb_generator_proc()*. It generates the test-bench by instantiating the array, the peripherals for each column (using the vector notation *<0:COLS-1>*) and it connects them properly. Then, by means of the sub-procedure *stimul_proc()*, it creates the input voltage generators and output capacitances needed for the simulation (an example is reported in Figure 4.3). Word-Line (WL) signal is supplied only to the first row, while all the remaining rows are grounded. Input (IN), Write Enable (WE), and Sense Amplifier Enable (SAE) signals are supplied to each column. Input stimuli are generated by

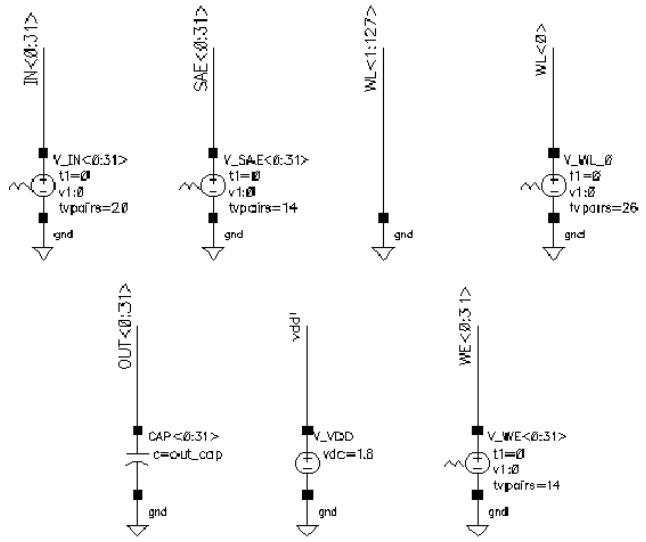


Figure 4.3: Example of test-bench stimuli generators and output capacitances for a 128×32 array

Piece-Wise Linear (PWL) voltage sources: thanks to this approach, completely parametric waveforms are generated. The complete transient simulation consists of Write '1'/Read/Write '0'/Read consecutive cycles. An example of a Write '1'/Read waveform is shown in Figure 4.4. Red arrows represent all the possible delays, pulse widths, and setup/hold times (measured from 50% voltage swing) that the user can customize during simulation; this is crucial for finding optimal timing for the generated memory array. For simplicity, all signals are supplied with the same user-defined rise and fall time. Before the input stimuli are supplied, all the signals are at logic '0' from 0 s to t_{start} time: this can be useful to evaluate memory static behaviour during transient simulations.
During a write cycle, WL, WE and IN are supplied according to the defined setup, hold times, and pulse widths. Setup and hold times can also be negative, indicating a signal's rising or falling edge in advance with respect to another. Actually, for

the write cycle, not all the timing parameters are independent: for this reason, setup and hold times, and WE pulse width are considered independent variables that can be modified by the user, while the other parameters can be derived as:

$$\begin{aligned} t_{WL,write-cycle} &= t_{WL,setup} + t_{WE,pulse\ width} + t_{WL,hold} \\ t_{IN,pulse\ width} &= t_{IN,setup} + t_{IN,hold} \\ t_{IN,delay} &= t_{WE,pulse\ width} - t_{IN,setup} \end{aligned} \tag{4.1}$$

During the read cycle, WL and SAE signals are supplied following the timing in Figure 4.4. All the read timing parameters are independent of each other. The delay of SAE with respect to WL can also be negative, meaning that SAE pulse can be supplied during WL high phase. For example, if $t_{SAE,delay} = -\ t_{SAE,pulse\ width}$, SAE pulse is supplied right before WL falling edge, and both signals falls at the same time.
WL period must be greater than the maximum between read and write cycles:

$$t_{WL,period} > max\{t_{WL,write-cycle}, t_{WL,read-cycle}\} \tag{4.2}$$

Furthermore, WL low phase must be sufficiently long to ensure a correct bit-lines precharge, especially after a write cycle when one of the bit-line is completely discharged. Timing parameters and output capacitance can be changed by the user in the simulation script *ocean_simulation.ocn*; timing parameters can be swept to find maximum operating frequency, minimum WL read/write cycles, pulse widths, and setup/hold times.

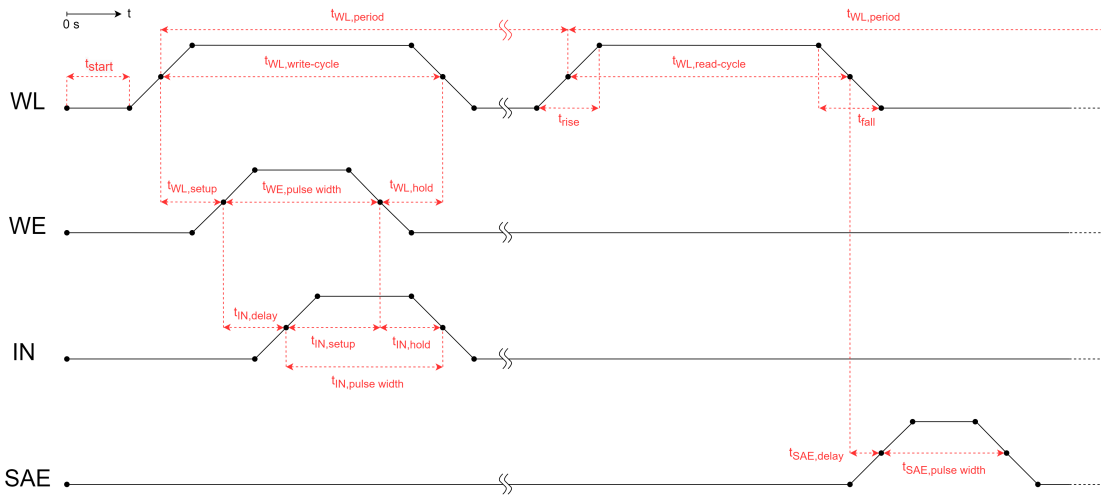


Figure 4.4: Write '1'/Read waveform example; black dots represents PWL points, and red arrows are the delays, pulse widths, and setup/hold times that the user can customize

Test-bench simulation

Simulation is the last step of the compiler flowchart in Figure 4.1. It is carried out by calling the function *ocean_simulation_proc()*. It creates the ADE XL view of the test-bench, it defines type of analysis to be run (i.e. dc and transient) and it set the cell view to be used, either schematic or calibre view. If the calibre view is to be used, Spectre Advanced Parallel Simulator (APS) is employed: it speeds up memory simulation using multi-threading and post-layout optimization, very useful for large arrays. The following part of the code sets calibre view and APS mode:

```
if( CALIBREVIEW_YES_NO == "YES"
    then

    envOption( 'analysisOrder  list("dc" "tran" ...)
               'switchViewList  list("spectre" ... "calibre" ...)

    option( ?categ 'turboOpts
            'psrSwitch  t
            'apsplus  t
            'uniMode  "APS"

    )
)
```

Listing 4.2: OCEAN XL commands for using calibre view and APS mode

Then, memory content is initialized using this part of code:

```
for( i 0 ROW-1
  for( j 0 COL-1
    ... ;(q and qb strings definition)
    if( (i==0)

    then
      converge( 'nodeset q_string "0" )
      converge( 'nodeset qb_string "1.8" )
      converge( 'ic q_string "0" )
      converge( 'ic qb_string "1.8" )
    else
      converge( 'nodeset q_string  "1.8" )
      converge( 'nodeset qb_string "0" )
      converge( 'ic q_string "1.8" )
      converge( 'ic qb_string "0" )
    )
  )
)
```

Listing 4.3: OCEAN XL commands for initializing memory data

As can be noticed, the first row of the array is initialized to a logic '0' because the first datum to be written in the simulation is a logic '1'. All the remaining rows, which are not accessed, are initialized to a logic '1' because this is the worst case scenario: having opposite data stored, access transistors leakage current will tend to keep bit-lines to V_{DD}, while the accessed cells tries to discharge them; this will slow down discharge rate and increase access time [30]. The user can in any case change this part of the code to initialize the memory to the wanted values. Output voltages are initialized to 0 V and bit-lines to V_{DD}.
The user can modify timing parameters, output capacitance, and peripherals transistors' widths by changing the following part of the code:

```
ocnxlSweepVar("IN_hold_time" "0")
ocnxlSweepVar("IN_setup_time" "WE_pulse_width+100p")
ocnxlSweepVar("out_cap" "50f")
ocnxlSweepVar("SAE_pulse_width" "0.1n")
ocnxlSweepVar("SAE_WL_delay" "-SAE_pulse_width")
ocnxlSweepVar("start_time" "1n")
ocnxlSweepVar("stop_time" "start_time+4*WL_period")
ocnxlSweepVar("t_fall" "10p")
ocnxlSweepVar("t_rise" "10p")
ocnxlSweepVar("w_nand_n" "6u")
ocnxlSweepVar("w_nand_p" "12u")
ocnxlSweepVar("w_prch" "1u")
ocnxlSweepVar("w_sa_iso" "0.5u")
ocnxlSweepVar("w_sa_latch_n" "5u")
ocnxlSweepVar("w_sa_latch_p" "10u")
ocnxlSweepVar("w_sa_sae" "10u")
ocnxlSweepVar("w_wd_inv1_n" "4u")
ocnxlSweepVar("w_wd_inv1_p" "8u")
ocnxlSweepVar("w_wd_inv2_n" "6u")
ocnxlSweepVar("w_wd_inv2_p" "12u")
ocnxlSweepVar("w_wd_passh" "10u")
ocnxlSweepVar("w_wd_passl" "6u")
ocnxlSweepVar("WE_pulse_width" "0.3n")
ocnxlSweepVar("WL_hold_time" "0")
ocnxlSweepVar("WL_read_cycle" "0.8n")
ocnxlSweepVar("WL_setup_time" "-10p")
ocnxlSweepVar("WL_period" "1.5n")
```

Listing 4.4: OCEAN XL commands for setting timining parameters, output capacitance and peripherals transistors' widths

Then, static power consumption, read/write delays and maximum currents are calculated by output expressions, using built-in calculator functions. The complete simulation results are stored in the sub-directory "SIM/SIM_RES" and a summary of the calculated parameters is reported in a ".csv" file in the sub-directory "SIM". The results can also be accessed via ADE XL GUI in the history tab.

Chapter 5

Results and Discussion

In this section some array sizes have been compiled and simulated, for which an optimal timing has been found. The designed cell (see Section 3.2.2) has an aspect ratio equal to:

$$A/R_{\ cell} = \frac{Width_{cell}}{Height_{cell}} = \frac{5.82\,\mu m}{1.48\,\mu m} = 3.93 \tag{5.1}$$

If an array has the number of rows equal to four times the number of columns, its aspect ratio is approximately unitary:

$$A/R_{\ array} = \frac{Width_{array}}{Height_{array}} = \frac{\#COLs \cdot Width_{cell}}{\#ROWs \cdot Height_{cell}} = \frac{A/R_{\ cell}}{4} = 0.98 \tag{5.2}$$

Hence, all the compiled and simulated arrays have this aspect ratio to minimize wiring parasitic capacitances and resistances, and to have a squared footprint.

5.1 Compiler Performances

Software compilation and simulation times have been evaluated by using the built-in SKILL function *measureTime()*. The results are shown in Table 5.1. Different dimensions have been compiled, from small arrays (256 bit) up to large ones (512 kbit, not reported in Table 5.1 for conciseness). As expected, compilation time is practically independent of array dimension since the compiler must extract parasitics only from the single-cell layout and not from the whole array. Different CPU workloads and memory availability may cause slight oscillations during compilation. Simulation time strongly depends on array dimension: considering that each cell comprises six transistors, even small arrays can contain up to tens of thousands of transistors. Moreover, the cell schematic extracted from the layout contains about three hundred parasitic elements and a hundred internal nets: this makes the mathematical problem solved by the simulator enormously complicated.

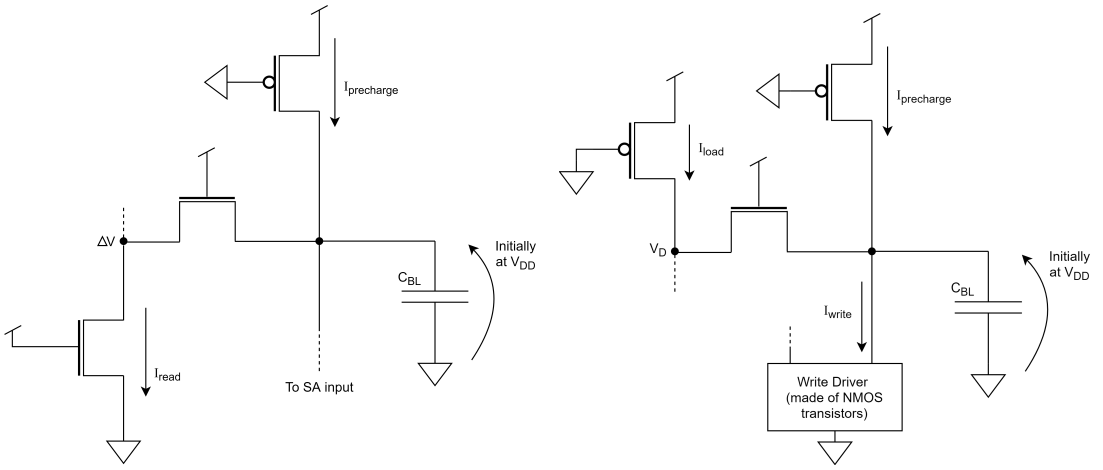
In fact, SRAM memories are usually characterized by employing simple models for a more rapid estimation of timing parameters at the expense of accuracy. Simply simulating the memory critical path instead of the whole array can take up weeks [38]. For the aforementioned reasons and due to limited server capabilities, only small and medium sized arrays (up to 4 kbit) have been simulated.

	Array dimension				
Elapsed time	32×8 (256 bit)	64×16 (1 kbit)	128×32 (4 kbit)	256×64 (16 kbit)	512×128 (64 kbit)
Compilation	33.9 s	34.1 s	33.9 s	34.1 s	34.6
Simulation	42.2 s	3 min 39 s	39 min 19 s	-	-

Table 5.1: Compilation and simulation times for different array dimensions

5.2 TT vs. SF Corners

Each array has been simulated for both Typical-NMOS-Typical-PMOS (TT) and Slow-NMOS-Fast-PMOS (SF) process corners. The SF corner represents a worst-case scenario for this SRAM memory design (where precharge transistors are static PMOS), for both read and write operations from a timing point of view. To understand better why this is true, Figure 5.1 shows an SRAM cell accessed for read (Figure 5.1a) and write (Figure 5.1b) operations.



(a) Read-accessed cell **(b)** Write-accessed cell

Figure 5.1: Schematics of an SRAM cell accessed for read and write operations

For the read operation, suppose the cell content to be a logic '0': the cell NMOS driver transistor is turned on providing the read current, while the PMOS load transistor is turned off (not drawn in Figure 5.1a for simplicity). Bit-line is initially at V_{DD} and is constantly charged by PMOS precharge transistor current. When the cell is accessed (i.e. word-line is asserted high), internal node voltage rises to a certain ΔV (less than $V_{th,n}$ for a non-disruptive read) and the read current flows through the series of driver and access transistors discharging bit-line capacitance C_{BL}. In an SF process corner, the read current sunk by NMOS transistors is lower, causing a lower bit-line discharge rate. Moreover, PMOS precharge current will be higher: this increases achievable minimum bit-lines voltage, increasing sense amplifier sensing delay. These two factors contribute to increasing read access time. For the write operation, suppose the cell content to be a logic '1' (remember that write starts at the node storing a logic '1' due to non-disruptive read requirement for the node storing a logic '0'): the cell PMOS load transistor is turned on keeping the datum node to V_{DD}, while the NMOS driver transistor is turned off (not drawn in Figure 5.1b for simplicity). When the cell is accessed, write driver pulls down bit-line almost to ground and internal node voltage V_D below $V_{th,n}$ (for a reliable writing) by sinking the write current I_{write}. In an SF process corner, the write current is lower since write drivers NMOS pass transistors are slower reducing bit-lines discharge rate. Moreover, PMOS cell load and precharge transistors are faster, providing higher currents that attempt to keep bit-lines to V_{DD}, slowing again their discharge. These factors contribute to increasing write delay.

5.3 Simulation Results

Optimal timing parameters have been found for three different array dimensions (256 bit, 1 kbit, and 4 kbit), and they are reported in Table 5.2. An example of optimal wave-forms is depicted in Figure 5.2, where the complete consecutive simulated Write '1'/Read/Write '0'/Read cycles are showed. Timing parameters have been found simulating the arrays with the worst-case SF process corner, output capacitance of 50 fF and rise/fall times of 10 ps. WL period is increased for the largest array of 4 kbit (from 1.5 ns to 1.7 ns) mainly due to an increased precharge rise time (due to increased bit-line parasitics). WL read cycle is increased for each array size to ensure a correct minimum bit-line voltage level needed to have reliable and fast sensing. SAE minimum pulse width is 60 ps, and SAE delay is set to -60 ps: this means that SAE pulse is issued just before WL falling edge (when bit-lines reach minimum voltage levels), and both signals fall simultaneously. It is important to limit SAE pulse width to reduce power consumption while still guaranteeing reliable sensing. For a write operation, minimum hold times are zero, meaning that WL, WE, and IN signals can fall simultaneously. Minimum WE

Timing Parameter	Array dimension		
	32×8 (256 bit)	64×16 (1 kbit)	128×32 (4 kbit)
$t_{WL,period}$	1.5 ns		1.7 ns
$t_{WL,read-cycle}$	300 ps	500 ps	700 ps
$t_{SAE,delay}$	-60 ps		
$t_{SAE,pulse\ width}$	60 ps		
$t_{WL,write-cycle}$	290 ps		
$t_{WL,setup}$	-10 ps		
$t_{WL,hold}$	0 s		
$t_{WE,pulse\ width}$	300 ps		
$t_{IN,delay}$	-100 ps		
$t_{IN,setup}$	400 ps		
$t_{IN,hold}$	0 s		
$t_{IN,pulse\ width}$	400 ps		

Table 5.2: Optimal timing parameters found for different array dimensions

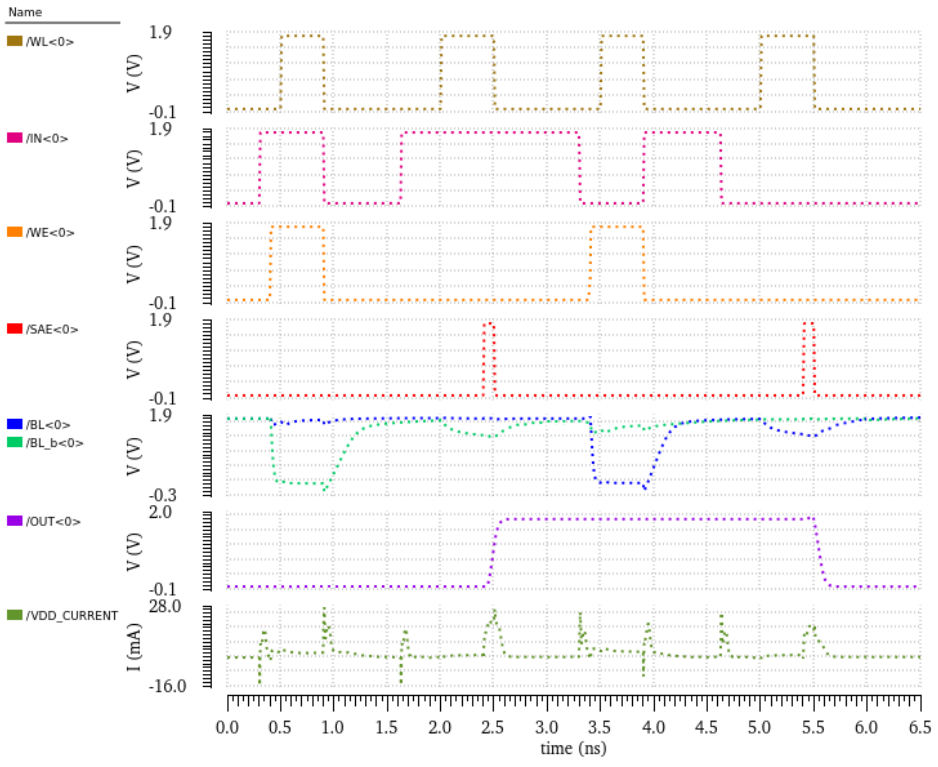


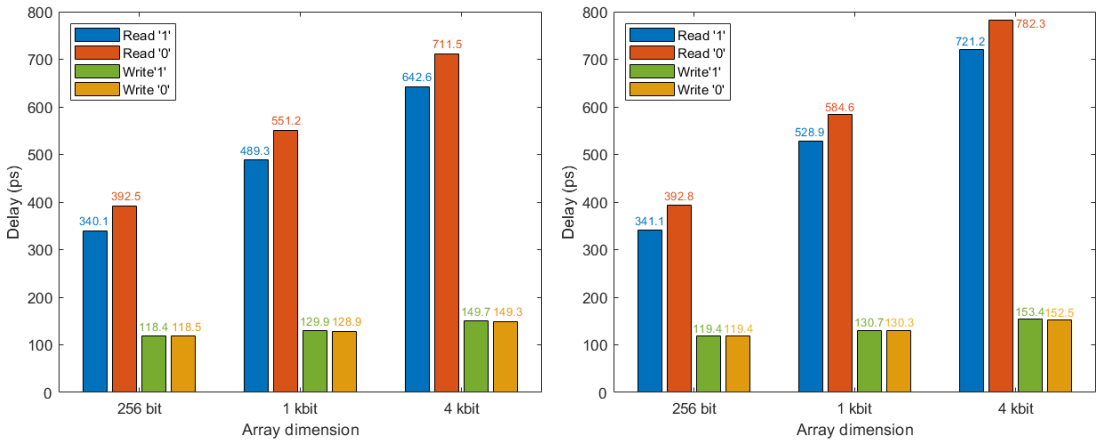
Figure 5.2: Optimal wave-forms example for a 64×16 (1 kbit) array

pulse width is 300 ps, and minimum IN pulse width is 400 ps. IN setup time is set to 400 ps, so that IN delay is -100 ps; this means that IN pulse is issued 100 ps before WE one: this is needed to ensure IN propagation through the two write driver's inverters before WE signal rise. Moreover, WL setup time is -10 ps (which makes WL write cycle equal to 290 ps), so that WL pulse is supplied 10 ps after WE activation: this is needed to wait for WE pass transistors to turn on before activating WL. The fact that IN, WE, and WL activate in succession ensures that only the correct bit-line is discharged: this prevents from wasting energy due to unintentional bit-line discharge or, in the worst case, writing incorrect datum.

The analysis now proceeds with a comparison of the measured read/write delays, peak currents, absorbed energies and static current for all the simulated arrays with both TT and SF process corners.

5.3.1 Read and Write Delays

Figure 5.3 shows read and write delays for the three simulated array sizes. Read delay is measured as 50% delay from WL signal assertion to a valid output: it can be thought as a WL access time. Write delay is measured as 50% delay from IN signal assertion to internal node voltage rise. Focusing on the TT corner (Figure 5.3a), read '0' delay is always larger than read '1' and their difference increases with increasing array dimension; this is caused by two factors: i) sense amplifier output latch has a slower high-to-low transition due to NAND gates, and ii) all the non-accessed rows have been initialized to a logic '1' and their leakage currents tend to slow down bit-line discharge. The last effect is more evident with increasing



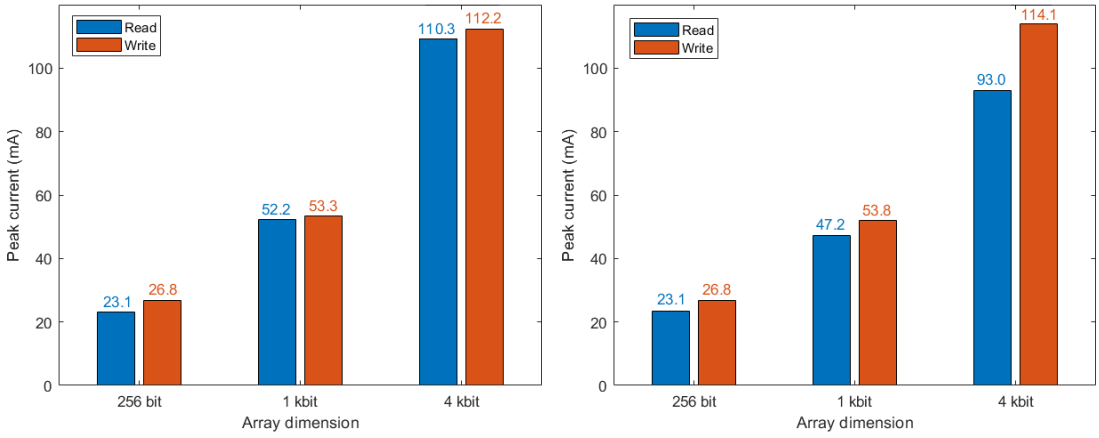
(a) TT corner **(b)** SF corner

Figure 5.3: Read and write delays for different array dimensions

number of rows. Write delays instead are similar for both logic '1' and '0', they are way smaller than read ones and they slightly increase with increasing array size: this is because write drivers transistors are larger than cell ones providing much higher currents. For the SF process corner (Figure 5.3b), as expected, read and write delays are always larger than TT corner's ones: this is particularly true for read, since they are caused by cells pull-down small transistors.

5.3.2 Peak Currents

Figure 5.4 shows read and write supply peak currents for the three simulated array sizes. An arithmetical mean has been carried out for logic '1' and '0' energies since they present similar values. For the TT process corner (Figure 5.4a), read and write peak currents are always very similar, and both increases as array dimension increases. Read peak current is primarily caused by sense amplifier switching and secondly by cells pull-down transistors. Write current is primarily caused by write driver pass transistors pulling down bit-lines and secondly by switching of the two inverters driven by input datum. For the SF process corner (Figure 5.4b), read currents are remarkably reduced, especially for the largest array: this is caused by reduction of sense amplifiers bias current and cells pull-down transistors current. For write currents a slight increment can be noticed: this is due to increased short-circuit current of write driver inverters.

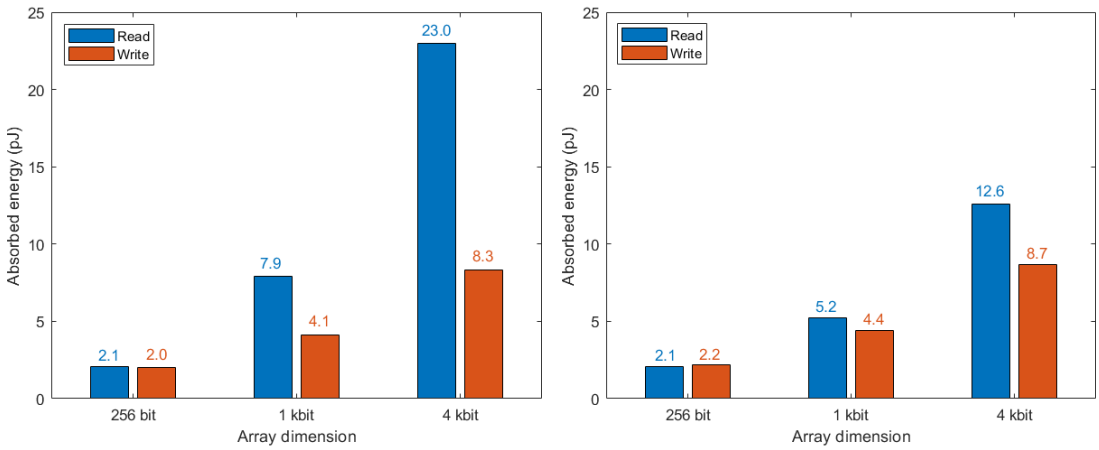


(a) TT corner **(b)** SF corner

Figure 5.4: Read and write peak currents for different array dimensions

5.3.3 Absorbed Energies

Figure 5.5 shows read and write absorbed energies for the three simulated array sizes. An arithmetical mean has been carried out for logic '1' and '0' currents since they present similar values. For the TT process corner (Figure 5.5a), read energy is similar to the write one only for the smallest array, while it is approximately double than the write one for bigger arrays: this is because of sense amplifiers that sink a significant amount of current as long as SAE signal is active, while write drivers sink current only during bit-line discharge that is very rapid. For the SF process corner (Figure 5.5b), read energies are greatly reduced, especially for larger arrays: this is because of the significant reduction in read currents which counterbalance delay increase. For write energies a slight increase is observed, caused by increase of both write current and delay.

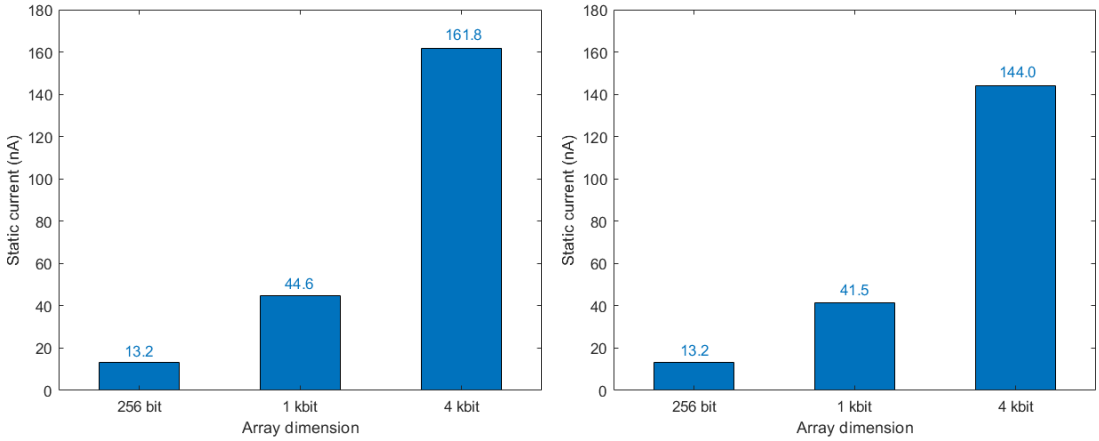


(a) TT corner **(b)** SF corner

Figure 5.5: Read and write absorbed energies for different array dimensions

5.3.4 Static Current

Figure 5.6 shows static current consumption for the three simulated array sizes. As expected, since each array is four times larger than the previous, static current increases significantly with increasing array size. SF process corner static currents (Figure 5.6b) are lower than TT ones (Figure 5.6a), especially for larger arrays, because NMOS pull-down transistors are the main responsible for leakages, since they are the biggest transistors of the cell.



(a) TT corner **(b)** SF corner

Figure 5.6: Static current for different array dimensions

Chapter 6

Conclusions and Future Work

In this work a modular and flexible array compiler that uses a 6T-SRAM cell in 180 nm TSMC technology with customizable transistor widths has been presented. Moreover, the software is fully integrated in Cadence Virtuoso suite and each produced SRAM component can be accessed and modified through its tools by the user. The following results have been achieved:

- Schematic design of 6T-SRAM cell, considering read-stability and write-ability opposing constraints. An optimal design has been found, product of a trade-off among read and hold stability, write-ability, cell area, and energy consumption;
- Layout design of 6T-SRAM cell, employing state-of-the-art "thin" layout. Cadence PCell concept have been applied, so that each transistor can be modified in its width. This is fundamental to have a customizable cell area, so that the user can decide for high density/low power, high performance memory or a trade-off between the two;
- Schematic design of essential column peripheral circuits (i.e. precharge circuit, sense amplifier, and write driver), needed to properly test the generated array. They have been designed with maximum performance and simplicity as main driving constraints;
- Array compiler design, mainly employing Cadence SKILL scripting language. SKILL scripts perform cell, peripherals and array schematic cell view generation and cell layout cell view creation;
- Automated cell DRC, LVS, and parasitic extraction using Calibre software by means of a bash script, which also extract cell netlist and layout GDS file;

- Test-bench automated generation with user-customized timing parameters, thanks to which minimum setup/hold times, pulse widths and word-line period can be easily found;
- Automated write/read transient simulation of the test-bench using an OCEAN script, which measure delays, peak currents, absorbed energies, and static current;
- Comparison of compiler performances and simulation results for three different array sizes (256 bit, 1 kbit, and 4 kbit) and two process corners (TT and SF).

In conclusion, this work represents a first step towards an open-source, flexible and highly customizable SRAM memory generator. A lot is still to be done to develop a complete memory generator.

Future works may include, starting from the most important ones:

- Complete array layout generation, DRC, LVS, and parasitic extraction;
- Layout design of peripheral circuits;
- Schematic and layout design of row decoders and column multiplexers;
- Timing generator design, especially with a self-timed approach;
- Generation of the complete memory layout;
- Generation of liberty files with delay and power reports, useful for digital synthesis automation;
- Implementation of other cell topologies, such as multi-ported cells used in register files and caches;
- Compiler porting to more recent technologies, especially FinFET;
- Development of a compiler GUI for a more user-friendly software.

Bibliography

[1] Andrei Pavlov and Manoj Sachdev. *CMOS SRAM Circuit Design and Parametric Test in Nano-Scaled Technologies: Process-Aware SRAM Design and Test*. en. Frontiers in Electronic Testing. Springer Netherlands, 2008. ISBN: 978-1-4020-8362-4. DOI: 10.1007/978-1-4020-8363-1 (cit. on pp. 1, 4, 7–24, 27, 39).

[2] M. R. Guthaus, J. E. Stine, S. Ataei, Brian Chen, Bin Wu, and M. Sarwar. «OpenRAM: An open-source memory compiler». In: *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. ISSN: 1558-2434. Nov. 2016, pp. 1–6. DOI: 10.1145/2966986.2980098 (cit. on p. 2).

[3] Zhongyuan Wu, Zhiqiang Gao, and Xiangqing He. «Development of a deep submicrometer embedded SRAM compiler». In: *10th IEEE International Conference on Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003*. Vol. 2. Dec. 2003, 707–710 Vol.2. DOI: 10.1109/ICECS.2003.1301883 (cit. on p. 2).

[4] *Dolphin Technology - Memory Products*. URL: http://www.dolphin-ic.com/memory-products.html (cit. on p. 2).

[5] Neil H. E. Weste and David Money Harris. *CMOS VLSI design: a circuits and systems perspective*. en. 4th ed. Boston: Addison Wesley, 2011. ISBN: 978-0-321-54774-3 (cit. on pp. 3–6, 16, 19, 21, 23–25).

[6] M. Gries. «The impact of recent DRAM architectures on embedded systems performance». In: *Proceedings of the 26th Euromicro Conference. EUROMICRO 2000. Informatics: Inventing the Future*. Vol. 1. 2000, 282–289 vol.1. DOI: 10.1109/EURMIC.2000.874644 (cit. on p. 4).

[7] R. Jacob Baker. *CMOS: circuit design, layout, and simulation*. 3rd ed. IEEE Press series on microelectronic systems. Piscataway, NJ : Hoboken, NJ: IEEE Press ; Wiley, 2010. ISBN: 978-0-470-88132-3 (cit. on pp. 5, 6, 25–27).

[8] Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolic. *Digital Integrated Circuits: A Design Perspective*. English. Second Edition. Upper Saddle River, N.J: Pearson College Div, Dec. 2002. ISBN: 978-0-13-090996-1 (cit. on pp. 5, 6, 8, 9, 11, 13, 14, 24–26).

[9] Tegze P. Haraszti. *CMOS Memory Circuits*. en. Springer US, 2002. ISBN: 978-0-7923-7950-8. DOI: 10.1007/b117067 (cit. on pp. 5, 24, 26).

[10] Kiyoo Itoh, Masashi Horiguchi, and Hitoshi Tanaka, eds. *Ultra-Low Voltage Nano-Scale Memories*. en. Integrated Circuits and Systems. Springer US, 2007. ISBN: 978-0-387-33398-4. DOI: 10.1007/978-0-387-68853-4 (cit. on p. 6).

[11] Kiyoo Itoh. *VLSI Memory Chip Design*. en. Springer Series in Advanced Microelectronics. Berlin Heidelberg: Springer-Verlag, 2001. ISBN: 978-3-540-67820-5. DOI: 10.1007/978-3-662-04478-0 (cit. on p. 7).

[12] E. Grossar, M. Stucchi, K. Maex, and W. Dehaene. «Read Stability and Write-Ability Analysis of SRAM Cells for Nanometer Technologies». In: *IEEE Journal of Solid-State Circuits* 41.11 (Nov. 2006), pp. 2577–2588. ISSN: 1558-173X. DOI: 10.1109/JSSC.2006.883344 (cit. on pp. 11, 15, 16).

[13] Meenakshi Devi, Charu Madhu, and Nidhi Garg. «Design and analysis of CMOS based 6T SRAM cell at different technology nodes». en. In: *Materials Today: Proceedings*. International Conference on Aspects of Materials Science and Engineering 28 (Jan. 2020), pp. 1695–1700. ISSN: 2214-7853. DOI: 10.1016/j.matpr.2020.05.130 (cit. on p. 11).

[14] A. J. Bhavnagarwala, Xinghai Tang, and J. D. Meindl. «The impact of intrinsic device fluctuations on CMOS SRAM cell stability». In: *IEEE Journal of Solid-State Circuits* 36.4 (2001), pp. 658–665. DOI: 10.1109/4.913744 (cit. on p. 15).

[15] F. Yang, J. Hwang, and Y. Li. «Electrical Characteristic Fluctuations in Sub-45nm CMOS Devices». In: *IEEE Custom Integrated Circuits Conference 2006*. 2006, pp. 691–694. DOI: 10.1109/CICC.2006.320881 (cit. on p. 15).

[16] J. Lohstroh. «Static and dynamic noise margins of logic circuits». In: *IEEE Journal of Solid-State Circuits* 14.3 (June 1979), pp. 591–598. ISSN: 1558-173X. DOI: 10.1109/JSSC.1979.1051221 (cit. on p. 15).

[17] E. Seevinck, F. J. List, and J. Lohstroh. «Static-noise margin analysis of MOS SRAM cells». In: *IEEE Journal of Solid-State Circuits* 22.5 (Oct. 1987), pp. 748–754. ISSN: 1558-173X. DOI: 10.1109/JSSC.1987.1052809 (cit. on pp. 15–18).

[18] Jiajing Wang, Satyanand Nalam, and Benton Calhoun. *Analyzing static and dynamic write margin for nanometer SRAMs*. Pages: 134. Jan. 2008. DOI: 10.1145/1393921.1393954 (cit. on pp. 15, 19).

[19] Manjul Bhushan and Mark B. Ketchen. *CMOS test and evaluation: a physical perspective*. en. New York: Springer, 2015. ISBN: 978-1-4939-1348-0 (cit. on p. 16).

[20] F. J. List. «The Static Noise Margin of SRAM cells». In: *ESSCIRC '86: Twelfth European Solid-State Circuits Conference*. Sept. 1986, pp. 16–18. DOI: 10.1109/ESSCIRC.1986.5468249 (cit. on p. 19).

[21] R. Kolhal and V. Agarwal. «A Power and Static Noise Margin Analysis of different SRAM cells at 180nm Technology». In: *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*. June 2019, pp. 6–12. DOI: 10.1109/ICECA.2019.8821868 (cit. on p. 19).

[22] M. Ishida, T. Kawakami, A. Tsuji, N. Kawamoto, M. Motoyoshi, and N. Ouchi. «A novel 6T-SRAM cell technology designed with rectangular patterns scalable beyond 0.18 /spl mu/m generation and desirable for ultra high speed operation». In: *International Electron Devices Meeting 1998. Technical Digest (Cat. No.98CH36217)*. ISSN: 0163-1918. Dec. 1998, pp. 201–204. DOI: 10.1109/IEDM.1998.746322 (cit. on pp. 19, 43).

[23] Dimitrios Balobas and Nikos Konofaos. «Design and evaluation of 6T SRAM layout designs at modern nanoscale CMOS processes». en. In: (2015), p. 4 (cit. on pp. 19–21).

[24] M. Khare et al. «A high performance 90nm SOI technology with 0.992 /spl mu/m2 6T-SRAM cell». In: *Digest. International Electron Devices Meeting,* 2002, pp. 407–410. DOI: 10.1109/IEDM.2002.1175865 (cit. on pp. 19, 21).

[25] R. W. Mann and B. H. Calhoun. «New category of ultra-thin notchless 6T SRAM cell layout topologies for sub-22nm». In: *2011 12th International Symposium on Quality Electronic Design*. 2011, pp. 1–6. DOI: 10.1109/ISQED.2011.5770761 (cit. on p. 19).

[26] J. S. Caravella. «A low voltage SRAM for embedded applications». In: *IEEE Journal of Solid-State Circuits* 32.3 (1997), pp. 428–432. DOI: 10.1109/4.557643 (cit. on p. 22).

[27] B. S. Amrutur and M. A. Horowitz. «A replica technique for wordline and sense control in low-power SRAM's». In: *IEEE Journal of Solid-State Circuits* 33.8 (1998), pp. 1208–1219. DOI: 10.1109/4.705359 (cit. on p. 23).

[28] Y. Tao and W. Hu. «Design of Sense Amplifier in the High Speed SRAM». In: *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. Sept. 2015, pp. 384–387. DOI: 10.1109/CyberC.2015.32 (cit. on p. 26).

[29] V. Borisov. «Development of parameterized cell using Cadence Virtuoso». In: *East-West Design Test Symposium (EWDTS 2013)*. 2013, pp. 1–2. DOI: 10.1109/EWDTS.2013.6673156 (cit. on pp. 30, 41).

[30] Jan Rabaey. *Low Power Design Essentials*. en. Integrated Circuits and Systems. Springer US, 2009. ISBN: 978-0-387-71712-8. DOI: 10.1007/978-0-387-71713-5 (cit. on pp. 33, 37, 61).

[31] H. J. M. Veendrick. «Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits». In: *IEEE Journal of Solid-State Circuits* 19.4 (1984), pp. 468–473. DOI: 10.1109/JSSC.1984.1052168 (cit. on p. 37).

[32] *Virtuoso Parameterized Cell Reference*. en. 2018 (cit. on p. 41).

[33] G. Pasandi and S. M. Fakhraie. «A new sub-300mV 8T SRAM cell design in 90nm CMOS». In: *The 17th CSI International Symposium on Computer Architecture Digital Systems (CADS 2013)*. 2013, pp. 39–44. DOI: 10.1109/CADS.2013.6714235 (cit. on p. 42).

[34] R. Sarpeshkar, J. L. Wyatt, N. C. Lu, and P. D. Gerber. «Mismatch sensitivity of a simultaneously latched CMOS sense amplifier». In: *IEEE Journal of Solid-State Circuits* 26.10 (1991), pp. 1413–1422. DOI: 10.1109/4.90096 (cit. on p. 45).

[35] *Cadence SKILL Language Reference*. en. 2018 (cit. on p. 50).

[36] *Virtuoso Design Environment SKILL Reference*. en. 2018 (cit. on p. 51).

[37] *OCEAN XL Reference*. en. 2018 (cit. on p. 51).

[38] Y. Chen, S. Huang, and Y. Chang. «Rapid and Accurate Timing Modeling for SRAM Compiler». In: *2009 IEEE International Workshop on Memory Technology, Design, and Testing*. 2009, pp. 73–76. DOI: 10.1109/MTDT.2009.26 (cit. on p. 63).