

Exploring SRAM Layout Using Open-Source Tools: A Case Study with Skywater 130nm PDK, Magic, Xschem, ngSpice, and OpenRAM

Om Bhatt

Dept. of Electronics and Telecom.

Engg

NMIMS Mukesh Patel School of Tech.

Mgt. & Engg.

Mumbai ,India

om.bhatt@nmims.in

Darshana Sankhe

Dept. of Electronics and Telecom.

Engg

D. J. Sanghvi College of Engineering

Mumbai, India

darshana.sankhe@djsce.ac.in

Avinash More

Dept. of Electronics and Telecom

Engg

NMIMS Mukesh Patel School of Tech.

Mgt. & Engg.

Mumbai ,India

avinash.more@nmims.edu

Abstract— This paper presents a comprehensive exploration of a 6T SRAM cell design and implementation using a suite of open-source tools, including Magic, OpenRAM, xschem, and ngspice. The research encompasses the complete design flow, from individual component design and simulation (6T SRAM cell, D flip-flop, write driver) to the generation of a 16KB SRAM array layout using OpenRAM. The design is implemented using the Skywater 130nm PDK, ensuring compatibility with industry-standard fabrication processes. Simulation results validate the functionality and performance of the designed components. OpenRAM facilitates the visualization and analysis of the complete memory array, providing insights into layout complexities and the impact of design choices on overall performance. This study underscores the efficacy of open-source tools for accessible and cost-effective SRAM design, paving the way for further innovation in memory system development

Keywords— SRAM, 6T SRAM cell, OpenRAM, Magic VLSI, Ngspice, Skywater 130nm PDK, memory design, open-source EDA tools.

I. INTRODUCTION

The design and verification of analog circuits necessitates specialized tools that provide precise simulation, layout, and analysis. These tools, offered by prominent EDA companies like Cadence, Synopsys, and Mentor Graphics, handle nonidealities and process variations crucial for reliable analog performance meticulously. However, the high licensing costs of these industry-grade EDA tools present a barrier for small enterprises, educational institutions, and researchers, especially in resource-constrained regions. Even with educational discounts, the expense can still be considerable. As a result, open-source EDA tools are gaining momentum as viable, cost-effective alternatives, offering most of the critical features of commercial tools, while requiring users to navigate their design flows independently and develop a strong grasp of analog principles.

This paper aims to examine the potential of open-source tools in the design and verification of analog layouts, with a particular focus on SRAM circuits, a fundamental component in memory systems. SRAM design presents unique challenges, necessitating a thorough understanding of transistor-level layouts, parasitic effects, and precise matching of circuit elements to ensure optimal performance. By leveraging the Skywater 130 Process Design Kit, the study guides the design of an SRAM cell using a suite of open-source tools: Xschem for schematic capture, Ngspice for simulation, Magic VLSI for layout, and OpenRAM to observe SRAM memory banks. Through this exploration, students,

researchers, and small design teams can carry out the essential steps in analog layout without the burden of licensing costs, gaining practical knowledge and a hands-on approach to the intricacies of layout design. This open-source design approach provides them with useful experience in understanding the way analog circuits operate and are laid out, typically available only through high-cost EDA packages.

Developing mixed-signal circuit design is one of the critical research areas and some very important product development in semiconductor technologies, providing platforms which bridge digital to real world signals. The research is specifically targeted at the design of SRAM cells [1], also—a fundamental building block for most ICs. Much like all SRAM designs it raises unique problems with its requirements specification including both operation stability, extreme low power consumption at full performance speed and area efficient. Adding to the complexity, during physical layout power plan synthesis many complex operations are carried out for this optimal functionality aspect of a circuit on how it functions properly within restrictive dimensions based on process technology [2].

As standard practice, analog layout design and verification require proprietary high-cost software which makes it intractable for academic or open-source research projects. The emergence of open-source design tools and publicly available processing kits like the Skywater 130nm PDK provides a special chance to liberate this domain. Higher-level universities and research labs can experiment with SRAM layout design using publicly available tools such as Magic for layout, Xschem for schematic capture, Ngspice for simulation and OpenRAM to generate an SRAM memory bank without having to worry about the costly EDA software[3][4].

In this paper we attempt to delve into the analog layout of SRAM for it is usually closed source by making use of these open-source tools. The research will follow the complete design and layout flow using Skywater 130nm PDK, based on schematic capture through to physical artwork generation and verification highlighting specific aspects of analog IC layout. The following tools and methods are used in the research [5].

Skywater 130nm Process Design Kit: The Skywater 130nm PDK include all necessary technology files and process parameters to design, simulate, and verify ICs in the sky water production level fabrication processes at a realistic process node of today for industry-relevant research [6].

Magic VLSI Layout Tool: Magic is an open-source layout editor with comprehensive capabilities for IC design typically used in academia. The latter is an interactive environment that may be used to hand write and optimize the layout, making it a learning aid for designers who are interested in understanding such intricate details of analog layouts as symmetry principles, matching or layout dependent effects [7][8].

Both Xschem and Ngspice: a mixed-signal simulator , great for generating schematics the latest stable release should have decent bug support. Both are key in enabling the simulation of SRAM circuit at schematic level to get a design verification and know before going ahead making it as layout [9][10][11].

OpenRAM: Open-source Static Random-Access Memory Family The next step is to make use of this generator in generating the16kb SRAM memory bank which has its own custom layout cells that mimics analog layout targets for some of its basic cells [12].

The research commences with the study of different SRAM cell architectures properties, design considerations given in [13] [14].

Based on the prior research, this paper intends to take a shot at understanding and providing analog layout flow for SRAM cells by using these opensource tools, giving insights into few of the challenges faced in physical design process used.

II. METHODOLOGY

For this study, the proposed research methodology is a holistic solution for developing and validating SRAM components through open-source tools combined with Skywater 130nm PDK. The key steps are as follows:

Setting up the Environment with SKY130 PDK

The SKY130 PDK is an open-source design kit for 130nm CMOS technology, including libraries and device models necessary to perform digital, analog or mixed-signal designs with standard cell. It includes the files and configuration for each of these tools, such as technology file SPICE models library etc. The PDK provides optimized n-channel and p-channel MOSFET transistors for 130nm technology, which are required to build the analog blocks of SRAM, which operate at 1.8 volts.

A. Component-Level Design and Validation

The schematics of each SRAM component, including the 6T cell, D flip-flop and sense amplifier are drawn in Xschem following conventional methods for operation of standard SRAM circuits. The schematics have been further simulated in Ngspice to check the electrical performance like threshold voltages, timing and stability at different corners.

B. Magic Layout Design:

We convert verified schematics into layouts in the Magic VLSI layout tool, making each component adhere to 130nm technology design rules (transistor sizing and metal routing/spacing parameters).

C. Design rule checks:

These are applied to validate that the designs conform not only with design guidelines but also to abide by physical aspects of such technology.

D. The SRAM Memory Bank in OpenRAM:

We use OpenRAM to have a visualization of how the whole of the structure of 16kb SRAM memory bank 16kb would look like. The arrangement of memory cells as well as replica and corner cells. Therefore, OpenRAM helps to provide valuable insight into the structure of SRAM memory banks and routing schemes assisting in comprehending integration of the SRAM components into a larger memory array and the layout strategies used for efficiency.

III. IMPLEMENTATION

The implementation starts by designing schematics for the core components necessary to SRAM cell operation. Furthermore, the research expands upon this component-level approach by generating a 16KB SRAM memory bank using OpenRAM, giving us an idea of how the memory bank with several of such cells would look like.

The 6-transistor SRAM cell is the heart of SRAM memory. It's built to store data reliably, using cross-coupled inverters to keep a single data bit without using power when it's not active. The cell uses complementary bit lines for smooth reading and writing. Its layout and transistor arrangement are fine-tuned to reduce data disturbances during access, which helps cut down on leakage and ensures it lasts through frequent use.

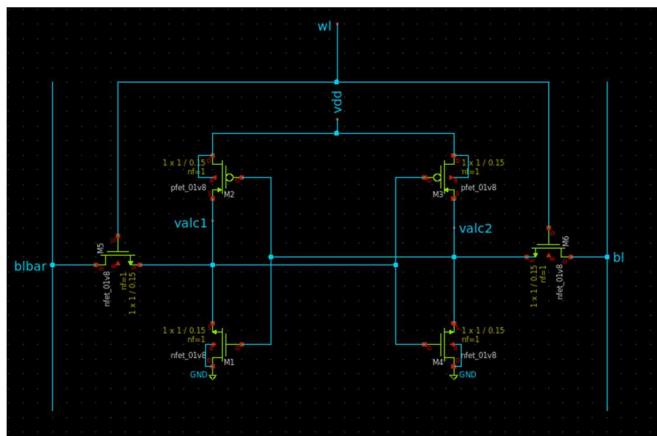


Fig. 1. 6T SRAM Cell in Xschem

Write Circuit: The SRAM write circuitry allows data input by changing the state of the bit lines. When a write command is issued, the circuit transmits a new data value to the SRAM cell while isolating adjacent cells, preventing interference across the array.

Read Circuit: A key characteristic of SRAM memory is its non-interruptive read process, which reads data from the cell without changing its state. This read mechanism is achieved by detecting voltage differences between the bit lines without modifying the cell contents using a differential sense amplifier . The read circuit's ability to maintain voltage stability on the bit lines enhances read reliability, making it suitable for high-frequency applications. By isolating the read process, the

circuit minimizes disruptions and preserves data accuracy across multiple reads.

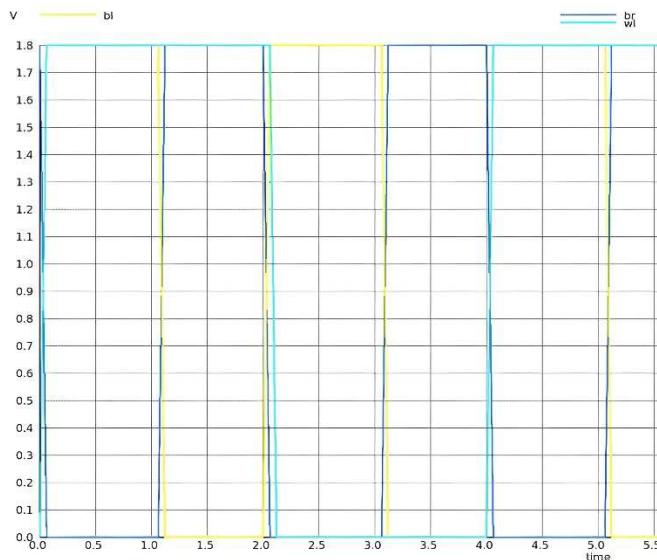


Fig. 2. 6T SRAM simulation in NgSpice

Precharge Circuit: To enhance each read and write cycle, the precharge circuit equally drives the bit lines by setting them to a defined voltage level before every operation. This precharging speeds up access time by reducing the signal transitions required during read and write operations.

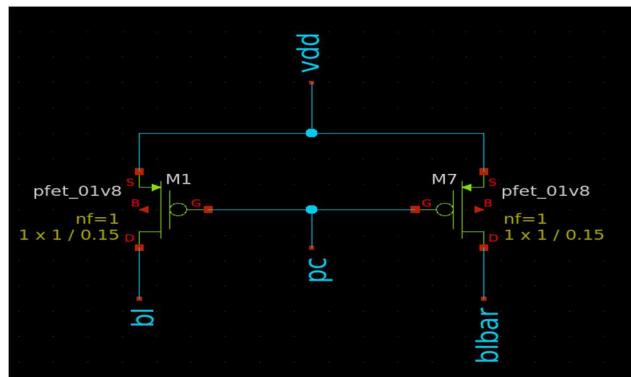


Fig. 3. Pre-Charge Circuit

Sense Amplifier:

Sense Amplifier: During read cycles, this amplifier boosts the small voltage differences on the bit lines, making data retrieval faster and more accurate. It ensures clear data outputs and keeps SRAM performance top-notch in terms of speed and signal clarity.

The sense Amplifier is necessary because in analog perspective the voltage is not always 0 or 1, it might be a middle value, therefore in order to avoid errors and read mismatch a clear differential sense amplifier is necessary.

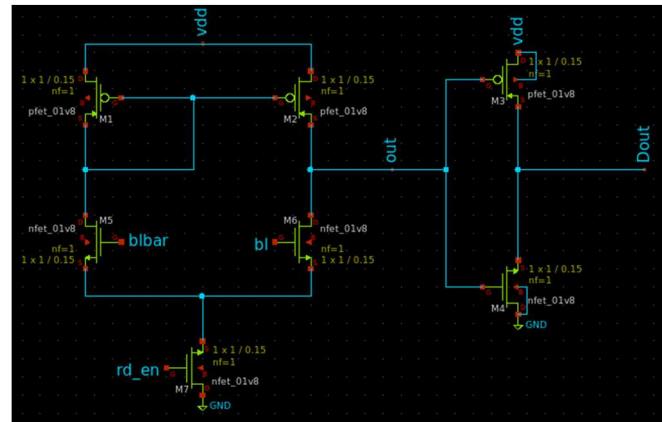


Fig. 4. Sense Amplifier in Xschem

Write Driver Circuit: This circuit applies the right data signals to the bit lines during write operations. Controlled by a Write Enable signal, it writes data into the cell when WE = 1 and does nothing when WE = 0.

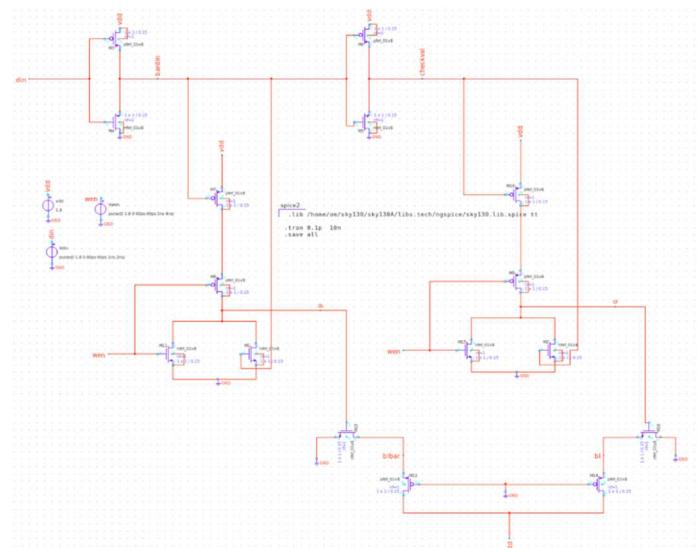


Fig. 5. Write Driver circuit with bit lines precharged

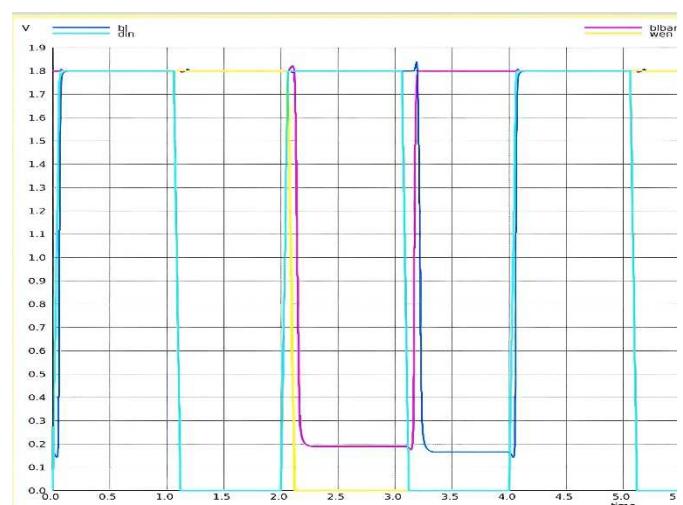


Fig. 6. Write driver Simulation using Ngspice

Flip-Flop: The D flip-flop (positive edge triggered) works like a storage mechanism within the SRAM framework, controlled by a clock signal to manage data input and synchronization. It ensures consistent timing across operations, which is crucial for larger memory arrays to enable orderly data access.

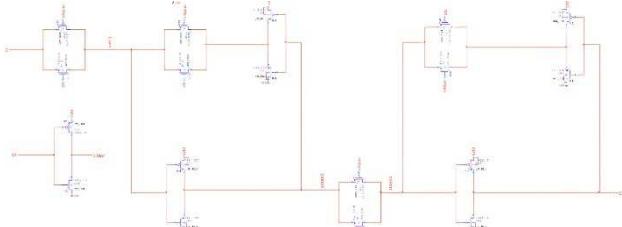


Fig. 7. D flip flop schematic in Xschem

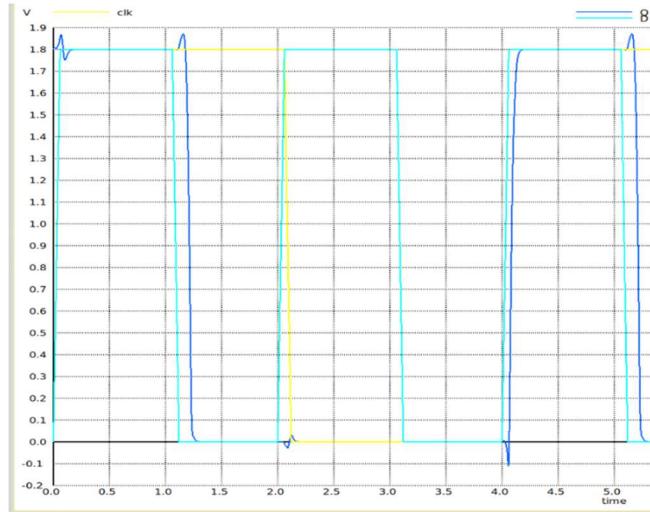


Fig. 8. D Flip Flop spice simulation

OpenRAM -This describes the setup and outputs of the open source SRAM compiler . After the individual SRAM components were developed and validated, we made OpenRAM generate a 16KB memory bank for visualization of a full memory array layout. The setup file dictates details such as word size, the number of words, and bit cell type. This setup file uses process technology details from the Skywater 130nm PDK, ensuring compatibility with OpenRAM that generates a layout according to technological physical constraints. By executing OpenRAM with the Skywater 130 PDK, the setup file will specify paths to the device models and design rules of the PDK. This setup file is now more critical than ever because it allows OpenRAM to work in tandem with physical features of the 130nm technology, ensuring that laid-out circuits meet expected process requirements.

OpenRAM Outputs: When Run, OpenRAM produces several outputs.

Layout (GDSII format): This output provides a graphical representation of the memory array's physical layout, showing how SRAM cells, bit lines, and control signals are arranged.

SPICE Netlist: A detailed netlist is generated, allowing for further circuit simulations to verify the performance and reliability of the memory bank.

These outputs construct a comprehensive view of how individual SRAM cells are structured and organized within a more giant array: while our custom-designed cells never enter

the output of OpenRAM, the visualization of a complete SRAM array renders a comprehensive understanding concerning layout intricacies, memory tiling, and connectivity to memory cells with bit lines and word lines.

IV. RESULTS

After plotting the schematic and viewing their waveform outputs, it is evident that the memory cell design and its constituent components operate as expected. Now we can plot the layout using the magic tool. The layout of 6T SRAM cell is shown along with D flip flop and write driver circuit .Note that write driver circuit waveform does not accurately match the schematic waveform due to no Precharge circuit .

The 6T SRAM Cell Layout involves the precise positioning of six transistors to form two inverters and two access transistors, resulting in a compact and symmetrical design. Careful attention to transistor spacing and alignment minimizes parasitic capacitance, ensuring stable operation during read and write cycles.

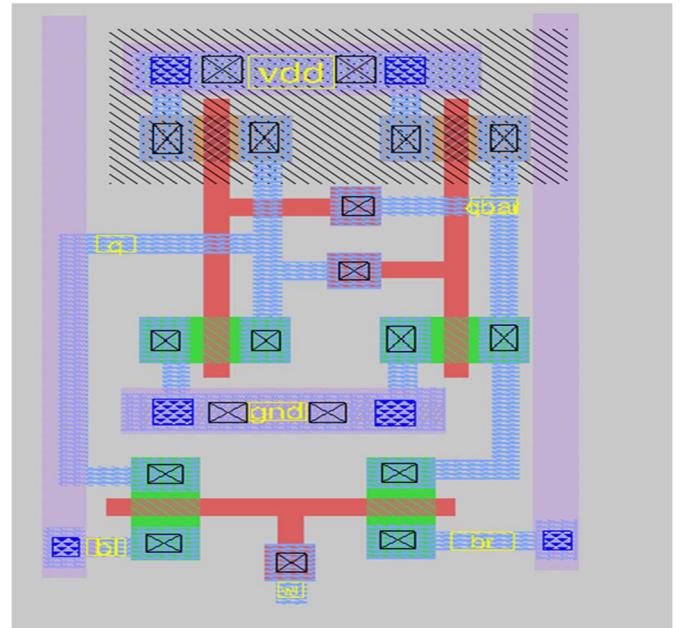


Fig. 9. Magic Layout of 6T SRAM

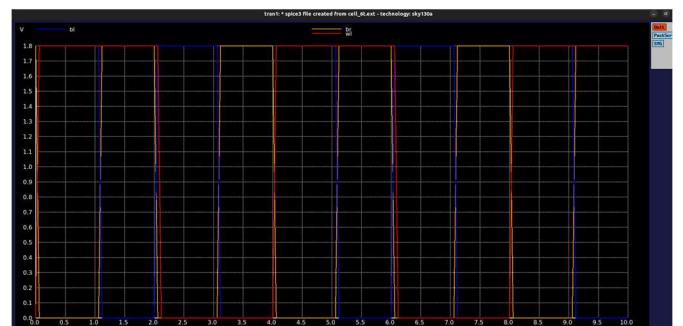


Fig. 10. Spice Simulation of SRAM layout

The D Flip-Flop Layout is done using a series of NMOS and PMOS transistors organized in a specific arrangement to enable efficient clocked data storage. Unlike the schematic

the layout is different because several PMOS and NMOS can share the same well.

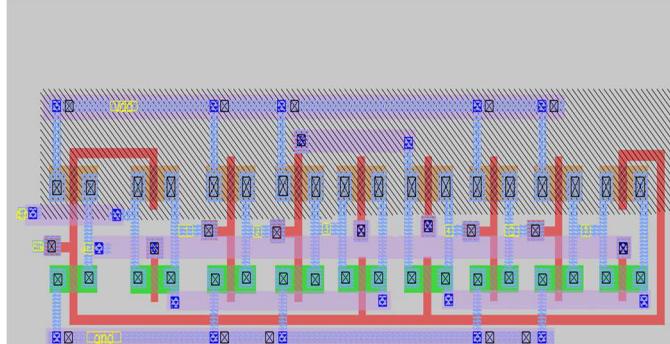


Fig. 11. Magic Layout of D Flip Flop

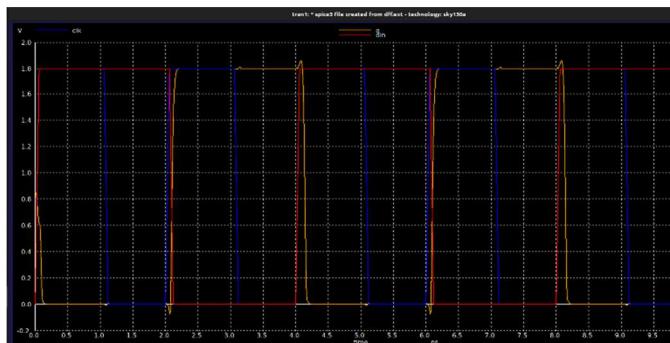


Fig. 12. Spice simulation of D Flip flop layout

Layout of Write Driver Circuit is done without the Precharge Circuit. As we can see the write driver circuit has inputs like din and write enable. Write Enable acts as a control signal for the circuit , if it is high the bit line gets charged to the value of Din.

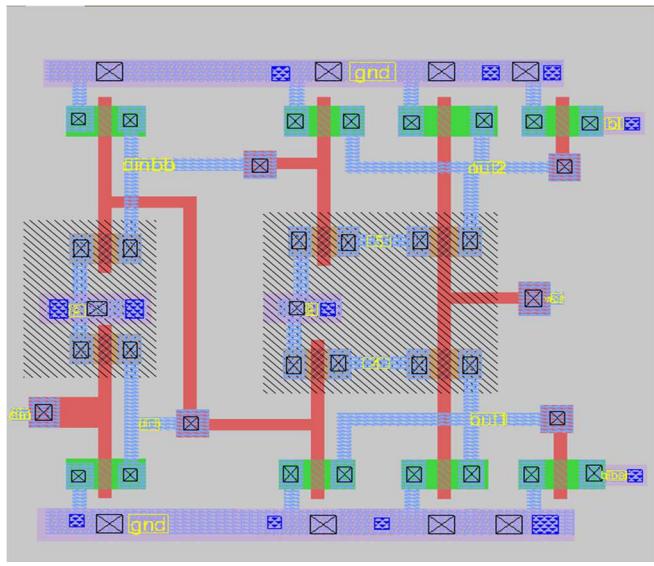


Fig. 13. Magic Layout of Write Driver Circuit

The visual representations presented in Figures 15, 16, and 17 illustrate the layout of a 16KB SRAM bank generated using OpenRAM, offering insights into the structural organization and layering within the SRAM array.

Figure 15 showcases the top-level layout view of the 16KB SRAM bank, depicting the entire memory array composed of

repetitive SRAM cells organized into a dense grid structure, with control and peripheral circuitry surrounding the memory core. The hierarchical design enables effective address decoding, bit-line and word-line control, and efficient access to individual cells within the bank.

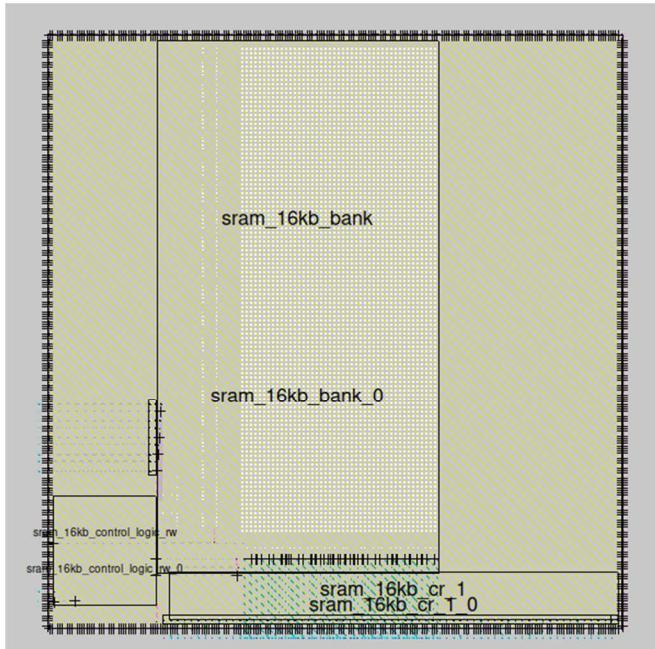


Fig. 14. 16 kb SRAM bank generated by OpenRAM

Figure 16 displays an expanded view of the SRAM bank, revealing the intricate layers and interconnections in greater detail. This layered perspective highlights the complexity of the SRAM layout, with metal layers utilized for routing power, ground, and signals, while ensuring adherence to design rules.

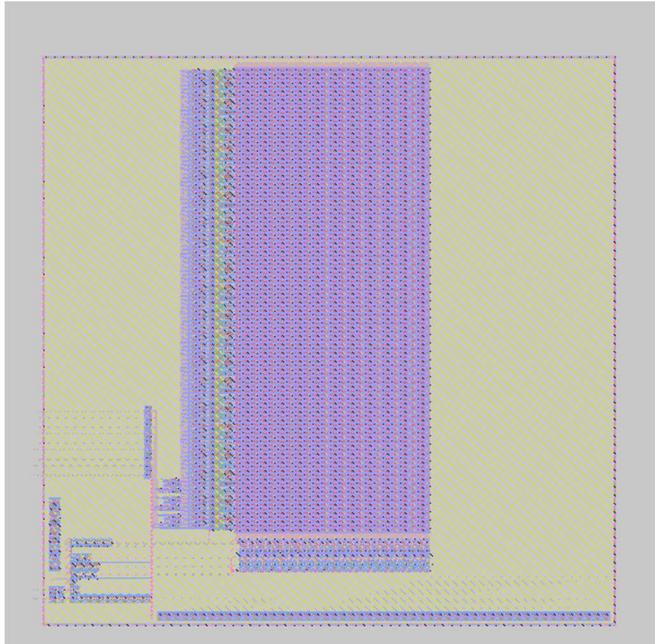


Fig. 15. Expanded version to view all layers of SRAM bank

Figure 17 provides a zoomed-in snapshot of a section within the SRAM array, clearly showcasing the regular pattern of SRAM cells. Each cell in this array follows a standard 6-transistor configuration, designed for minimal footprint and

optimized electrical performance. This close-up view also unveils the intricate arrangement of vias and metal layers, crucial for signal propagation and robust cell operation.

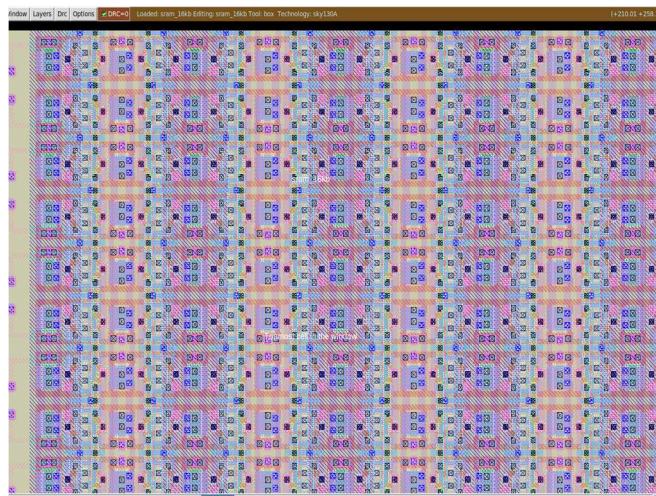


Fig. 16. Zoomed in picture to view layout in detail

Area Analysis of the SRAM Layout

The area of the designed SRAM bank and its complete layout was analyzed using the Magic tool to evaluate the compactness and feasibility of the layout. The analysis considered both the root cell box dimensions and the internal area occupied by the layout, with measurements provided in microns and lambda units for precision.

Description	Microns (μm)	Area (μm^2)	Lambda (λ)	Area (λ^2)	Internal Dimensions (λ)	Internal Area (λ^2)
Root Cell Box	238.950 × 451.005	1,07,767.64	23,895.00 × 45,100.50	1,07,76,76,416	47,790 × 90,201	4,31,07,05,790
Complete SRAM Layout	487.180 × 502.275	2,44,698.33	48,718.00 × 50,227.50	2,44,69,83,424	97,436 × 100,455	9,78,79,33,380

Fig. 17. Area analysis of the SRAM layout

Addressing Validation and Compatibility with Industry-Standard Tools

The SRAM design leveraged open-source tools like Magic for Design Rule Checks (DRC) and Layout Versus Schematic (LVS) validation, employing Skywater 130nm PDK rule decks that align with commercial EDA standards. This ensures compliance with industry-standard workflows and provides confidence in the design's manufacturability.

Designs were exported in GDSII format and SPICE netlists, ensuring compatibility with proprietary EDA platforms such as Cadence Virtuoso Suite. Although direct testing on these tools was not performed, the ability to generate universally accepted outputs guarantees interoperability with industry-standard environments. The Skywater 130nm PDK, validated by the open-source community and widely used in fabricated designs, reinforces the reliability of this approach. Community-driven validation and successful projects using tools like Magic and OpenRAM bolster the credibility of the methodology.

By adhering to standardized workflows and leveraging universally recognized formats, this research aligns with industry standards, proving that open-source tools can deliver SRAM designs compatible with proprietary environments.

V. CONCLUSION

This research comprehensively explored the design and implementation of a 6T SRAM cell using open-source tools like Magic, OpenRAM, xschem, and ngspice. The project encompassed the entire design flow, from individual component design and simulation to the generation of a complete 16KB SRAM array layout.

The successful simulation of the 6T SRAM cell, D flip-flop, and write driver circuit validated the functionality and performance of the proposed design. Utilizing OpenRAM facilitated the visualization and analysis of a full-scale SRAM array, providing valuable insights into layout complexities, memory organization, and the impact of design choices on overall performance.

This study demonstrates the efficacy of open-source tools in facilitating accessible and cost-effective SRAM design, paving the way for further exploration and innovation in memory system development. Future research could focus on optimizing the design for specific applications, exploring alternative SRAM architectures, and investigating advanced layout techniques for improved performance and density.

REFERENCES

- [1] Akashe, Shyam & Sharma, Sanjay. (2013). High density and low leakage current based SRAM cell using 45 nm technology. International Journal of Electronics. 100. 536-552. 10.1080/00207217.2012.713023.
- [2] A. Amerasekera and C. Bittlestone, "Ultra low power design and future device interactions," in Proceedings of the 2004 International Symposium on Low Power Electronics and Design, 2004, pp. 300-305.
- [3] T. Ajayi, A. Payne, and R. Manohar, "Toward an Open-Source Digital Flow," IEEE Solid-State Circuits Magazine, vol. 13, no. 1, pp. 48-55, 2021.
- [4] M. P. Lin, Y. Chang, and C. Hung, "Recent research development and new challenges in analog layout synthesis," in Proceedings of the IEEE Custom Integrated Circuits Conference, 2000, pp. 205-212.
- [5] C. S. H. Kaushik, R. R. Vanjralapati, V. M. Krishna, T. Gautam, and V. Elamaran, "VLSI design of low power SRAM architectures for FPGAs," in Proceedings of the International Conference on VLSI, Communication & Instrumentation, 2011, pp. 401-405.
- [6] Google. "Open source process design kit for usage with SkyWater Technology Foundry's 130nm node." [Online]. Available: <https://github.com/google/skywater-pdk> [Accessed: October 26, 2023].
- [7] R. Rutenbar and J. Cohn, "Layout tools for analog ICs and mixed-signal SoCs: a survey," *Proceedings of the Design Automation Conference (DAC)*, 2000, pp. 76-83, doi: 10.1145/332357.332378.
- [8] J. Scheible and J. Lienig, "Automation of Analog IC Layout: Challenges and Solutions," in *Proceedings of the 2015 Symposium on International Symposium on Physical*

Design (ISPD '15), Monterey, CA, USA, 2015, pp. 33–40, doi: 10.1145/2717764.2717781.

[9] Ngspice User's Manual, Ngspice Software, Version 38, 2023. [Online]. Available: <https://sourceforge.net/projects/ngspice/files/ng-spice-rework/>

[10] R. Saleh, T. Inoue, and S. Ido, "Enhanced circuit simulation: Expectations, problems, implementation, and integration," in Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers & Processors, 1991, pp. 28-31.

[11] B. Wan and X. Wang, "Overview of commercially-available analog/RF simulation engines and design environment," in Proceedings of the IEEE Radio and Wireless Conference, 2003, pp. 343-346.

[12] "OpenRAM: An open-source static random access memory compiler." [Online]. Available: <https://github.com/VLSIDA/OpenRAM> [Accessed: October 26, 2023].

[13] J. Cirimelli-Low, M. Guth, B. Bassen, and M. Schoeberl, "SRAM Design with OpenRAM in SkyWater 130nm," in Proceedings of the 2nd Workshop on Open-Source EDA Technology, 2019, pp. 1-6.

[14] V. Elamaran and H. N. Upadhyay, "CMOS VLSI Design of Low Power SRAM Cell Architectures with New TMR: A Layout Approach," International Journal of Computer Applications, vol. 42, no. 11, pp. 23-28, 2012.