International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 3 (2026)

ijerst.editor@gmail.com

editor@ijerst.com

ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

Design-Space Exploration for Macro-Based SRAM Configuration on the SKY130 PDK

Porkkodi Ma P
Lecturer Department of Electronics and Communication Engineering (ECE) Government Polytechnic College for Women, Coimbatore
Email: porkkodi.ece@gmail.com

Dr. G. Kulanthaivel
Professor & Head Department of Electronics and Communication Engineering (ECE) National Institute of Technical Teachers Training and Research (NITTTR), Chennai
Email: gkveldr@gmail.com

Dr. P. Sivasankar
Professor Department of Electronics and Communication Engineering (ECE) National Institute of Technical Teachers Training and Research (NITTTR), Chennai
Email: sivasankar@nitttrc.ac.in

***Abstract*—** The SKY130 Macro Memory Cell Generator interleaves three fixed PDK macro sizes into custom SRAM arrays but evaluates no candidates: memory type and placement style are chosen once, with no automated comparison. This paper replaces that manual step with a search across nine memory-type and placement combinations, scored on die area, static leakage, and estimated half-perimeter wirelength (HPWL). A constraint-aware variant of the same search accepts a maximum die width and height and returns the best candidate that fits, rather than the unconstrained optimum. Area comes from the baseline tool's own placement routine, not a closed-form estimate. Candidate selection moves away from a bare minimum-area rule to a Pareto-optimality filter with a documented weighted tie-break, benchmarked against five other established selection rules — weighted-sum, TOPSIS, rank-sum, lexicographic ordering, and naive minimum-area — across fifteen realistic design sizes (4 KB–1 MB). All six methods agreed completely on only four of fifteen sizes (26.7%), showing that selection-method choice is not a formality.

***Keywords*—** *SRAM Design-Space Exploration, Constraint-Aware Optimization, SKY130 PDK, Pareto-Optimal Selection, Multi-Criteria Decision Making (MCDM), Application-Specific Memory Generator.*

I. INTRODUCTION

ASIC memory rarely matches the sizes a process design kit ships with. A system-on-chip project hands down a word width and depth dictated by architecture, or a fixed silicon budget dictated by die size, and neither number cares what SKY130's three stock SRAM macros happen to offer. Two situations recur. First, a designer holds a fixed area budget and needs the best memory configuration that fits inside it — a defined block of die space, nothing more. Second, a designer starts from a specific word width and address depth and needs that memory implemented as efficiently as possible, since checking every candidate arrangement by hand does not scale past the nine memory-type and placement combinations enumerated in Section VI.

Baungarten-León et al. built the Macro Memory Cell Generator to close part of this gap [1]. Published in IEEE Access in 2024, the tool interleaves SKY130's three fixed macros — 8x1024, 32x256, 32x512 — into arbitrary custom sizes, and produces the Verilog, macro placement file, and OpenLane configuration needed to carry a design from RTL through to GDSII. What it does not do is choose. Memory type and placement style go in exactly as typed, with zero comparison against any alternative the tool could have picked instead.

This work covers both real-world scenarios directly. For the fixed-space case, a constraint-aware search takes a maximum width and height and returns the best feasible configuration inside that envelope. For the fixed-requirement case, manual memory-type and placement selection is replaced by an automated search across nine candidates, scored on real area, real static leakage, and estimated wirelength, with the selection rule itself benchmarked against five established alternatives to confirm it is not an arbitrary choice.

II. PROBLEM STATEMENT

The SKY130 Macro Memory Cell Generator automates interleaving of fixed macros into custom-sized SRAM arrays, but it evaluates none of its own choices. Memory type and placement style are typed in by hand, once, with no automated comparison against any alternative the tool could have produced instead. There is no path from a fixed die-area budget to a suitable configuration — nothing computes the best memory that fits inside a given space. Whatever configuration is typed in is what gets built, better or worse, with no mechanism to check it against alternatives. Area itself is never evaluated by real placement code inside the search, and no principled multi-metric selection rule exists to weigh area against leakage and wirelength together. Therefore, an automated, validated, constraint-aware design-space exploration layer is required on top of the existing generator to overcome these limitations.

A. Contributions

- Automatic search across memory-type and placement- style combinations in place of manual, uncompared selection.
- Area evaluation grounded in the baseline tool's own real placement routine, not a closed-form estimate.
- A Pareto-optimal filter with a documented, equally-weighted tie-break for candidate selection.
- Benchmarking of that selection rule against five established multi-criteria decision-making (MCDM) methods across fifteen realistic sizes, to confirm it is not an arbitrary choice.
- A constraint-aware mode that accepts a maximum die width and height and returns the best feasible configuration for that fixed-space ASIC scenario.
- An open-source extension of the baseline repository, reproducible from the published methodology alone.

III. LITERATURE SURVEY

The Macro Memory Cell Generator for the SKY130 PDK interleaves three fixed SRAM macros — 8x1024, 32x256, and 32x512 — into arbitrary custom sizes, and produces the Verilog, macro placement file, and OpenLane configuration files needed to carry a design from RTL through to GDSII. This tool is the direct baseline for the present work, and it performs no candidate evaluation of its own [1].

OpenRAM automates SRAM compiler generation for the same class of PDKs, but its practical ceiling sits near 4 to 16KB before layout problems start appearing at larger sizes [2].

FastMem pairs CACTI's design-space exploration with OpenRAM to search for area- and power-delay-efficient memory configurations across several process nodes. The approach is entirely theoretical, however; nothing in the pipeline is checked against a real place-and-route run [3].

Multiple Attribute Decision Making surveys the mathematical foundations of methods such as TOPSIS, providing the ideal-point distance ranking approach adopted as one of the six selection methods benchmarked in this paper [4].

Decisions with Multiple Objectives develops multi-attribute utility theory, underpinning the equally-weighted metric-sum approach used here as the weighted-sum comparator method [5].

The Microarchitecture of Pipelined and Superscalar Computers documents why interleaving one large memory unit into several smaller banks improves concurrent access and fault tolerance — the foundational technique both the baseline generator and this work build on [6].

A fine-grained fault-tolerant interleaved memory design on FPGA demonstrates the same interleaving principle applied for fault tolerance rather than custom sizing, confirming the technique's broader relevance to memory system design [7].

Synopsys' educational generic memory compiler offers comparable interleaving flexibility as a closed, licensed commercial tool, illustrating that the underlying capability already exists commercially without an accompanying validated selection methodology [8].

Dolphin Technology similarly markets commercial memory compilers with custom-size flexibility, closed and licensed, reinforcing the gap this paper targets in the open-source tooling space [9].

ReSPIR applies Pareto-based response-surface iterative refinement to application-specific design-space exploration in chip design, directly informing the Pareto-optimal filtering approach adopted as the primary selection method in this paper [10].

The original treatment of Pareto optimality and later work on Borda-count rank aggregation and lexicographic decision rules provide three of the five established comparator methods benchmarked against the Pareto-plus-tie-break approach in Section VI [11], [12].

A comparative study of six multi-criteria decision-making methods on building-design case studies found that most cases produced different solutions, with no single method dominating [13]. To confirm the Pareto-optimal-plus-tie-break method adopted in this paper was not an arbitrary choice, it is benchmarked in Section X against five alternative selection methods on the same SKY130 SRAM candidate data.

IV. RESEARCH GAP

Although the baseline SKY130 generator, OpenRAM, FastMem, and commercial compilers each address some part of custom SRAM sizing, none of them couples an automated search to a validated area model, a principled multi-metric selection rule, or a comparison across selection methods to establish whether method choice changes anything. Area evaluation, where it exists at all, relies on closed-form estimates rather than the real placement code that would actually be used to build the memory. No existing SKY130-targeted tool offers a constraint-aware mode that returns the best configuration fitting inside a fixed die-area budget. Therefore, there is a need for a design-space exploration layer that evaluates every candidate on real, placement-derived metrics, selects among them using a defensible and benchmarked rule, and supports both the fixed-requirement and fixed-space design scenarios common in real ASIC development.

V. EXISTING SYSTEMS

The current SKY130 Macro Memory Cell Generator accepts a memory type, word width, address depth, and placement style typed in directly by the user, and produces the interleaved memory's RTL and physical-design files. It offers three predefined base macro sizes — 8x1024, 32x256, and 32x512 — and five placement styles, but performs no evaluation of its own. Memory type and placement style are chosen once, by hand, with zero comparison against any alternative the tool could have produced instead, and nothing computes a suitable configuration starting from a fixed area budget. OpenRAM and FastMem extend flexibility for the same class of designs but either hit practical size ceilings or remain purely theoretical, never checked against a real place-and-route run. Commercial compilers from Synopsys and Dolphin Technology offer comparable interleaving capability but are closed and licensed, and none of these existing systems benchmarks its own selection logic against alternative established decision-making methods. This means the current tooling landscape is not unified around a validated, reproducible, and constraint-aware selection methodology, making more advanced and integrated solutions necessary.

VI. PROPOSED METHODOLOGY

The proposed system extends the baseline SKY130 generator with an automated design-space exploration layer. Given a word width and address depth, the search builds every combination of memory type (T0, T1, T2) and placement style (row, column, grid) where the word width divides evenly into the macro's native data width — up to nine candidates. Each surviving candidate is evaluated on three metrics computed directly rather than modeled, then filtered and selected using a Pareto-optimal rule benchmarked against five established alternatives. An optional constraint-aware mode filters candidates by a maximum die width and height before selection ever runs.

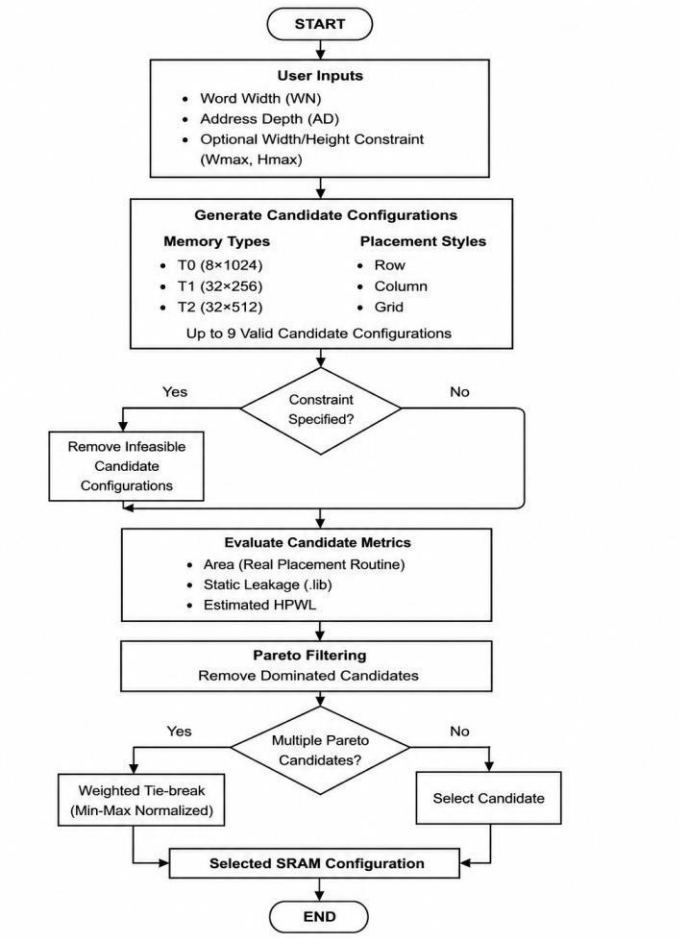

Fig. 1. Proposed design-space exploration framework for the SKY130 macro-based SRAM generator.

A. Candidate Generation and Evaluation Metrics

Area is obtained by calling the baseline tool's own placement routine directly, so the value matches exactly what the generator itself would produce, not a separate closed-form formula. Static leakage is read directly from the cell_leakage_power field of each macro's SKY130 .lib file, scaled by total block count. Estimated HPWL is the half-perimeter bounding box over the macro coordinates the placement routine already generates — a standard, low-cost proxy for post-route wirelength, not a measurement of actual routed wire length.

B. Pareto-Optimal Selection with Weighted Tie-Break

A candidate survives the Pareto filter only if no other candidate matches or beats it on all three metrics while beating it on at least one. Where more than one candidate survives, a weighted tie-break decides between them — area, leakage, and estimated HPWL weighted equally at one-third each by default — using min-max normalized scores computed only within the surviving subset, following normalize(x) = (x − min) / (max − min).

C. Multi-Criteria Decision-Making Validation

To confirm Pareto-plus-tie-break is not an arbitrary choice, five other established methods run against the identical candidate data: weighted-sum, TOPSIS, rank-sum (Borda count), lexicographic ordering, and naive minimum-area. Weighted-sum and TOPSIS normalize over the full nine-candidate field; rank-sum assigns per-metric ranks via a stable sort. All six methods run across fifteen realistic (word width, address depth) sizes spanning 4 KB to 1 MB.

D. Constraint-Aware Optimization

For the fixed-space ASIC scenario, two optional arguments — a maximum width and height, in micrometers — filter out any candidate whose footprint overflows them before the Pareto filter or tie-break ever run. What comes out is the best configuration that actually fits, not the unconstrained global optimum.

VII. ADVANTAGES OF PROPOSED SYSTEM

The proposed system offers several advantages over the baseline generator and comparable tools. Area, leakage, and wirelength are all grounded in real data — placement code, .lib files, and placement coordinates — rather than closed-form estimates that can misjudge the true cost of a configuration. Candidate selection is no longer a bare, unjustified minimum-area rule but a Pareto-optimal filter whose defensibility is checked against five independent established methods on the same fifteen-size dataset. The constraint-aware mode directly supports the common real-world scenario of a fixed silicon budget, something none of the surveyed existing systems provides. Because every extension builds on the baseline tool's own placement and library files rather than external estimation, the results remain reproducible and open-source, with the complete methodology and benchmark data published for independent verification.

VIII. WORKED EXAMPLE

For a 32-bit, 2048-deep target with no constraint applied, the search lands on the global area minimum, which also happens to be the sole Pareto-optimal candidate (MT2/grid, area = 1,636,222 sq. units). Adding a 5000 x 1000 micrometer envelope to the same request causes that previously dominant candidate to be rejected — its height of 1022 micrometers clears the 1000-micrometer ceiling by just 22 micrometers — and the search falls back to the next feasible candidate, MT2/row, which fits within bounds. This is the fixed-space scenario in miniature: a designer with a defined block of silicon gets the best memory that actually fits inside it, not the best memory in the abstract.

IX. COMPARISON OF EXISTING SYSTEM AND PROPOSED SYSTEM

Feature / Parameter	Existing System (Baseline Generator)	Proposed Design-Space Exploration System
Memory Type Selection	Manual, typed in by the user	Automatic search across all valid memory types
Placement Selection	Manual, typed in by the user	Automatic search across row, column, and grid layouts
Area Evaluation	Not performed	Real placement code, not a closed-form estimate
Candidate Evaluation	None	Area, static leakage, and estimated HPWL, jointly
Selection Rule	N/A (manual choice only)	Pareto-optimal filter with weighted tie-break, benchmarked against 5 alternative methods
Fixed-Space (ASIC) Support	None	Constraint-aware feasibility filtering against a width/height budget

X. RESULTS

The proposed system was implemented and evaluated across fifteen realistic (word width, address depth) design sizes, spanning 4 KB to 1 MB, using the baseline generator's own placement and library files.

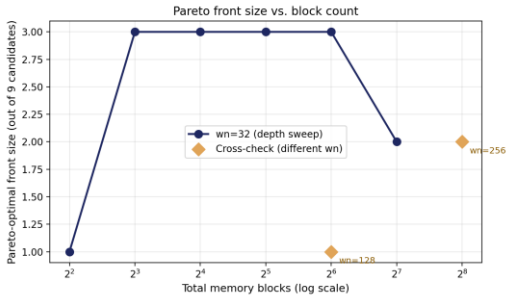

Fig. 2. Pareto-optimal front size versus total block count, word-width-32 sweep with cross-checks at other word widths.

Front size does not climb steadily with block count. It jumps from a single dominant candidate at four blocks to three by eight blocks, holds at three through sixty-four blocks, then drops to two at one hundred twenty-eight. Block count alone does not explain the pattern: two configurations sharing an identical count of sixty-four blocks produced front sizes of three and one, depending only on word width.

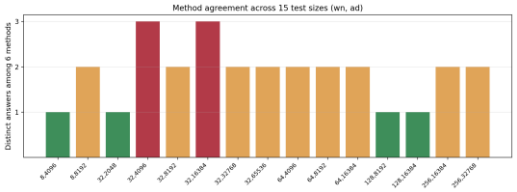

Fig. 3. Number of distinct answers among six selection methods, across fifteen test sizes.

Full agreement across all six selection methods happened on only four of fifteen test sizes (26.7%). Minimum-area, TOPSIS, and lexicographic ordering — three methods built on entirely different logic — returned identical picks in all fifteen cases. Weighted-sum and rank-sum matched in fourteen of fifteen. Pareto-based selection did not consistently side with either cluster, breaking from both twice to pick a third configuration neither group chose. These results confirm that selection-method choice materially changes the recommended SRAM configuration, and that grounding area, leakage, and wirelength in real data — rather than assumption — is necessary at every scale tested.

XI. CONCLUSION

The SKY130 Macro Memory Cell Generator automates interleaving and stops there — no candidate evaluation, no path from a fixed area budget to a suitable configuration. This paper closes both gaps. An automated search scored on area, leakage, and estimated HPWL replaces manual selection, grounded in the baseline tool's own real placement code rather than a closed-form shortcut. A constraint-aware filter returns the best configuration that fits inside a specified physical envelope. Benchmarking confirmed Pareto-plus-tie-break was not an arbitrary pick: six methods, run on the same real data across fifteen sizes, collapsed into two consistent families rather than acting as six independent opinions, with full agreement in barely a quarter of cases. Area estimation and selection method both need checking against real data — neither survives on assumption alone, at any scale tested here.

XII. FUTURE SCOPE

- Validating estimated HPWL against actual post-route wirelength from a full OpenLane run, since it remains an unvalidated proxy carrying equal weight with area and leakage.
- Extending the constraint-aware search to explore across depths automatically, finding the largest memory that fits a given area budget rather than only checking feasibility for one specified depth.
- Carrying selected candidates through a complete RTL-to-GDSII flow to confirm these area, leakage, and wirelength numbers hold up once real routing is accounted for.

REFERENCES

[1] E. I. Baungarten-Leon, S. Ortega-Cisneros, G. Pinedo-Diaz, M. A. Rivera Acosta, F. J. Rodriguez Navarrete, U. Jaramillo-Toral, C. Torres Gonzalez, and J. C. Garcia Lopez, "Macro memory cell generator for SKY130 PDK," IEEE Access, vol. 12, pp. 59688–59701, 2024, doi: 10.1109/ACCESS.2024.3393479.

[2] M. Guthaus et al., "OpenRAM: An open-source memory compiler," Tech. Rep., Dept. Comput. Eng., Univ. California Santa Cruz, 2016.

[3] A. Parmar, K. Prasad, N. Rao, and J. Mekie, "FastMem: A fast architecture-aware memory layout design," in Proc. 23rd Int. Symp. Quality Electron. Design (ISQED), 2022, pp. 120–126.

[4] C.-L. Hwang and K. Yoon, Multiple Attribute Decision Making: Methods and Applications. Berlin, Germany: Springer-Verlag, 1981.

[5] R. L. Keeney and H. Raiffa, Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Cambridge, U.K.: Cambridge Univ. Press, 1993.

[6] A. R. Omondi, The Microarchitecture of Pipelined and Superscalar Computers. Dordrecht, Netherlands: Springer, 1999.

[7] S. Das and S. Dey, "FPGA based design of a fine-grained fault tolerant interleaved memory," in Proc. IEEE Int. Conf. Adv. Commun. Control Comput. Technol., Ramanathapuram, India, 2014, pp. 565–568.

[8] R. Goldman, K. Bartleson, T. Wood, V. Melikyan, and E. Babayan, "Synopsys' educational generic memory compiler," in Proc. 10th Eur. Workshop Microelectron. Educ. (EWME), Tallinn, Estonia, 2014, pp. 89–92.

[9] Dolphin Technology, "Memory products," 2024. [Online]. Available: https://www.dolphin-ic.com/about/market.html

[10] G. Palermo, C. Silvano, and V. Zaccaria, "ReSPIR: A response surface-based Pareto iterative refinement for application-specific design space exploration," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 28, no. 12, pp. 1816–1829, Dec. 2009, doi: 10.1109/TCAD.2009.2028681.

[11] J.-C. de Borda, "Mémoire sur les élections au scrutin," Histoire de l'Académie Royale des Sciences, Paris, France, 1784.

[12] P. C. Fishburn, "Lexicographic orders, utilities and decision rules: A survey," Manage. Sci., vol. 20, no. 11, pp. 1442–1471, 1974.

[13] K. Mela, T. Tiainen, and M. Heinisuo, "Comparative study of multiple criteria decision making methods for building design," Adv. Eng. Informat., vol. 26, no. 4, pp. 716–726, 2012, doi: 10.1016/j.aei.2012.03.001.