DESIGN AUTOMATION IN VLSI – CE 357

MAGNETIC-FIELD-AWARE FLOORPLANNING AND OBSTACLE-DRIVEN ROUTING FOR MRAM/MTJ AND RF-RICH SOCs

FINAL REPORT

CAN AFACAN

1. Introduction

The trajectory of VLSI technology, characterized by relentless miniaturization and the escalating complexity of Systems-on-Chip, continually introduces new paradigms and challenges for physical design automation. A significant trend in modern SoC design is the heterogeneous integration of diverse functionalities, including non-volatile memory technologies like Spin-Transfer-Torque Magnetic Random Access Memory (STT-MRAM) and analog/RF components such as on-chip inductors and transformers. While these elements offer substantial benefits in terms of performance, power efficiency, and form factor, their incorporation introduces physical phenomena previously of secondary concern in purely digital designs. Specifically, magnetically active components can generate appreciable stray magnetic fields. These fields, if not managed, present a critical reliability risk, capable of inducing unintended bit-flips in adjacent memory arrays or causing significant detuning and performance degradation in sensitive analog and RF front-end circuitry.

Conventional physical design methodologies have historically prioritized geometric constraints, such as area minimization and overlap avoidance, with multi-physics considerations like magnetic interference often addressed only in late-stage verification. This reactive approach frequently necessitates the application of overly conservative guard-banding margins around sensitive or emitting components, leading to suboptimal area utilization, or, in more severe cases, requiring costly and time-consuming design iterations if violations are discovered post-layout. The inherent inefficiency and potential for compromised performance underscore the need for a paradigm shift towards proactive management of such cross-domain interactions.

This project endeavors to address these emergent challenges by developing a comprehensive physical design automation framework that intrinsically incorporates magnetic-field awareness from the nascent stages of floorplanning through to detailed routing. The central tenet of this work is the elevation of magnetic integrity to a primary design objective, on par with traditional geometric and connectivity requirements. By treating magnetic keep-out zones as fundamental, first-class constraints, the system aims to ensure chip reliability and performance "by design," rather than through subsequent, often palliative, interventions. The developed tool leverages a simulated annealing optimization engine for placement, integrates custom obstacle-aware routing algorithms cognizant of these magnetic exclusion zones, and, introduces a three-dimensional visualization technique to map E-field intensity, proxied by local wire density, across the chip. This E-field map, augmented with representations of MTJ component locations, provides designers with novel insights into potential signal integrity concerns, regions of high routing congestion, or electromagnetic interference (EMI) hotspots. This report provides a detailed account of the project's current status, elucidates the architecture of the developed system, discusses its innovative contributions in the context of the course objectives and contemporary research directions in VLSI CAD, and outlines potential avenues for future research and enhancement.

All algorithms and code discussed in this document are available in a GitHub repository, the link to which is provided in Section 9.

2. Detailed Project Description and Current System Architecture

The project has matured into a tangible software prototype with a rich set of functionalities. This system provides an end-to-end, albeit simplified, workflow from block definition and MTJ parameterization to optimized floorplanning, constrained routing, and post-routing analysis. The software is implemented entirely in Python, utilizing the Tkinter library for its graphical user interface (GUI), Matplotlib for 2D and 3D data visualization, and NumPy for numerical operations for geometric and physics-based calculations. The architecture is modular, comprising several distinct but interconnected components:

2.1. Graphical User Interface (GUI) and System Control (main.py)

The GUI serves as the primary interaction point for the designer, offering a visual representation of the chip layout and controls for various design operations. A central Matplotlib canvas dynamically renders the floorplan, displaying standard logic blocks as shaded rectangles and designated Magnetic Tunnel Junction (MTJ) blocks as distinct circles. A key visual feature is the depiction of "danger zones" around MTJ blocks (after the user inputs material thickness and saturation magnetization characteristics), semi-transparent red circular regions indicating the calculated magnetic field keep-out radii that other components must respect.

User interaction capabilities are extensive. Designers can instantiate new blocks, specifying their dimensions, or remove existing ones. Blocks can be resized dynamically, and their positions can be manually adjusted via a drag-and-drop interface, facilitating coarse initial placements or fine-tuning of automated results. For connectivity, users can define and attach an arbitrary number of color-coded pins to the edges of each block, specifying their side (top, bottom, left, right) and fractional offset along that edge. A dedicated dialog allows for the detailed parameterization of MTJ blocks. Users can input the layer stack information, specifically, the thickness and saturation magnetization (Ms) for the free, barrier, reference, fixed, and antiferromagnetic layers, which are critical inputs for the subsequent magnetic field calculations. The GUI also provides controls to select initial placement strategies (e.g., "rectangular" grid or "wheel" arrangement), choose routing styles (Manhattan, Steiner, or Euclidean), and to initiate the automated floorplanning (simulated annealing) and routing processes. Finally, a distinct control enables the generation of the 3D E-field visualization map after routing is completed.

2.2. MTJ Device Modeling and Field Zone Calculation (mtj_calc.py)[2, 3]

This module encapsulates the physics-based modeling of MTJ devices. The MTJDevice class is central to this, representing an MTJ stack and its constituent layers. Its primary function is to calculate the field_zone_radius for each MTJ component. This calculation is based on a magnetic dipole field model, where the magnetic moment of the MTJ is first determined from the sum of the products of saturation magnetization, volume (area × thickness) for each ferromagnetic layer. For a single ferromagnetic layer, the formula is used such that:

$$m_i = M_{s,i} \cdot V_i = M_{s,i} \cdot A \cdot t_i$$

where:

- m_i is the magnetic moment of layer i.
- $M_{s,i}$ is the saturation magnetization of layer i.
- V_i is the volume of layer i.

- A is the cross-sectional area of the MTJ.
- t_i is the thickness of layer i.

Then the total magnetic moment for the MTJ stack, considering only ferromagnetic layers, is:

$$M_{total} = \sum_{i \in \text{ferro}} m_i = A \sum_{i \in \text{ferro}} \left(M_{s,i} \cdot t_i \right)$$

Given this moment and a user-specified magnetic field strength threshold (B_thresh), representing the maximum tolerable stray field for sensitive neighboring circuits, the module computes the radius at which the MTJ's perpendicular stray field component attenuates to this threshold. Based on a simplified dipole model, the perpendicular magnetic field component (B_z) at a radial distance r in the plane of the MTJ can be approximated. The formula used to find the radius r_m where the field drops to B_{thresh} is:

$$B_z(r) \approx \frac{\mu_0 M_{total}}{2\pi r^3}$$

Setting $B_z(r_m) = B_{thresh}$, we solve for the keep-out radius r_m:

$$r_m = \left(\frac{\mu_0 M_{total}}{2\pi B_{thresh}} \right)^{1/3}$$

where:

- r_m is the calculated magnetic field zone radius before margin.
- μ_0 is the vacuum permeability ($4\pi \times 10^{-7}$ T·m/A).
- M_{total} is the total magnetic moment of the MTJ.
- B_{thresh} is the user-specified magnetic field threshold.

The final keep-out zone radius $R_{keepout}$ is then:

$$R_{keepout} = r_m + \text{MTJ\_MARGIN}$$

This radius, augmented by a configurable MTJ_MARGIN for additional safety, defines the magnetic keep-out zone. This calculation is dynamic; any changes to an MTJ's layer parameters or its allocated floorplan area (which influences its magnetic volume if thickness is fixed) will trigger a re-computation of its keep-out radius, which is then immediately reflected in the GUI.

2.3. MTJ-Aware Floorplanning Engine (floorplanning.py)

The floorplanning engine is responsible for determining the optimal placement of all blocks on the chip canvas, subject to a variety of constraints and objectives. It is built around the simulated annealing (SA) metaheuristic, a probabilistic technique well-suited for large-scale combinatorial optimization problems.

The Block class serves as the fundamental data structure for representing individual components, storing their unique identifiers, dimensions, and mutable (x, y) coordinates for their top-left corner. The SA algorithm iteratively perturbs the current floorplan by applying one of two types of moves: randomly

swapping the positions of two selected blocks or applying a small random displacement ("jitter") to a single selected block. The magnitude of this jitter is scaled by the current annealing temperature.[5]

An MTJ-aware cost function (make_cost_fn) guides the SA process. This function evaluates the quality of a given floorplan configuration and incorporates several terms. The general form of the cost function C can be expressed as:

$$C(\text{state}) = C_{hard}(\text{state}) + C_{soft}(\text{state})$$

1. **Hard Constraints:** These are conditions that must ideally be met, and their violation incurs a substantial PENALTY in the cost, effectively making such configurations highly undesirable.
    - *Overlap Prevention:* No two blocks (MTJ or standard) may overlap, considering a dynamically adjustable gap_target that dictates the minimum required spacing.
    - *MTJ Keep-Out Enforcement:* No standard logic block may intrude into the calculated magnetic keep-out zone of any MTJ block. MTJ blocks themselves are permitted to be close to each other, as their self-interference or mutual interaction is not yet modeled.
2. **Soft Objectives:** These are qualities that the annealer attempts to optimize by minimizing their contribution to the cost.
    - *Area Minimization:* The total bounding box area enclosing all placed blocks is minimized to promote a compact layout. The bounding box area is calculated as:

      $$A_{BB} = (\max(x_i + w_i) - \min(x_i)) \cdot (\max(y_i + h_i) - \min(y_i))$$

      where:
        - (*xi*,*yi*) are the top-left coordinates of block *i*.
        - *wi*, *hi* are the width and height of block *i*.
        - The min/max operations are taken over all blocks in the layout.
    - *Slack Reduction:* The "slack," defined as the sum of squared deviations from the gap_target for inter-block spacing, is minimized to encourage uniform block distribution without excessive empty space. For any two blocks I and j, given d is their center to center distance, the edge-to-edge separations are:

      $$s_{x,ij} = \max\left(0, d_{x,ij} - \left(w_i + w_j\right)/2\right)$$

      $$s_{y,ij} = \max\left(0, d_{y,ij} - \left(h_i + h_j\right)/2\right)$$

      The Euclidean gap g_{ij} between them is:

      $$g_{ij} = \sqrt{s_{x,ij}^2 + s_{y,ij}^2}$$

The extra gap beyond the target is $e_ij = max(0, g_ij - gap\_target)$

That makes the total slack:

$$S = \sum_{i<j} e_{ij}^2$$

The soft component previously defined as $C_{soft}(\text{state})$ is then:

$$C_{soft}(\text{state}) = A_{BB} + w_{slack} \cdot S$$

A crucial aspect of SA implementation is its handling of hard constraints. Any proposed move that results in a violation of the overlap or MTJ keep-out rules is rejected immediately by the core SA logic, prior to a full evaluation of the soft objectives in the cost function. This early rejection strategy ensures that all configurations accepted and explored by the annealer are, by construction, valid with respect to these critical constraints. The cost function itself performs a redundant check of these hard constraints as a "belt-and-braces" measure.

The SA process also features an adaptive gap mechanism. It begins with a relatively generous INIT_GAP to allow for greater freedom in block movement during the high-temperature exploration phase. As the temperature cools and the annealer enters a more exploitative phase (specifically, when T < 1.0), the gap_target within the cost function is tightened to a smaller FINAL_GAP. This staged approach facilitates a transition from coarse global placement to fine-grained compaction.

To initiate the annealing, the system can employ one of two seed layout strategies: a "rectangular" grid-based arrangement or a "wheel" layout where blocks are placed circumferentially. When the wheel layout is selected, an additional penalty term is incorporated into the cost function to discourage significant deviations from the ideal circular arrangement, thereby attempting to preserve the wheel topology during optimization.

The wheel misalignment error (E_{wheel}) for a block (k) at center (c_x, c_y) relative to its ideal target position (t_x, t_y) on the wheel (with radius (R_{wheel}) and angle (θ_k)) is:

$$t_{x,k} = X_{center} + R_{wheel} \cos(\theta_k)$$

$$t_{y,k} = Y_{center} + R_{wheel} \sin(\theta_k)$$

$$E_{wheel} = \frac{1}{N_{blocks}} \sum_{k=1}^{N_{blocks}} \left(\left(c_{x,k} - t_{x,k}\right)^2 + \left(c_{y,k} - t_{y,k}\right)^2 \right)$$

For the wheel layout, the cost function becomes:

$$C_{wheel}(\text{state}) = C_{hard}(\text{state}) + A_{BB} + w_{slack} \cdot S + w_{wheel} \cdot E_{wheel}$$

where w_{wheel} is a weighting factor for the wheel penalty.

2.4. Obstacle-Driven Routing Engine (wiring.py)

Once floorplanning is complete, the routing engine connects the specified pins on different blocks. The router is designed to be MTJ-aware, treating the keep-out zones of MTJs as obstacles. However, a crucial distinction is made: a wire segment connecting to a pin on an MTJ block is permitted to traverse that specific MTJ's own keep-out zone. It must, however, avoid the keep-out zones of all *other* MTJ blocks present on the chip. This nuanced approach balances connectivity requirements with magnetic interference mitigation. All keep-out zones are inflated by a small CLEARANCE value to ensure routed wires maintain a safe distance. So, the effective radius of an obstacle (MTJ keep-out zone) for routing purposes is defined as:

$$R'_{obs} = R_{keepout} + \text{CLEARANCE}$$

Three distinct routing algorithms are provided:

1. **Manhattan Routing (manhattan_route):** This algorithm prioritizes rectilinear paths. It first attempts simple L-shaped (one-bend) routes between a source and target pin. If this direct L-route is obstructed by an MTJ keep-out zone, the algorithm iteratively explores Z-shaped (two-bend) "staircase" routes by incrementally offsetting one of the segments until a clear path is found or search limits are exceeded.

2. **Euclidean Routing (euclidean_route):** This algorithm initially attempts a direct straight-line connection. If this line intersects an MTJ obstacle, the router identifies the first blocking circle and attempts to find a "via" point by moving tangentially away from the circle's center, effectively creating a two-segment path around the obstacle. This tangential offset is also increased iteratively.

3. **Steiner Routing (steiner_route)**:
   This mode builds an approximate rectilinear Steiner tree that can connect three-or-more pins in one shot instead of routing them pair-by-pair. It works in two phases:
   1. **Vertical trunk selection**: it takes the median *x*-coordinate of all pins and places a tentative vertical “trunk” there.
   2. **Branch routing**: each pin is then connected to that trunk with the same obstacle-aware Manhattan router used elsewhere, so every branch is individually clearance-checked against MTJ keep-out circles and (if enabled) block rectangles.

   Once every pin has reached the trunk, the router stitches a single vertical segment between the highest and lowest branch attachment points, giving one continuous tree that minimizes total wire length while still honoring all obstacles.

Route Around the Blocks Checkbox:

When this option is ticked, all normal (grey) blocks, except the ones that actually own the pins being routed, are inserted into the obstacle list as inflated rectangles. Consequently, no segment from any routing mode can cross**;** only the surrounding whitespace is legal routing area. Untick it and the algorithms ignore block rectangles, considering only the red MTJ danger discs, which can yield shorter but block-spanning paths.

Nets are implicitly defined by connecting pins that share the same index (different indexes are shown as different colors) across different blocks. For instance, pin 0 on block A will be connected to pin 0 on block B, and then to pin 0 on block C, forming a multi-terminal net. The routing is performed sequentially for

these pin-pairs. The routing process can be animated, with each segment drawn incrementally on the GUI, providing a visual understanding of how the algorithms navigate the obstacle field.

2.5. E-Field (Wire Density) Visualization Module (main.py)

A distinctive feature of this project is the post-routing E-field visualization tool, designed to offer novel insights into the electrical characteristics of the layout. This module is invoked by the user after the routing phase is complete.

The process begins by discretizing the chip canvas area into a uniform grid. For each cell in this grid, a "wire density" metric is computed. This metric is defined as the sum of the geometric lengths of all individual wire segments whose midpoints fall within the boundaries of that particular grid cell. This aggregated length serves as a proxy for local routing congestion and, by extension, can be interpreted as an indicator of areas with potentially higher E-field intensity due to increased capacitive coupling or current flow.

To enhance visual clarity and represent a more continuous field, this raw wire density map can optionally undergo Gaussian smoothing, provided the SciPy library is available in the user's Python environment. The smoothed (or raw) density data is then rendered as a 3D surface plot using Matplotlib's 3D capabilities. In this plot, the X and Y axes correspond to the chip's floorplan coordinates, while the Z-axis represents the calculated wire density, color-coded to indicate varying intensities (e.g., hotter colors for higher density).

To contextualize this "E-field" landscape, the locations of MTJ blocks are explicitly represented within the 3D visualization. Each MTJ block is depicted as a semi-transparent cylindrical column, rising from the base of the plot (Z=min_density) to its top (Z=max_density) at the MTJ's (X,Y) floorplan coordinates. This overlay allows designers to visually inspect the proximity of MTJs to regions of high wire density, which could be critical for assessing potential EMI coupling or understanding the impact of MTJ placement on overall routing patterns and electrical stress. The 3D plot is fully interactive, allowing rotation and zooming for detailed examination.

2.6. Current Development Status and Iterative Refinements

As of this report, the integrated system has achieved a high degree of functionality. All core modules, GUI, MTJ modeling, floorplanning, routing, and E-field visualization, are implemented and operational. Users can successfully define chip layouts with standard and MTJ blocks, specify detailed MTJ parameters, execute the simulated annealing floorplanner (for both rectangular and wheel initializations), perform MTJ-aware routing, and subsequently generate the 3D wire density visualization.

The development process has been iterative, with significant effort dedicated to ensuring the robustness and correctness of the interactions between modules. For instance, the initialization sequence of attributes within the main FloorplanApp class was carefully refined to prevent errors arising from methods being called before their required data structures (e.g., pin_specs, mtj_radii, Matplotlib axes ax) were properly instantiated. The simulated annealing algorithm, particularly its cost function and the handling of hard constraints, underwent several revisions to ensure consistent behavior, especially during the adaptive gap tightening phase and when dealing with the compound cost function used for the wheel layout (which includes an additional penalty for deviation from the wheel topology). The E-field visualization module was a later addition, built upon the completed routing infrastructure.

3. Project Novelty and Alignment with Academic Context

This project distinguishes itself through several innovative aspects and aligns commendably with the objectives outlined in the CE357 course project guidelines, aiming to extend classroom learning towards practical application and early research exploration.

3.1. Intrinsic Magnetic-Field Awareness in Physical Design

The primary innovation of this work lies in its proactive and integrated approach to managing magnetic interference in VLSI physical design. Traditional CAD flows often defer the consideration of such multi-physics effects, treating them as late-stage verification concerns. This project, conversely, elevates magnetic integrity to a foundational requirement. This is achieved through two principal mechanisms:

1. **Physics-Informed Constraint Generation:** The system does not rely on arbitrary or overly conservative fixed margins for magnetic components. Instead, it dynamically calculates magnetic keep-out zones based on the physical layer stack parameters of each MTJ device and a defined stray field threshold. This allows for more tailored and potentially less pessimistic exclusion regions.[2]

2. **Native Integration into Optimization and Routing:** These calculated keep-out zones are not only visual aids; they are incorporated as hard constraints within the cost function of the simulated annealing floorplanner. Consequently, the placement engine actively seeks solutions that inherently respect these magnetic boundaries. Similarly, the routing algorithms are designed to treat these zones as inviolable obstacles (with the exception of nets terminating on the MTJ itself), ensuring that interconnect paths are magnetically compliant by construction.
This holistic integration of magnetic awareness from the outset represents a significant departure from conventional practices and addresses a growing concern in the design of SoCs featuring MRAM, RF, or other magnetically active elements.

3.2. Three-Dimensional E-Field (Wire Density) Visualization

The second contribution is the development of the 3D E-field visualization tool. While the concept of wire density is established in VLSI CAD as an indicator of routing congestion, its representation as a continuous 3D surface, analogous to an "E-field intensity map," offers a uniquely intuitive and powerful analytical perspective. This visualization provides:

- **Enhanced Congestion Analysis:** It moves beyond simple 2D heatmaps of congestion by providing a volumetric sense of wire density, where peaks clearly indicate critical hotspots.

- **Correlation with Component Placement:** Superimposing the MTJ block locations as cylindrical columns within this 3D E-field landscape allows designers to immediately assess the relationship between the placement of these specialized components and the surrounding wiring density. This could reveal, for instance, if MTJs are inadvertently placed in or contributing to regions of extreme routing pressure, or if dense signal routing is occurring undesirably close to sensitive MTJs.[4]

- **Proxy for Electrical Stress and EMI:** High wire density can imply increased capacitive coupling between adjacent wires, greater dynamic power consumption in highly active regions, and a higher potential for electromagnetic interference. The E-field map serves as a visual proxy for these

electrical stress factors, guiding designers towards potential areas of concern for signal integrity or power integrity.

3.3. Alignment with Suggested Project Categories (from Assignment Guidelines)

The developed system resonates with several of the project categories suggested in the course guidelines, demonstrating its relevance to contemporary research challenges in VLSI CAD:[1]

- **Category 2: "New floorplanning approach"**: The project introduces a specialized floorplanning methodology that is fundamentally MTJ-aware. The core SA algorithm is adapted with a cost function and constraint handling mechanisms specifically designed to manage magnetic keep-out requirements. The philosophy, as outlined in the initial proposal, of prioritizing "interconnection" (in this case, magnetic non-interference) alongside area packing aligns with the spirit of developing novel placement strategies.
- **Category 4: "Routability-driven placement"**: Although the current implementation does not employ a closed-loop system where routability metrics directly influence placement decisions during the annealing process, the E-field (wire density) map serves as a powerful post-hoc analysis tool for routability and congestion. The visual feedback provided by this map highlights regions where routing resources are heavily utilized or where congestion is problematic. This forms a critical first step towards a truly routability-driven placement system, as the insights gained could be used to formulate new cost terms or constraints for subsequent optimization runs. Furthermore, the inherent MTJ-awareness in both floorplanning and routing contributes to overall routability by ensuring paths are not blocked by unforeseen magnetic constraints.
- **Category 8: "Combining floorplanning with placement techniques..."**: While the project does not tackle the mixed-mode placement problem in the traditional sense of handling standard cells alongside large macros, it does manage two distinct classes of components: standard logic blocks (represented as rectangles) and MTJ blocks (represented as circles with unique physical properties and associated field constraints). The floorplanning engine (SA) is tasked with optimally arranging these heterogeneous elements, considering their specific geometric and non-geometric (magnetic) interactions. This requires a combination of techniques typically found in both macro-cell floorplanning (managing large, potentially irregularly shaped keep-out zones) and detailed placement (achieving a compact arrangement).

4. Implementation Details, Challenges Encountered, and Solutions

From concept to a functional prototype involved overcoming several technical challenges and making key implementation decisions:

- **Simulated Annealing Parameterization and Constraint Management:** A significant portion of the development effort was dedicated to the robust implementation of the simulated annealing algorithm. The choice of an appropriate cooling schedule (initial temperature, final temperature, cooling rate α), the number of moves attempted at each temperature, and the design of the move set (block swaps and jitters) required iterative tuning. A primary challenge was the strict enforcement of hard constraints (no block overlaps, no MTJ keep-out violations). The adopted solution involves an "early rejection" mechanism within the SA core: any proposed move that

violates these fundamental rules is immediately undone and discarded *before* a full cost function evaluation. This significantly improves efficiency by avoiding costly evaluations of invalid states and ensures that the annealer only explores the valid region of the solution space. The cost function itself also includes checks for these hard constraints, providing a redundant safety layer. The adaptive gap tightening mechanism, transitioning from INIT_GAP to FINAL_GAP, required careful handling of the cost function re-parameterization, particularly for the wheel layout where the cost function includes an additional term for topological adherence. Ensuring that this compound cost function was correctly reconstructed and that its constituent parts (base cost and wheel penalty) were appropriately managed during the gap transition was a non-trivial debugging exercise.

- **Geometric Primitives and Collision Detection:** The system relies heavily on accurate and computationally efficient geometric operations. Functions for calculating block bounding boxes, detecting overlaps between rectangles (considering a specified gap), and detecting overlaps between rectangles and circles (for MTJ keep-out zones) form the bedrock of the constraint checking system. These were implemented with careful attention to edge cases and computational cost.
- **MTJ-Aware Routing Logic:** The development of the Manhattan, Euclidean, and Steiner-Tree routing algorithms necessitated careful logic to correctly interpret MTJ keep-out zones as obstacles. The critical nuance was allowing a net segment to enter the keep-out zone of an MTJ if that MTJ was the source or destination of that specific segment, while simultaneously prohibiting entry into the zones of all other MTJs. This required dynamically tailoring the list of active obstacles for each segment being routed. The iterative nature of the obstacle avoidance in both routers (e.g., expanding L-bends to Z-bends in Manhattan, or increasing tangential offsets in Euclidean) also required careful management of search limits to prevent excessive runtimes for complex scenarios.
- **Graphical User Interface Responsiveness and Visualization Fidelity:** Creating a responsive GUI with real-time updates during potentially long-running processes like animated SA or routing presented challenges. Effective use of Tkinter's event loop and Matplotlib's canvas drawing mechanisms (master.update() versus master.update_idletasks(), canvas.draw_idle()) was essential to prevent the interface from freezing while still providing timely visual feedback. For the 3D E-field map, achieving a clear, understandable, and aesthetically pleasing visualization involved experimentation with Matplotlib's 3D surface plotting parameters, colormaps, lighting, transparency (for MTJ columns), and viewing angles.

5. Preliminary Results and System Demonstration

The implemented system provides compelling visual and functional demonstrations of its capabilities. Upon launch, the user is presented with an initial random (but valid, non-overlapping) placement of blocks, with MTJ components and their calculated keep-out zones clearly distinguished.

- **Interactive Design and Parameterization:** Figures 1 and 2 demonstrate the ease with which users can add, resize, or delete blocks, modify pin configurations, and input detailed layer parameters for MTJ devices through dedicated dialogs. The dynamic update of MTJ keep-out radii in response to parameter changes are shown in Figure 3.

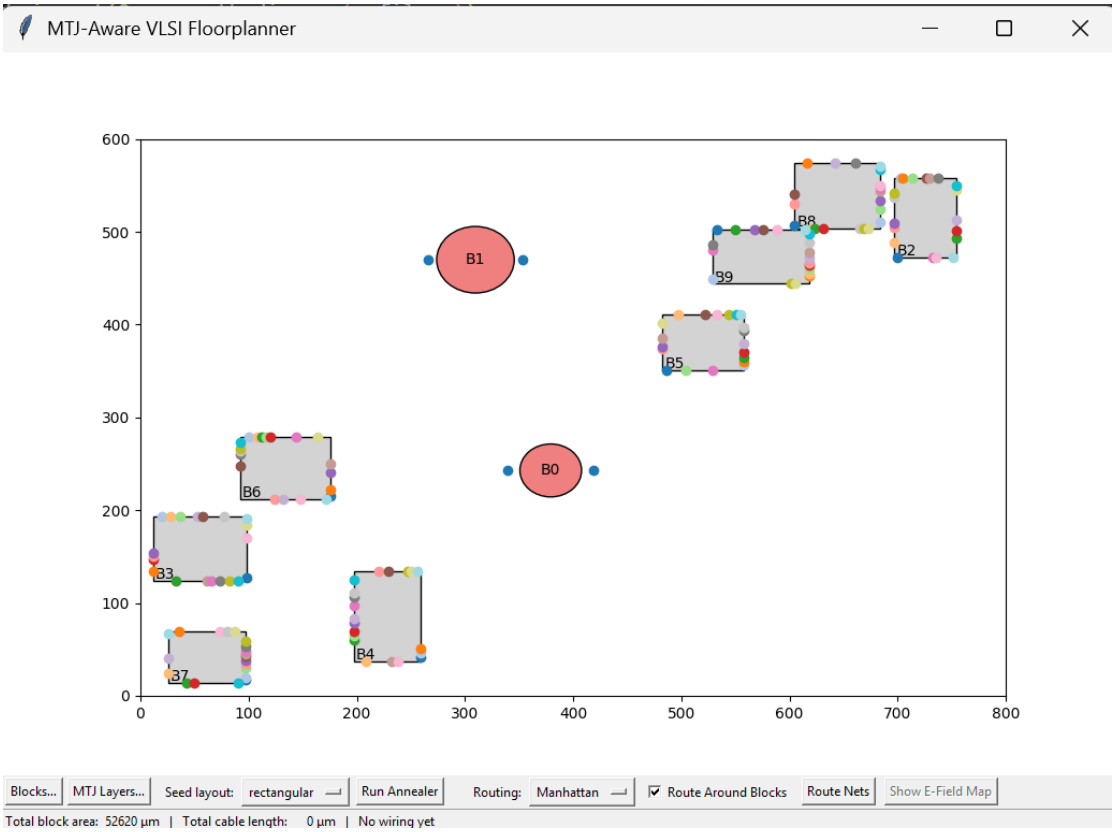

Figure 1: *Initial Menu with Random Blocks Created as an Example*

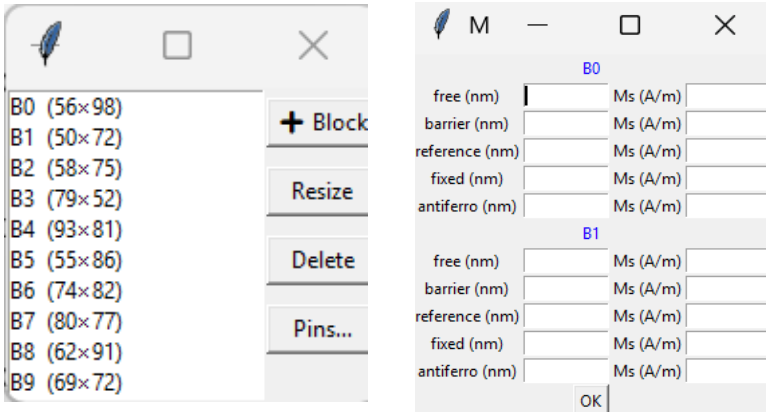

Figure 2: *Blocks and MTJ Layers Menu, respectively*

- **Simulated Annealing in Action:** A sequence of screenshots in Figure 3 showcases the simulated annealing process. Starting from a chosen seed layout (e.g., rectangular or wheel), the pictures depict blocks gradually moving, exploring different configurations, and eventually converging towards a more compact and optimized floorplan that respects all geometric and magnetic constraints. The visual difference between the initial and final annealed states are emphasized.

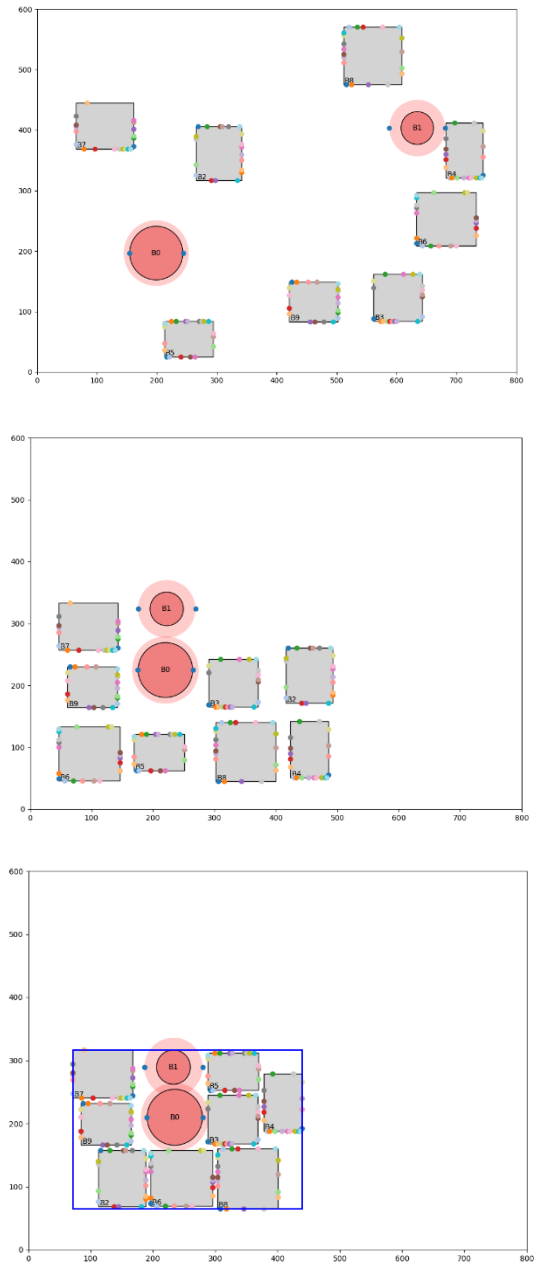

Figure 3: *Initial, Middle, and Final Steps of Rectangular Simulated Annealing Process with Predefined Characteristics to Create Visible Stray Fields around MTJs*

- **MTJ-Aware Routing:** Figure 4 illustrates the outcome of the routing phase for Manhattan routing algorithm. This image shows nets connecting pins across different blocks, with a focus on how the routes navigate around the keep-out zones of uninvolved MTJs while being permitted to enter the zone of a connected MTJ. Different colors for different nets enhance clarity.

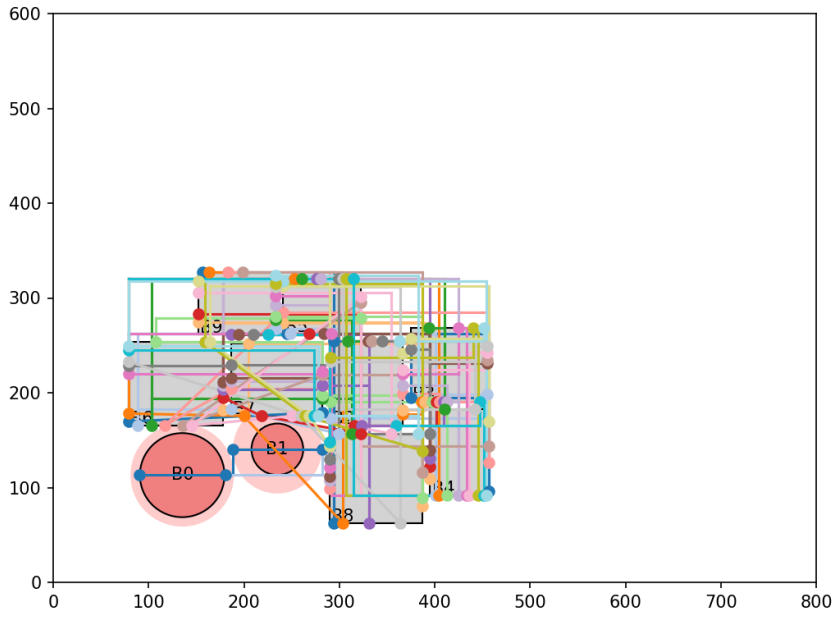

Figure 4: *A View from a Layout after Manhattan Style Routing Algorithm is Conducted*

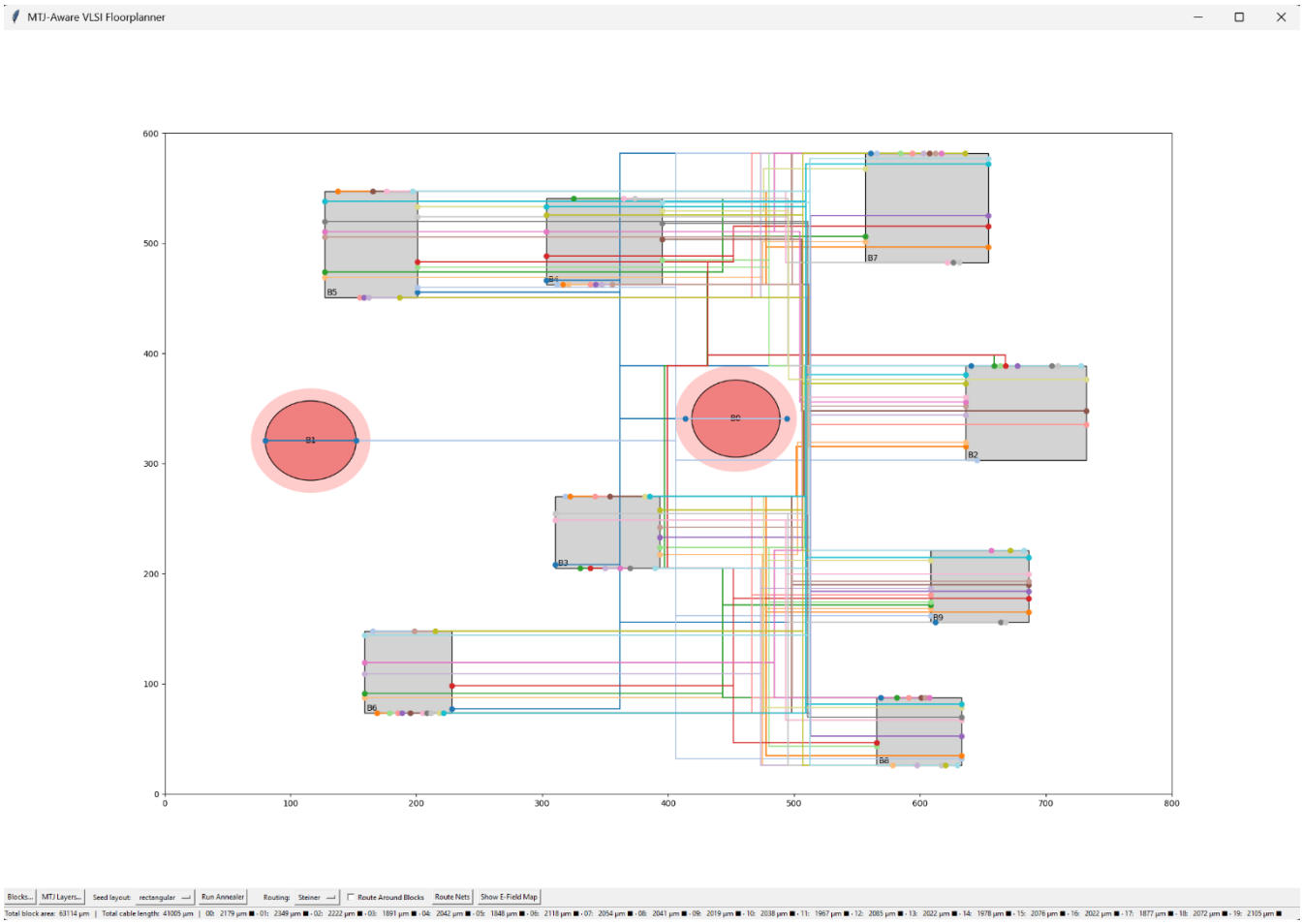

Figure 5: *A View from a Layout after Steiner Tree Style Routing Algorithm is Conducted (without floorplanning (SA) beforehand)*

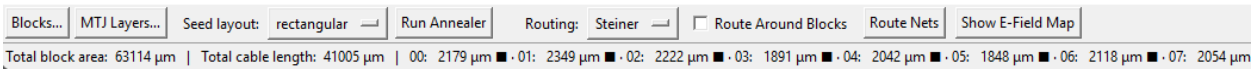

Figure 6: *Bottom Left of Figure 5*

Figure 5 illustrates the Steiner Tree style routing with the interface indicating the length of each routing segment below – shown in detail in Figure 6.

- **3D E-Field (Wire Density) Visualization:** Figure 7 shows the interactive 3D plot that displays the wire density surface, with color gradients indicating regions of low and high density. The semi-transparent cylindrical columns represent MTJ locations on the actual layout, allowing for an immediate visual correlation between MTJ placement and local wiring patterns. Different viewing angles of this 3D map is shown from 2 different angles.

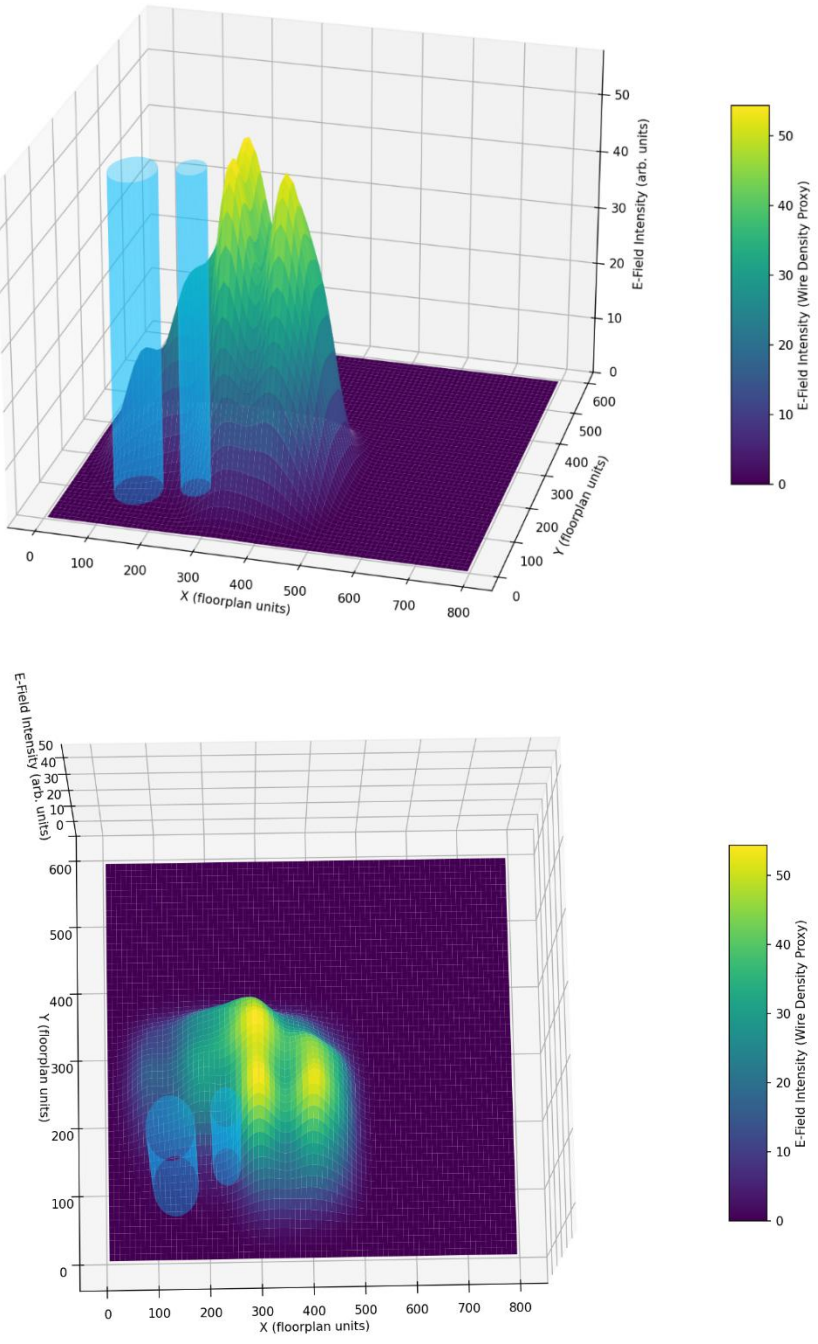

Figure 7: *E-field Density Map Using the Routing given in Figure 4*

The performance of the simulated annealer has been observed to be sensitive to initial parameters such as INIT_GAP and the spacing/margin values used in the seed layouts. More generous initial spacing generally allows the annealer to find valid, low-penalty starting points more easily, leading to more effective exploration of the solution space. The E-field map has proven effective in qualitatively identifying areas that become heavily congested after routing, particularly when layouts are very compact or when MTJ keep-out zones force detours for many nets.

6. A Cumulative Example

Below, in Figure 8, 9, and 10 a cumulative example on how the system and its main functions work has been shown. We see a random initialization of the system in Figure 8. In Figure 9, we see what happens when wheel floorplanner is running. In Figure 10, we observe what happens when Figure 9's E-Field density map is calculated.

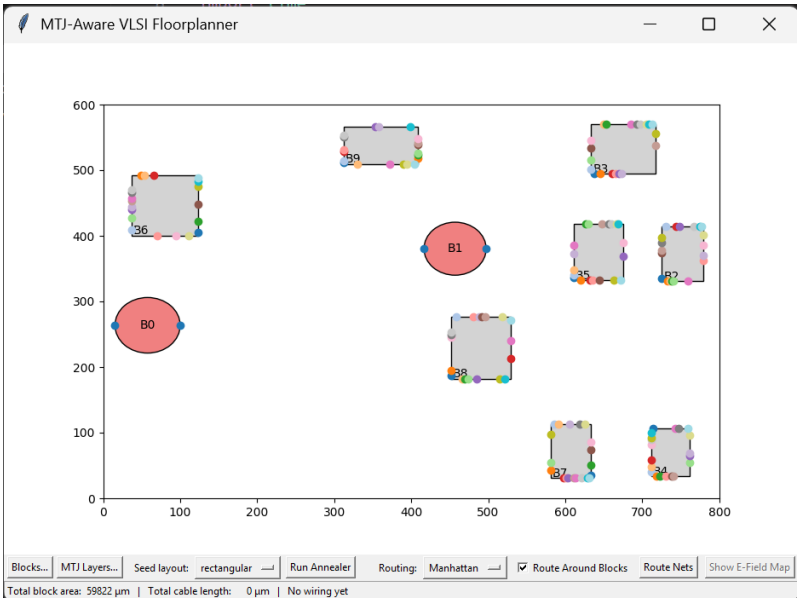

Figure 8: *Initial State of the System*

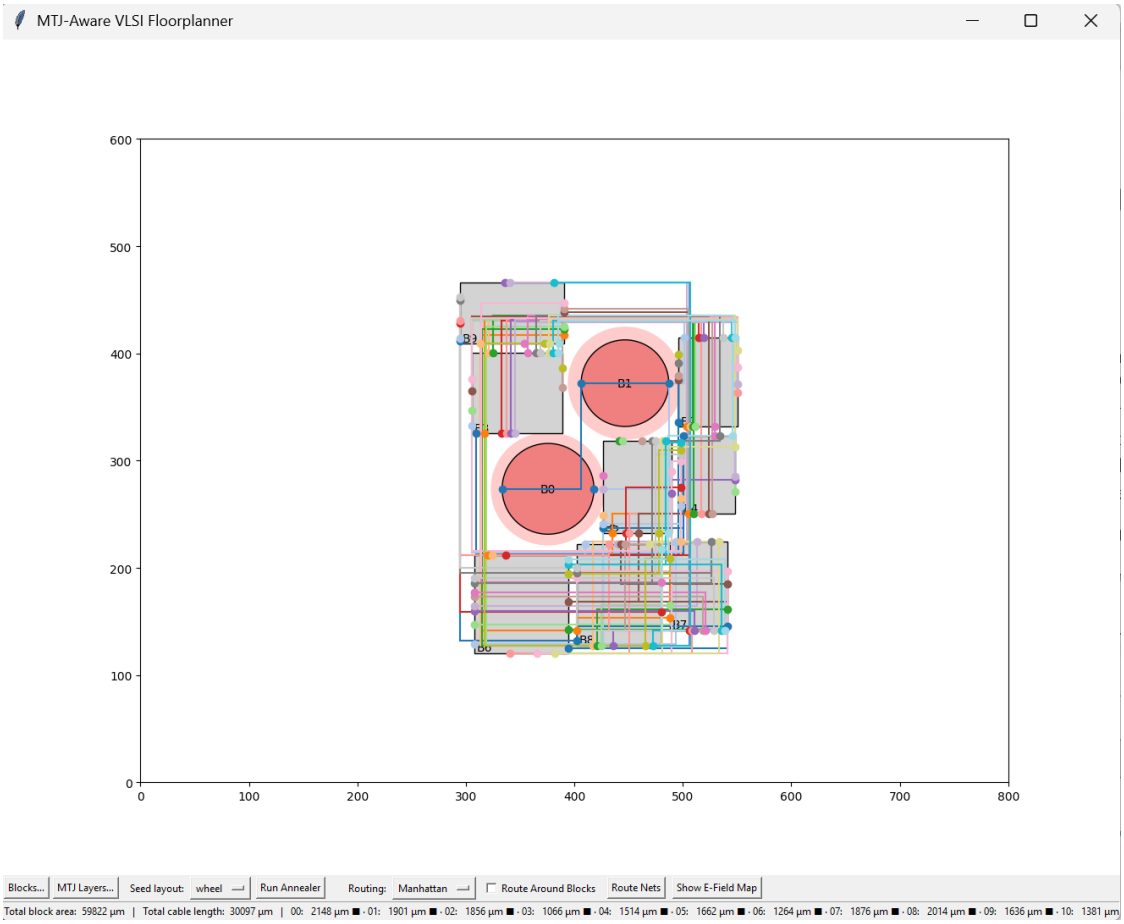

Figure 9: *After Wheel Annealer Floorplanning and Manhattan Style Routing*

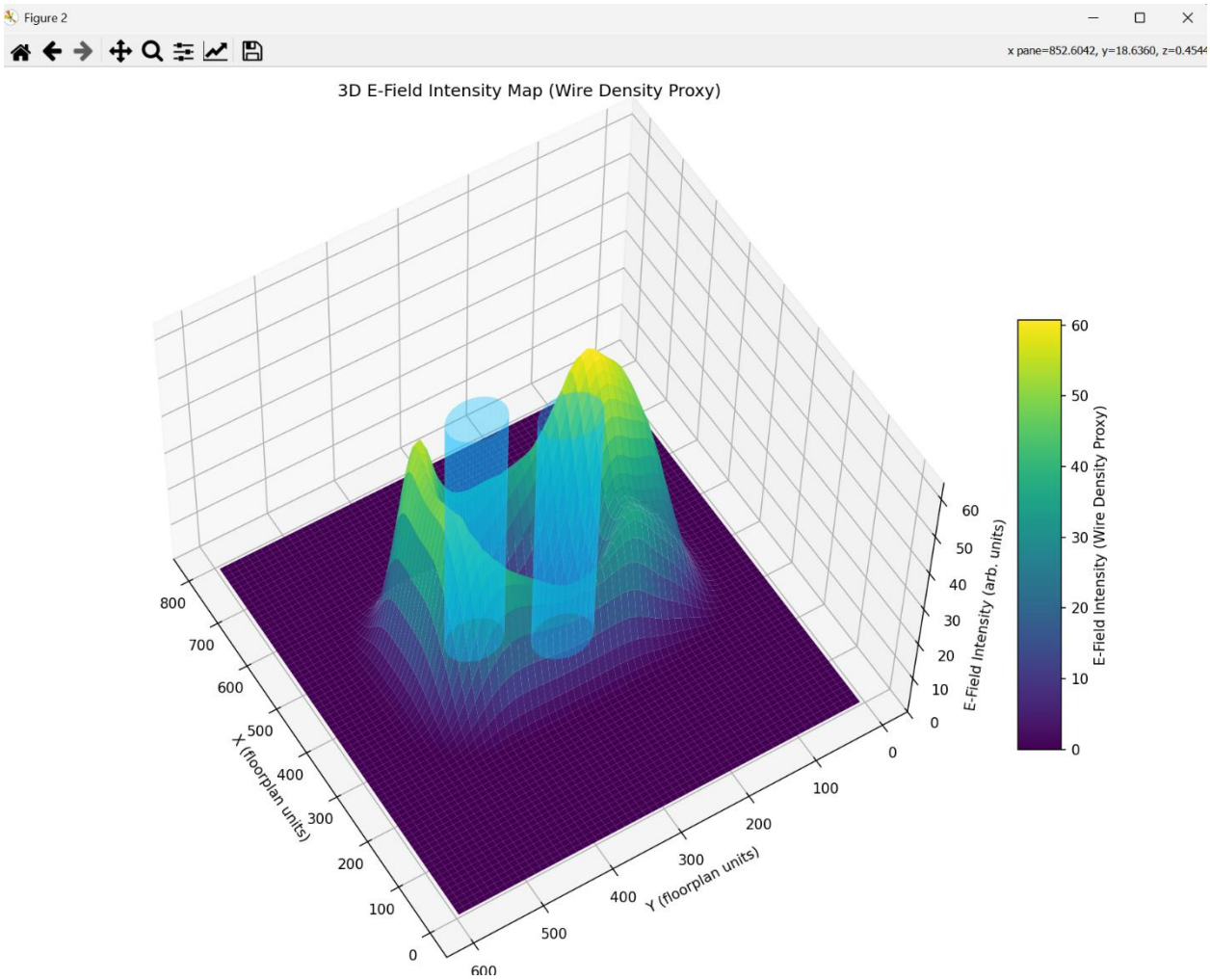

Figure 10: *E-Field Density Map of Figure 9*

Another example is shown below in Figures 11 and 12, demonstrating what happens when “Route Around Blocks” box is checked. The example is shown on a random distribution of blocks after no floorplanning has been conducted.

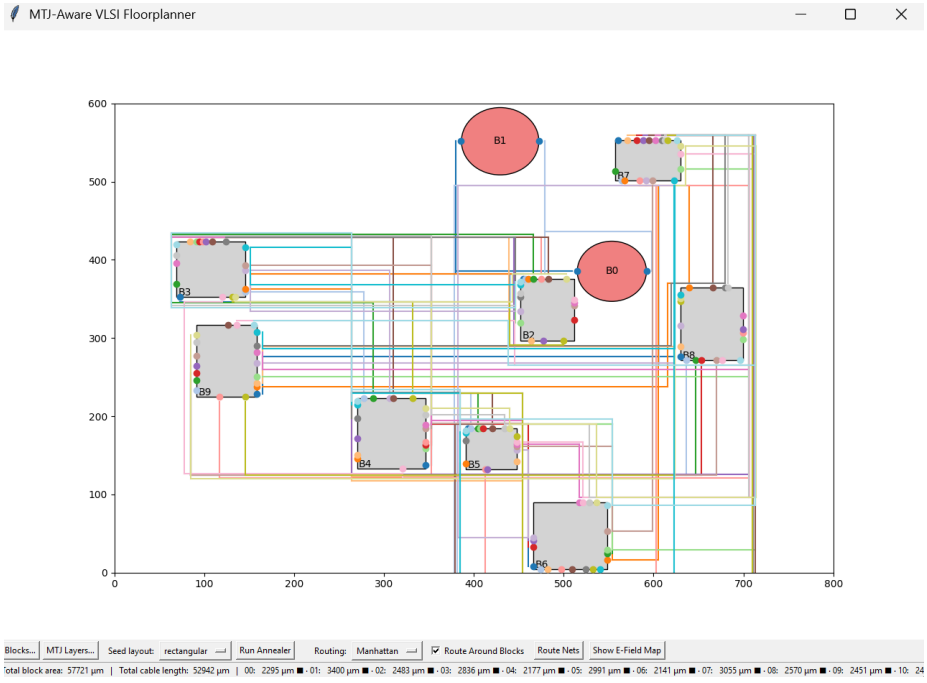

Figure 11: *When the “Route Around Blocks” Box is Checked*

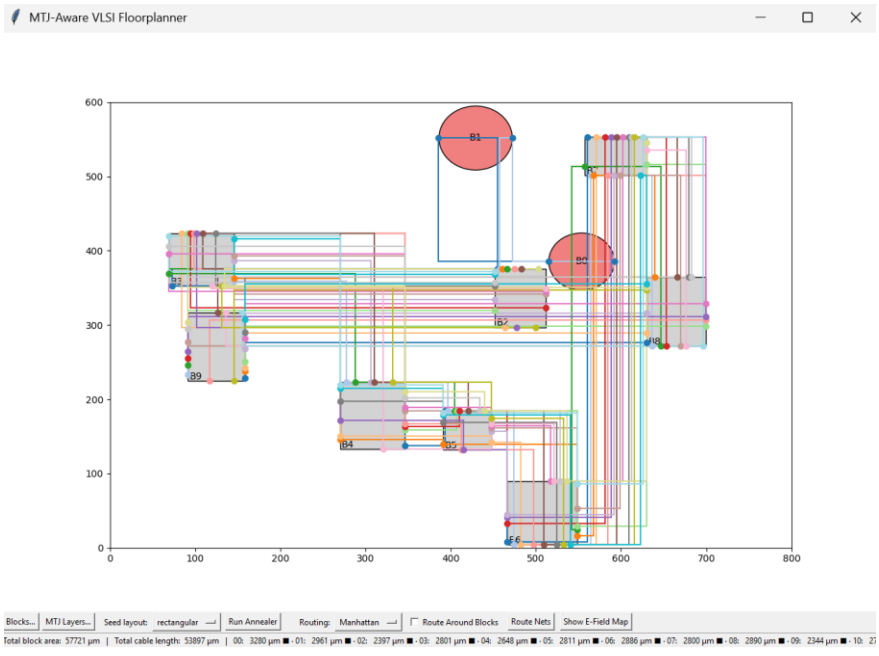

Figure 12: *When the “Route Around Blocks” Box is not Checked*

7. Future Research Directions

The current framework, while functional and innovative, serves as a robust platform for numerous exciting future research and development avenues that could further enhance its practical utility and academic contribution:

1. **Closed-Loop Routability-Driven Floorplanning:** A significant extension would be to transform the E-field (wire density) map from a post-hoc analytical tool into an active component of the optimization loop. The calculated wire density, or a derivative metric representing congestion, could be incorporated as an additional penalty term in the simulated annealing cost function. This would enable the floorplanner to proactively seek placements that not only minimize area and respect magnetic constraints but also promote more uniform wire distribution and alleviate potential routing hotspots, thereby directly improving global routability.

2. **Integrated Thermal Awareness:** Given that high wire density often correlates with increased current density and localized power dissipation, the E-field map could serve as a first-order proxy for a thermal map of the chip. Future work could involve developing more sophisticated electro-thermal models and integrating thermal constraints or objectives (e.g., minimizing peak temperature, ensuring thermal uniformity) into the SA cost function. This would be particularly relevant for 3D ICs or designs with high power density.

3. **Advanced Magnetic and RF Component Modeling:** The current MTJ model employs a simplified radial keep-out zone based on a dipole approximation. For more complex magnetic components, such as spiral inductors in RF circuits or MTJs with non-uniform field patterns, more sophisticated field modeling techniques (e.g., using analytical expressions for specific geometries, or even linking to finite element method (FEM) field solvers for high accuracy) could be integrated. This would allow for more precise and potentially less conservative keep-out region definitions.

4. **Timing-Driven Physical Design:** The framework could be extended to incorporate timing awareness by estimating net delays (based on wire length and potentially buffer models) and including critical path delay or timing slack objectives within the SA cost function. This would involve integrating a static timing analyzer (STA) or a proxy for timing estimation.

5. **Systematic Benchmarking and Comparative Analysis:** To rigorously evaluate the efficacy of the MTJ-aware approach and the utility of the E-field visualization, the tool should be applied to a suite of standard academic or industry-derived benchmarks. These benchmarks might need to be augmented with representative MTJ or RF components. Key metrics for comparison against traditional (non-MTJ-aware) floorplanners or other specialized tools would include final layout area, total wire length, routing completion rates, computational runtime, and, uniquely, quantitative measures of magnetic integrity and wire density uniformity.

6. **Exploration of Alternative Optimization Algorithms:** While simulated annealing is a versatile heuristic, exploring other optimization techniques such as genetic algorithms, particle swarm optimization, or even machine learning-based approaches (inspired by projects like AlphaChip/AlphaDev) for the MTJ-aware floorplanning problem could yield interesting comparative results or performance improvements.

7. **Integration with Industry-Standard EDA Flows (e.g., OpenROAD):** A long-term goal could be to explore methods for integrating the core concepts of MTJ-aware constraint management or the E-field/wire density analysis capabilities into open-source EDA platforms like OpenROAD. This would allow for validation and application of the techniques developed within a more comprehensive physical design environment and on larger, more complex designs.

8. Conclusion

This project has culminated in the successful development and demonstration of a VLSI physical design tool that uniquely integrates magnetic-field awareness into its core floorplanning and routing algorithms, complemented by an innovative 3D E-field (wire density) visualization capability. The system effectively employs simulated annealing to optimize block placements while enforcing both standard geometric constraints and physics-derived magnetic keep-out zones for MTJ components. The implemented routing algorithms demonstrate an ability to navigate these complex obstacle fields, ensuring connectivity while preserving magnetic integrity. The E-field visualization module offers designers a novel and intuitive means to analyze post-routing wire density, identify potential congestion or EMI hotspots, and understand the interplay between component placement and routing resource utilization.

The work undertaken directly addresses the challenges posed by the increasing integration of magnetically active components in modern SoCs. The project has involved substantial problem-solving across algorithm design, software engineering, and data visualization. The resulting framework serves as a functional prototype and establishes a versatile platform for future research into advanced topics such as closed-loop routability-driven design, integrated thermal management, and the physical design of next-generation heterogeneous SoCs.

9. Code Repository

The complete source code for this project is maintained and accessible via the following public GitHub repository:
https://github.com/CanAfacan/Magnetic-Field-Aware-Floor-Planning-and-Obstacle-Driven-Routing-for-MRAM-MTJ-

REFERENCES

[1] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, 2nd ed. Cham, Switzerland: Springer, 2022. doi: 10.1007/978-3-030-96415-3.

[2] B. Dieny, R. B. Goldfarb, and K.-J. Lee, *Introduction to Magnetic Random-Access Memory*. Hoboken, NJ, USA: Wiley-IEEE Press, 2017. doi: 10.1002/9781119079415.

[3] D. Apalkov, et al. 2013. Spin-transfer torque magnetic random access memory (STT-MRAM). J. Emerg. Technol. Comput. Syst. 9, 2, Article 13 (May 2013), 35 pages. https://doi.org/10.1145/2463585.2463589.

[4] J. Lou, S. Krishnamoorthy, and H. S. Sheng, "Estimating routing congestion using probabilistic analysis," in *Proc. Int. Symp. Phys. Des. (ISPD)*, Apr. 2001, pp. 112–117

[5] S. Kirkpatrick et al., Optimization by Simulated Annealing. Science 220,671-680(1983). DOI:10.1126/science.220.4598.671.