Massively Parallel GPU Rasterizer for Next-Generation Computational Lithography

Loay Hegazy
Calibre
Siemens EDA
Cairo, Egypt
loay.hegazy@siemens.com

Mohamed Taher
Calibre
Siemens EDA
Cairo, Egypt
mohamed.taher@siemens.com

Sherif Hammouda
Calibre
Siemens EDA
Cairo, Egypt
sherif.hammouda@siemens.com

***Abstract*—This poster explores the important role of rasterization in both computer graphics and Electronic Design Automation (EDA), with a focus on its application in high-resolution mask synthesis, lithography simulation, and Optical Proximity Correction (OPC). Through an innovative approach utilizing GPU acceleration, our work addresses the substantial computational demands of rasterization with nanometer-scale precision, while ensuring efficiency and accuracy.**

***Keywords*—Rasterization, GPU, Electronic Design Automation, Optical Proximity Correction, High-Precision Algorithms.**

I. INTRODUCTION

Rasterization [1] is the process of converting continuous geometric shapes, such as polygons or curves, into a discrete grid of pixels, enabling various digital applications. While this technique is foundational in computer graphics and vision, it plays an equally critical role in EDA, particularly in OPC [2], where it simulates light propagation and resist behavior with high accuracy. Here, rasterization must be not only fast but also pixel-accurate and connectivity-preserving due to the high stakes of slight errors in circuit layouts. As semiconductor nodes shrink below a few nanometers, the computational demands for precision increase, necessitating the development of GPU-optimized algorithms.

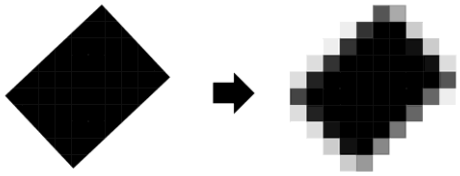

Figure 2: Example of converting a polygon into a pixel-based representation through rasterization.

II. BACKGROUND AND MOTIVATION

Traditional rasterization in graphics typically employs a binary coverage model, where pixels are either fully covered or not at all. Although this suffices visually in graphics, such an approach proves inadequate for computational lithography and OPC. These domains require fractional pixel coverage models to accurately simulate light intensity and resist behavior at nanometer scales. The challenge lies in maintaining connectivity and manufacturable geometries in thin features while processing billions of polygons and trillions of pixel evaluations. GPUs offer a solution by enabling data parallelism, yet they confront issues such as irregular memory access and lack of accuracy. Our research tackles these challenges with a GPU rasterization algorithm that ensures floating-point precision and sub-pixel connectivity.

III. METHODOLOGY

Our methodology is designed to optimize the rasterization process using GPUs:

1. **Initialization**: We start by setting all pixels in grid to zero, reserving shared memory for polygons based on the one with the maximum vertices, minimizing global memory latency.
2. **Polygon Assignment**: Polygons are assigned to thread blocks, allowing simultaneous processing. Vertices are transferred to shared memory, accelerating access and processing.
3. **Bounding Box Calculation:** A bounding box encloses each polygon, simplifying computations to focus on relevant grid sections. Boundary points are calculated to define minimal and maximal vertex coordinates.

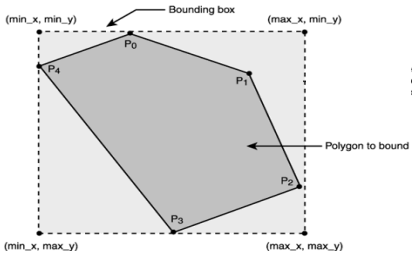

Figure 1: Example of computing bounding box for a polygon.

4. **Thread-Pixel Allocation:** Threads are mapped uniquely to pixels withing the bounding box, allowing detailed per-pixel assessments to determine their status relative to the polygon – inside, outside, or on the boundary.
5. **Pixel Classification and Processing:** Inside pixels are marked active, boundary pixels undergo further analysis to adjust edges and calculate trapezoidal areas for accurate boundary representation. Atomic operations ensure proper handling of boundary conditions with multi-polygon overlapping.

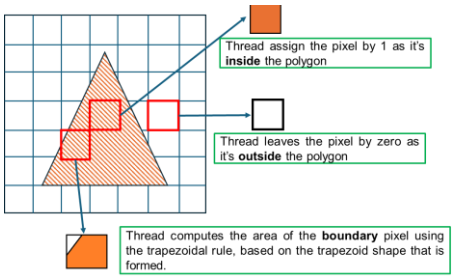

Figure 3: Simple example on pixel classification and processing

IV. RESULTS

Performance benchmarks on a NVIDIA H100 GPU highlighted significant improvements over CPUs in rasterizing Manhattan and curvilinear datasets, emphasizing GPU efficiency. The performance is compared against a highly optimized CPU algorithm that surpasses our algorithm when run on a CPU.

- **Manhattan Dataset:** Comprising 172,585,897 polygons (average of 8 vertices), GPU configurations showed drastic execution time reductions compared to CPUs—with speeds improving from 2.2 seconds versus 639 seconds in a 1:1 CPU:GPU setup to 0.29 seconds versus 27.5 seconds in a 32:4 configuration.
- **Curvilinear Dataset:** With 47,071,509 multigons (average of 114 vertices), GPUs maintained their edge, from initial runs of 29-30 seconds down to 0.9 seconds in a 32:4 setup, strongly outperforming CPU times.

Overall, GPU rasterization yielded up to 290× speedups for Manhattan shapes and 45× for curvilinear shapes. Accuracy benchmarks exhibited less than 1% absolute error against CPU results, validating high-fidelity processing across diverse geometries.

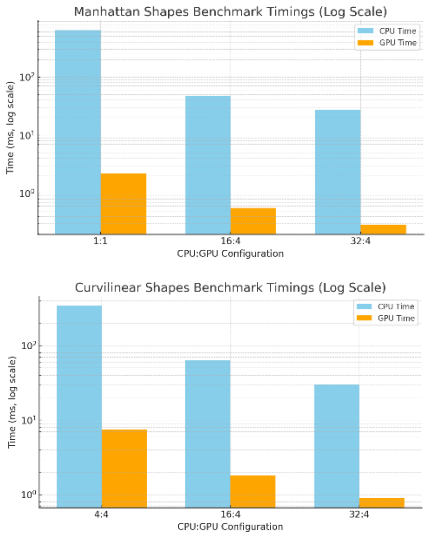

Figure 4: CPU and GPU runtimes for Manhattan and curvilinear datasets. GPU achieved large speedups with pixel errors under 1%.

V. CONCLUSION

The findings effectively demonstrate the superior capacity of GPU-based algorithms in performing large-scale, high-precision rasterization tasks inherent to EDA workflows. By achieving substantial speedups and negligible error rates, we attest to GPUs' important role in advancing semiconductor design and manufacturing capabilities. Our results advocate for ongoing research into GPU optimization techniques to amplify scalability and functional outreach in increasingly complex applications.

VI. ACKNOWLEGMENT

The authors acknowledge the assistance provided by ChatGPT in refining sections and sentences throughout the writing of this poster.

REFERENCES

[1] W. Liu and R. K. Cavin III, "Rasterization theory, architectures, and implementations for a class of two-dimensional problems," Integration, vol. 6, no. 2, pp. 179–199, 1988, doi: 10.1016/0167-9260(88)90038-7.

[2] T. Matsunawa, B. Yu, and D. Z. Pan, “Optical proximity correction with hierarchical bayes model,” Journal of Micro/Nanolithography, MEMS, and MOEMS (JM3), vol. 15, no. 2, p. 021009, 2016.

[3] Yixiao Ding, Chris Chu, and Xin Zhou. 2015. An efficient shift invariant rasterization algorithm for all-angle mask patterns in ILT. In Proceedings of the 52nd Annual Design Automation Conference (DAC '15). Association for Computing Machinery, New York, NY, USA, Article 72, 1–6.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.