# [15418] Project Milestone: Parallel VLSI Partition Algorithms Based on Message Passing Model

James Wu (jhensyuw) and Yueqi Song (yueqis)

Project Web Page: https://github.com/yueqis/Parallel-VLSI-Partition

## 1 Summary of Work Completed

Firstly, we have completed the starter code to parse circuit inputs and have constructed the data structure capable of storing necessary information required for VLSI partitioning, such as cells, nets, and the partitioner. In addition, we have completed and thoroughly tested the sequential implementation of our simulated-annealing-based algorithm. While the final implementation of the MPI version for the simulated-annealing-based algorithms is not completed, we've conducted extensive testings on various preliminary parallel versions. For example, we leverage batch strategies similar to assignment 4, where cells are distributed as tasks across different processors. Additionally, we're considering the implementation of diverse communication strategies, such as broadcast between nodes or employing a master/slave node setup. We will identify and employ the most efficient strategies that yield the greatest speedup. We're also exploring tactics like load balancing across processors by strategically sorting cells based on the number of pins they contain. These measures are helpful to optimizing the performance and scalability of our parallel algorithm implementation.

Secondly, we have done comprehensive survey on existing algorithms on partitioning. We explored various algorithms, including multilevel recursive spectral bisection [1], multilevel partitioning based on coarse approximation using Random Matching + Spectral Bisection + Kernighan-Lin (KL) [2], and multilevel paritioning based on coarse approximation using RM + heavy-edge heuristic + KL/Greedy-Graph + growing Boundary KL [3]. The main idea behind the second approach and the third approach are very similar, where they both contain three steps. Step 1 is coarsening, step 2 is partitioning, and step 3 is un-coarsening, but the second approach and the third approach use different algorithms for each step. Among these three advanced approaches, the first approach is limited to spectral bisection, so we decided not to adopt this approach; the second approach is easier to comprehend and implement than the third approach, but shows not as good performance as compared to the third approach. Therefore, we decided to adopt a mixture of the second approach and the third approach, where we use Heavy Edge Matching (HEM) for step 1, KL algorithm for step 2, and KL refinement for step 3. Using this combination, the algorithm is not too difficult to understand, implement, and parallel, while maintaining good performance.

## 2 New Goals and Deliverables

Currently, we are mostly up to date with our schedule, and the only thing we miss now is paralleling over the multilevel partitioning algorithm. We expect to be able to produce all our deliverables, if we are able to obtain satisfying results. We are uncertain about deploying on GPU, where we planned to write code that could take advantage of GPUs and test our algorithms over different types of GPUs. This is because based on our survey of past research, efforts on deploying such algorithms on GPU is quite limited, so deploying such algorithms on GPUs may either be very complicated or gives less good speedup than expected. Other than that, our list of goals is as follows:

- Implement a basic MPI parallel partitioning algorithm based on Simulated Annealing
- Implement the MPI parallel multilevel partitioning algorithm based on HEM + KL + KL refinement
- If time allowing: implement another MPI parallel partitioning algorithms based on other research
- Integrate GPU CUDA acceleration with the MPI solutions based on other research
- If time allowing: solve a more difficult problem formulation, such as hyper-edges and non-unit weight costs
- If time allowing: implement different CUDA codes and compare their combinations with different MPI parallel algorithms

# 3 New Schedule

- Checkpoint 1 (4/16 - 4/19) Complete implementation of two MPI parallel algorithms and refine the performance
- Checkpoint 2 (4/20 - 4/22) Complete a third version of MPI parallel algorithm and compare the results
- Checkpoint 3 (4/23 - 4/26) Collect the experimental results of all MPI algorithms; Draft the CUDA codes for GPU accelerations
- Checkpoint 4 (4/27 - 4/29) Complete implementation of CUDA codes
- Checkpoint 5 (4/30 - 5/2) Compare results from different algorithms and different CUDA accelerations; Draft the final report
- Checkpoint 6 (5/3 - 5/5) Complete the final report; Prepare for the poster

# 4 Demo Plan

Our demo plan includes three components. Firstly, we will showcase graphs illustrating the comparative performance and speedup achieved across various MPI parallel algorithms. This visual representation will offer insights into the efficiency and scalability of different parallelization strategies, aiding in algorithm selection and optimization decisions. Secondly, we will present graphs demonstrating the speedup enhancements resulting from the integration with GPU CUDA. These visuals will underscore the impact of GPU acceleration on message passing parallel programming model, showcasing the potential for performance improvements across diverse problem domains. Lastly, we will analyze experimental results obtained across different problem sizes, providing a comprehensive understanding of our solution's scalability and robustness.

# 5 Preliminary Results

No, we implemented two parallel algorithms simultaneously, so we still cannot compare their performance.

## 6 Challenges and Concerns

One concern lies in the time overhead associated with multilevel partitioning. The coarsening and uncoarsening phases of this approach requires additional time, potentially impacting overall efficiency. Furthermore, achieving load balance in multilevel partitioning poses challenges compared to the single-level version. Given the interdependency among cells sharing common nets and the varied pin connections, devising an effective strategy to coarsen the graph into tasks with equitable loads and distributing them across processors can be a challenge.

Another concern is about the second phase of our project, where we are not sure about the potential speedup gained from utilizing GPUs. While existing research suggests promising improvements, there remains uncertainty regarding whether the synchronization costs—such as data transfer between CPU and GPU, workload distribution, and scheduling—could potentially outweigh the benefits of GPU acceleration. This may lead to an overall degradation in performance, so we may need to evaluate which tasks should be shifted to GPU to ensure optimal utilization of resources.

## References


[1] Stephen Barnard and Horst Simon. "A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems." In: Jan. 1993, pp. 711–718.

[2] Bruce Hendrickson and Robert Leland. "A multilevel algorithm for partitioning graphs". In: *Proceedings of the 1995 ACM/IEEE Conference on Supercomputing*. Supercomputing '95. San Diego, California, USA: Association for Computing Machinery, 1995, 28–es. ISBN: 0897918169. DOI: 10.1145/224170.224228. URL: https://doi.org/10.1145/224170.224228.

[3] George Karypis and Vipin Kumar. "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs". In: *SIAM Journal on Scientific Computing* 20.1 (1998), pp. 359–392. DOI: 10.1137/S1064827595287997. eprint: https://doi.org/10.1137/S1064827595287997. URL: https://doi.org/10.1137/S1064827595287997.