# Project Title & Contributors

**Title:** Research Paper Review & Parallel Strategy Proposal  
**Group Members:**  
• Alishba Naveed  
• Hamza Ahmed  
• Saffi Muhammad Hashir  
**Submission Date:** 20 April 2025

# **Selected Research Paper**

**Title:** A Community Detection-Based Parallel Algorithm for Quantum Circuit Simulation Using Tensor Networks  
**Authors:** Alfred M. Pastor, Jose M. Badia, Maribel Castillo  
**Published:** January 2025 (Springer)

# **Why This Problem Matters**

• Simulating large quantum circuits on classical machines is challenging due to:  
– High memory usage  
– Exponential time complexity  
• Efficient parallel and memory-aware simulation strategies are required.

# **What is ComPar?**

**ComPar = Community-based Parallel Contraction Algorithm**  
• Treat tensor network as a graph  
• Apply community detection  
• Contract communities in parallel  
• Final contraction executed on GPU

# **Community Detection (Girvan–Newman)**

• Graph-based partitioning algorithm  
• Detects weakly connected clusters (communities)  
• Enables independent contraction of sub-networks in parallel

# **ComPar Execution Pipeline**

Circuit → Tensor Network

Tensor Network → Community Detection

Community Contraction (CPU, Parallel)

Final Global Contraction (GPU)

# **Benchmarking Results**

**Tested on Circuits:**  
• QFT (Quantum Fourier Transform)  
• RQC (Random Quantum Circuits)  
• GHZ (Greenberger–Horne–Zeilinger)

**Performance:**  
• Outperforms GN\_gpu & ParSli  
• Simulates up to **144 qubits**

# **Our Parallel Strategy**

**Technologies Used:**  
• **MPI** – Community-wise distribution  
• **OpenMP** – Intra-community contraction  
• **OpenCL** (Optional) – Final contraction on GPU  
• **METIS** – Alternate partitioning strategy

# **Technology Usage Breakdown**

**MPI – Message Passing Interface**  
• **Why:** Distributed memory parallelism  
• **Where:** Each MPI process contracts one or more communities

**OpenMP – Thread-Level Parallelism**  
• **Why:** Parallel contraction within a community  
• **Where:** Multithreaded tensor contractions inside MPI processes

**OpenCL – GPU Acceleration (Optional)**  
• **Why:** Speed up the final contraction  
• **Where:** Final tensor contraction after all communities are reduced

**METIS – Graph Partitioning Library**  
• **Why:** Course requirement and performance comparison  
• **Where:** Alternate to Girvan–Newman for graph partitioning

# **GN vs METIS – Partitioning Strategy**

We will compare both partitioning approaches:  
• **Girvan–Newman** (used in the paper)  
• **METIS** (used for coursework)

**Metrics for Comparison:**  
• Number of edge cuts  
• Contraction execution time  
• Overall speedup and efficiency

# **Tools & Libraries**

| Tool/Library | Purpose |
| --- | --- |
| MPI | Process-level parallelism |
| OpenMP | Thread-level parallelism |
| METIS | Graph partitioning |
| NetworkX | GN community detection |
| QXTools (Julia) | Tensor operations & contractions |
| Python/Julia | Preprocessing, orchestration, visualization |

# **Evaluation Plan**

We will:  
• Implement both **GN** and **METIS** versions  
• Benchmark on QFT, RQC, and GHZ circuits  
• Measure:  
– Execution Time  
– Memory Usage  
– Speedup

# **Final Summary & Goal**

• Implement and evaluate **ComPar** with **dual partitioning strategies**  
• Apply **hybrid parallelism** using MPI + OpenMP  
• Explore **GPU acceleration** using OpenCL  
• Fulfill **course requirements** and extend the research work