Design choice: Tomasulo. We chose Tomasulo's algorithm over explicit register renaming since Tomasulo seems to have more documentation. We are aware that Tomasulo's algorithm poorly scales to large instruction windows, but we believe that by choosing Tomasulo's algorithm over explicit register renaming, we will have more time to focus on advanced features such as turning our processor into a superscalar.

Progress report:

Contributions: Diagram (Nathan), Fetch (Yuqi), Queue (Yunxuan), Roadmap & report (together)

Functionalities: Fetching with parameterizable queue. These are the only current functionalities for checkpoint 1, but we will add more later.

Testing strategies: Verdi, assembly test cases. Ensured that instruction is fetched correctly and is added to instruction queue.

Timing and energy analysis:

- data required time: 1.965273 ns
- data arrival time: 1.878935 ns
- fmax: 532 MHz
- energy:

Roadmap:

Features and functionalities:

Decode state of the processor (must be done before the following features).

Re-order Buffer (ROB): will only be able to test after implementing the rest of the features.

Reservation stations: hold instructions until their respective functional units are ready.

Adder unit (should be mostly the same as mp_verif or mp_pipeline).

Integrate the provided multiplier into the processor as a functional unit.

Common data bus: data from functional units, ROB, and register file should all be connected.

Register File (architectural): should be the same module from mp_verif and mp_pipeline.

Connect RVFI for testing purpose (before doing above steps).

Future contributions: We are unsure of who will actually end up doing which parts of checkpoint 2, but our current plan is: Decode state of the processor, ROB, Reservation stations(Yuqi), Adder unit, multiplier(Yunxuan), Common data bus, Register File, Connect RVFI(Nathan)