We implemented Tomasulo's Algorithm, excluding the memory and control operations. Our processor now has reservation stations, an adder unit and a multiplier unit, a common data bus, an architectural register file, and a reorder buffer. The ROB is capable of receiving instructions out of order, and then committing them in the proper order.

Progress report:

Contributions:
Yuqi: reservation stations, adder unit, multiplier unit, and dispatch.
Yunxuan: ROB and architectural register file.
Nathan: decode, CDB, and progress report.

Functionalities:
Capable of holding instructions in the queue until reservation stations are ready.
Capable of holding instructions in reservation stations until they are ready.
Capable of doing alu operations and multiplication instructions.
Can send several instructions to the ROB out of order and the ROB will commit them in order.
Architectural register file connected to the reservation stations and the ROB through the common data bus.

Testing strategies: Verdi and assembly test cases. We used the provided assembly test cases, ooo_test.s and dependency_test.s to ensure that the ROB was committing the correct data. We looked at verdi to confirm that the instructions were reaching the ROB out of order, but were then being committed in the correct order by the ROB as quickly as possible.

Timing and energy analysis:

- data required time: 1.967688 ns
- data arrival time: 1.966129 ns
- fmax: 509 MHz
- energy: not required for checkpoint 2

Roadmap:

Features and functionalities:

Control instructions: need branch prediction unit connected to fetch and decode.

Memory instructions: need address unit, load buffers, and a memory unit.

Cache integration: need to integrate with instruction cache and decode cache.

Future contributions: We are unsure of who will actually end up doing which parts of checkpoint 3, but our current plan is: memory instructions - (Yuqi), cache integration - (Yunxuan), control instructions - (Nathan).