**Progress report: cp3**

Load store queue: leon ku

cache: jimmy yu

arbiter, adaptor : Haoyu Yu

verification and debug: group

Advanced feature: collapsed queue, forwarding in load store queue.

Testing strategy:

We first tested all the functions without load store brach jump in magic memory, then tested the functions of load/store branch/jump separately, finally ran through coremark, and then finally hooked up the cache and cpu and arbiter and adaptor all together, tested load/store, branch/jump separately, and finally tested all the functions together

lsq size: 8

arbiter: D-cache priority

adaptor: burst adaption for bmem

**Progress report: advanced feature (tentative)**

Branch predictor: Haoyu Yu

Superscalar: leon ku

Pipelined cache: jimmy Yu

Having multiple execution units (e.g. integer arithmetic units, floating-point units, load/store units, etc.), allowing multiple instructions to be executed in parallel.

Cache accesses are split into multiple pipeline stages, e.g., address computation, data reads, data writes, etc. Allowing multiple cache access operations to be performed simultaneously improves the throughput of the cache subsystem. Wasted CPU cycles due to cache access latency can be reduced. Streamlined caches are typically more complex and require a finer-grained design to avoid access conflicts and maintain data consistency. Each of these techniques helps to increase instruction-level parallelism (ILP) of the CPU, thereby improving processor performance while keeping the clock frequency constant. However, these techniques also increase the complexity of the processor design, requiring delicate hardware mechanisms to properly handle multiple concurrent operations and avoid performance bottlenecks.