CS154, Lab 3 Section

Thanawat Techaumnuaiwit

UCSB

Mar-7-2024

Lab Overview

In this lab you will be building a Reorder Buffer. We'll go over:

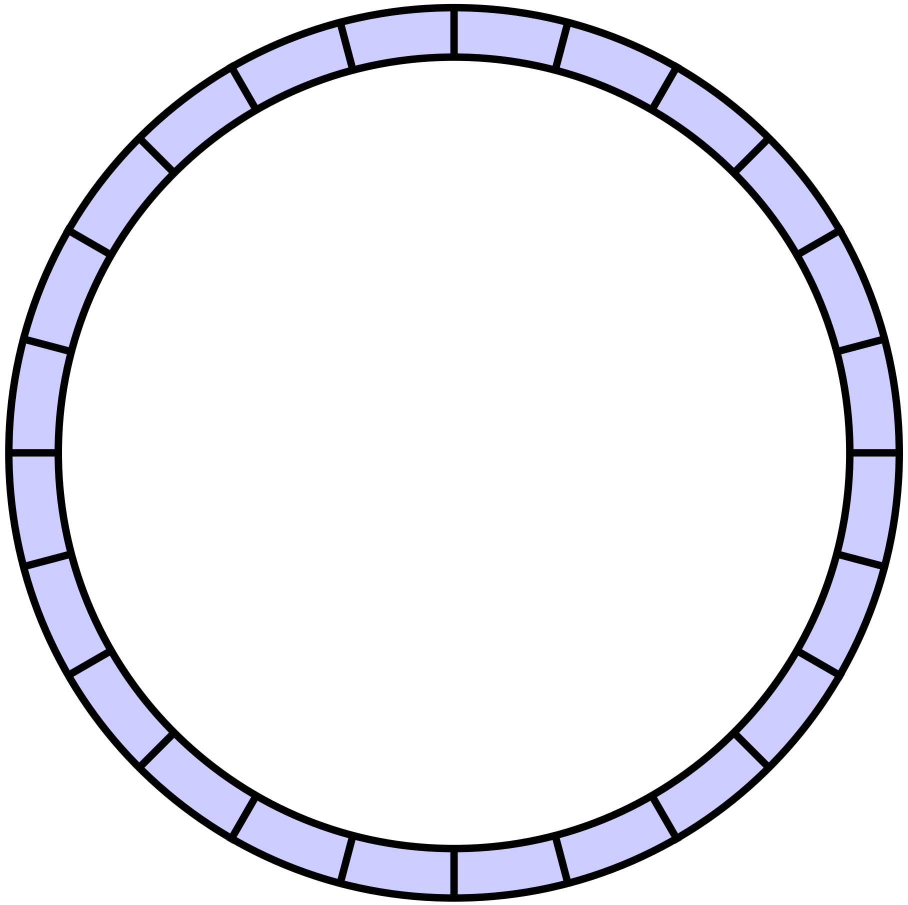
1. Review from lecture: What is a "reorder buffer"
2. A useful mental model for a reorder buffer
3. What are some of the things (edge cases) you need to think about when designing your ROB

Ring Buffer

Fun fact - a friend of mine actually had to design a ring buffer for an internship interview! So, it'd be good to know how these things work.

A ReOrder buffer is a **ring buffer** or a **circular queue**: https://en.wikipedia.org/wiki/Circular_buffer

Ring Buffer



Ring Buffer

The "queue" is kept in an "array" that wraps around. The "first" and "last" elements of the array needs to be kept track of in a pointer - because the array has to wrap around! You'll need to make two registers to keep track of this:

1. `commit_pointer` - the first element of the circular buffer. This corresponds to the next instruction we want to commit.
2. `alloc_pointer` - the last element of the circular buffer. Corresponds to where we want to allocate the next instruction that we recieve.

The ROB interface

The ROB has 3 “interfaces”:

1. Alloc interface - inputs and outputs signals relating to new allocating new instructions. When `rob_alloc_req_rdy_o` is held high - it means that your ROB is ready to recieve new allocate requests! A good question to ask is, when is the ROB *not* ready to recieve new requests?
2. Writeback interface - When is writeback occuring. When there’s an instruction writeback - that’s when the instruction “finishes”.
3. Commit interface - The “main” output of the ROB. The main questions to ask are: What is the next ROB slot to commit? What should happen when something gets “committed”?