# SocLib's way of handling locks

Traditionnal setups implements locks through RCU (Read/copy/update) -- ie atomic -- operations. This is archieved by not releasing the bus between a read and a write operation. Most of the time with SocLib designs, we have NoC-centric designs rather than buses therefore we can't lock bus access between operations, and can't avoid race conditions.

A new scheme is introduced in a specific ram component:

- Every accessible word is a spin lock
- Read operation returns the spin lock's status
    - 0 if not locked
    - other if locked
- Read operation locks the spin lock (so the spin lock is always taken after a read)
- Write operation releases the spin lock (even if locked by another CPU)

# Weirdness

- Usual setups implements atomic operations which are used for locks
- Soclib implements spin locks which are used to protect atomic operations