Cephalopode Snapshot Memory Architecture

Jeremy Pope

2023-05-23

1 Idea

A memory system that supports ordinary reads and writes, and also reads from a saved "snapshot" of the contents of RAM. A redirect-on-write scheme is used to make this comparatively efficient. Allocation and free-list maintenance is not handled here, but is rather implemented on top of this by a different unit.

2 Design

2.1 Data stored

Each address is associated with:

- A cell in main memory, consisting of two nodes ("A" and "B").
- Three bits of metadata in a smaller, high-speed memory:
  - `live`, indicating which of A and B is the current version of the node.
  - `saved`, indicating which of A and B is the saved snapshot version of the node.
  - `mark`, indicating whether the cell is marked (only used by GC).

  Note that the two bits are independent; all four configurations are possible.

The memory controller itself stores an additional bit of state named `mode` that indicates whether we are maintaining a snapshot. We refer to the modes `0` and `1` as *non-snapshot* and *snapshot*, respectively.

2.2 Behavior

In non-snapshot mode (`mode = 0`), ordinary memory reads and writes to an address are both directed to node A/B as indicated by the address's `live` bit, regardless of its `saved` bit. There is no snapshot, so snapshot reads are not permitted.

Before we take a snapshot, we first set `saved = live` in the metadata for every address. This is slow but may be done incrementally (whenever the metadata memory is idle). Then to take the snapshot, we simply set `mode = 1`.

When in snapshot mode, ordinary reads are directed to A/B based on `live`, just as before. However, when an ordinary write occurs, `live` is updated to the negation of `saved`, and the write is directed based on this new value of

`live`. This ensures that the node indicated by `saved` is left unmodified (i.e., the snapshot is retained). Snapshot reads are unsurprisingly directed to A/B based on the `saved` bit.

To discard a snapshot, we simply set `mode = 0`.

3 Performance considerations

Every memory operation requires first reading metadata (although reads may be done by speculatively reading the entire cell, depending on how memory is set up). In the case of a write in snapshot mode, the metadata must also be updated, although the metadata write can be carried out in parallel to the main memory write. Caching metadata may improve performance; more aggressive schemes are also possible (e.g. user caches metadata themselves) but difficult to reason about, and should probably be added onto Cephalopode later if time permits.

4 Interface

Unless indicated otherwise, all of the operations listed should be handshaked with a reasonable protocol and have running time proportional to that of accessing memory. They are divided into groups. Operations within the same group may be assumed never to be invoked simultaneously (we should verify this, though). However, it is possible for simultaneous or overlapping requests to occur *between* the groups, so arbitration will be needed to handle this case, as most operations cannot safely run in parallel.

Group 1: ordinary

- `read : addr:ADDR -> dout:NODE`. Protocol: twophase or pulseecho. Performs an ordinary read.
- `write : (addr:ADDR, din:NODE) -> ()`. Protocol: twophase or pulseecho. Performs an ordinary write.

Group 2: snapshot

- `snap_prepare : ()`. Protocol: pulseecho. Discards the current snapshot if one exists (i.e., sets `mode = 0`), and launches the slow preparation process to take a new one. May be called from either snapshot or non-snapshot mode.
- `snap_take : ()`. Protocol: pulseecho. Takes a snapshot (sets `mode = 1`). May only be called when the system is in non-snapshot mode and is ready (see `snap_ready`).
- `snap_read : addr:ADDR -> dout:NODE`. Protocol: twophase or pulseecho. Performs a snapshot read. May only be called in snapshot mode.

Group 3: marking

- `mark_get : addr:ADDR -> bit`. Protocol: pulseecho. Reads the mark bit.
- `mark_set : addr:ADDR -> bit -> ()`. Protocol: pulseecho. Writes the mark bit.

Group 4: handshake-less

These operations should not have handshakes.

- `snap_ready : bit`. Indicates whether or not the system is ready to take a snapshot, i.e. whether we are done setting `saved = live` for every address. In snapshot mode, the value should be `0`.