Chapter 4: Foundations of Shared Memory - Supplemented Notes

These notes provide additional detail on the concepts and constructions of registers, drawing from the provided slides to give a more comprehensive understanding for the exam's essay questions.

Register Properties and Correctness

The slides illustrate the differences between the three main types of concurrent registers: Safe, Regular, and Atomic, emphasizing their behavior during overlapping read and write operations.

- **Safe Register:** When a read overlaps a write, the read can return **any value** within the register's range[111]. This is a very weak guarantee and a read can even return a "garbage" value that was never written[2].

- **Regular Register:** This provides a stronger guarantee. An overlapping read must return either the value of the last completed write or the value of the overlapping write[3]. The key takeaway is that the read will

  not return an arbitrary value[4]. The slides show examples of overlapping operations and how the outcomes are considered "regular" even if the value appears to "flicker" between old and new[5].

- **Atomic Register:** This is the most stringent form, where each read appears to return the value of the last write[6]. The slides demonstrate that with an atomic register, all operations can be linearized[7]. For example, if a read completes after a write has started and another read, it must return the value of the last write and not the old value from the first read[888].

Register Constructions (Essay Question)

The slides provide a "Road Map" for constructing more complex registers from simpler ones[999999999]. The implementations are designed to be "wait-free," meaning every method call completes in a finite number of steps, guaranteeing independent progress for each thread[10].

- **Constructing a Safe MRSW Register:** A **Safe MRSW (Multi-Reader Single-Writer) register** is built from an array of safe SRSW (Single-Reader Single-Writer) registers[11].

The single writer iterates through this array, writing the new value to each reader's dedicated register one at a time[12121212]. Each reader, in turn, only reads from its own location in the array[13131313].

- **Constructing a Regular Boolean MRSW Register:** A **Regular Boolean MRSW Register** can be built from a Safe Boolean MRSW register[14]. The key insight is that the writer only performs a write operation if the new value is different from the last value it wrote[1515]. This prevents the "flickering" that a safe register might exhibit if the same value is written repeatedly[16161616].

- **Constructing a Regular M-Valued MRSW Register:** This construction uses an array of M-valued Boolean registers and a unary representation for values[17]. The writer writes

  true to the location corresponding to the new value and then writes false to all lower locations[181818]. The reader scans from the lowest value up, returning the first value that is

  true[191919]. This ensures a regular property by only providing values that have been explicitly written[20].

- **Constructing an Atomic SRSW Register:** Atomicity is achieved by associating a timestamp with each value[21212121]. The writer increments a local timestamp and writes the new

  (value, stamp) pair[22]. The reader keeps track of the latest

  (value, stamp) it has seen and only updates its value if a new read has a higher timestamp[23]. This prevents a later read from returning an earlier value, a key violation of atomicity[242424242424242424].

- **Constructing an Atomic MRSW Register:** This is a more complex construction that uses an n-by-n array of StampedValues[25]. The writer writes the new value and timestamp only to the diagonal entries of the array[26]. A reader reads its own row's diagonal entry and then checks its entire column to find the

  StampedValue with the highest timestamp[27]. It then writes this maximum value to all entries in its own row, "helping" other readers and ensuring atomicity[28].

- **Constructing an Atomic MRMW Register:** This is the most complex construction, building an atomic multi-reader, multi-writer register from an array of atomic multi-reader, single-writer registers, one for each writer[29]. A writer first reads all other writers' registers to find the highest timestamp[30]. It then writes a new value with a timestamp one greater than the maximum it found to its own dedicated register[31]. A reader simply reads all the registers and returns the value with the highest timestamp[32]. This approach provides a linearizable view of the shared memory by ensuring all reads and writes are ordered correctly by their timestamps[33].