Theory Assignment 2: Safe & Regular Registers
CO21BTECH11004

Que 4.2)
Sol:
True, Given construction works.

There are two cases: -
Case 1:- Reader and writer are not overlapping
In this case, the reader will always read the value written by the last writer that precedes the reader.

Case 2:- Reader and writer are overlapping
Given that each block in the array is a regular Boolean SRSW register.
If there is an overlap, the reader will read either the old or new value since the array is made of a regular Boolean SRSW register.

Since in both cases, regular register behavior is shown, therefore it is **true** if we replace the safe boolean SRSW register with an array of regular SRSW registers, then construction yields a regular boolean MRSW register.

Que 4.7)
Sol:
Yes, this proposal will work.

Here for proof of correctness first satisfy conditions (4.1.1), (4.1.2) and (4.1.3) for atomic, and all values from (0 to M) can be read if some write set them for M-valued.

No read call returns a value from the future:
It is never the case that $R^i -> W^i$ **(4.1.1)**
Proof:-

Assume for a reader $R^i \rightarrow W^i$ is true.

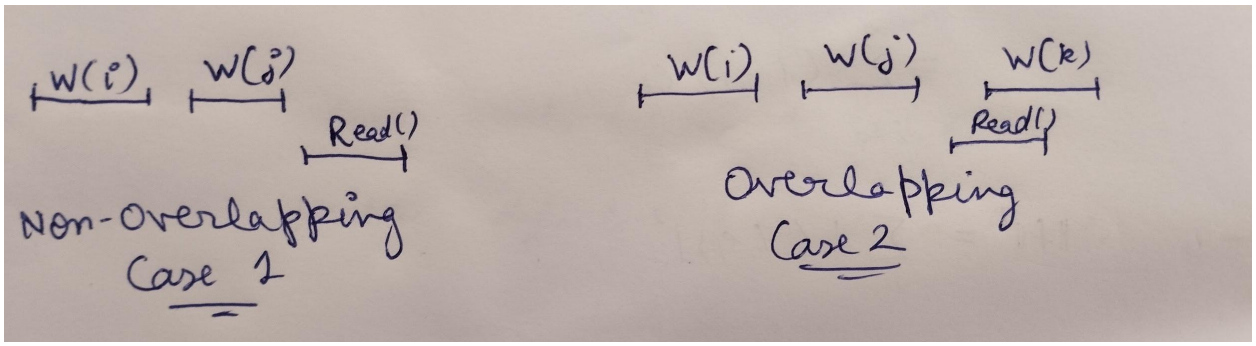
Let's say W^i sets r_bit[i] to 1. If a reader (R^i) is reading r_bit[i], then 1 should be written at the reader's block in the array. If 1 is written in the reader's block, it can happen only if the writer has written it. But this would not imply $R^i \rightarrow W^i$, i.e., the reader precedes the writer. This contradicts our assumption.

So, **Condition (4.1.1) holds.**

No read call returns a value from the distant past, that is, one that precedes the most recently written nonoverlapping value:

It is never the case that for some j, $W^i \rightarrow W^j \rightarrow R^i$ **(4.1.2)**

Proof : -



Case 1:- Non-overlapping condition

Case 1a: $j > i$

If $j > i$, then W^j will set r_bit[j] to 1, and all the bit less than j (including i) to 0. So when a reader starts from the starting of the array, it would stop at r_bit[j] as it set to 1, and read j.

Case 1b: $j < i$

If $j < i$, then W^j will set r_bit[j] to 1, and all bit less than j to 0. Here r_bit[i] is still set to 1. When the reader starts from the starting of the array it would stop at r_bit[j] as it is set to 1, and read j, and will never go to r_bit[i].

Case 2:- Overlapping condition:- Here reader overlaps with W^k

In this case if writer (W^k) has not set r_bit[k], same as case 1.

If writer (W^k) has set r_bit[k], then,

- If (j>i), r_bit[i] would be set to 0, so for any value of w other than i,j , reader would reader would found either r_bit[j] or r_bit[k] to be set to 1, so would reader j or k.
- If (i>j), then r_bit[i] would not be set to 0.
    - If k is less than j and i, then reader would find r_bit[k] as set to 1, and read w.
    - If k in between j and i, then the reader would find r_bit[j] or r_bit[k] as set to 1, and read j or k.
    - If k greater than j and i, then also reader would find r_bit[j] or r_bit[k] as set to 1, and read j or k.

So in all possible cases, by following the order in the condition, the reader would never be able to read i.

So, Condition (4.1.2) holds.

If $R^i -> R^j$, then i ≤ j **(4.1.3)**

This condition states that an earlier read cannot return a value later than that returned by a later read.

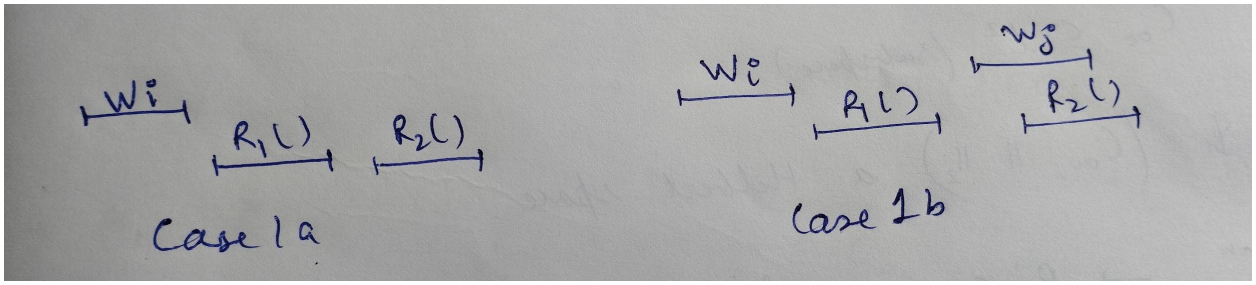
by a later read.

Proof: -

Here, underlying registers in each array block are atomic boolean SRSW registers.

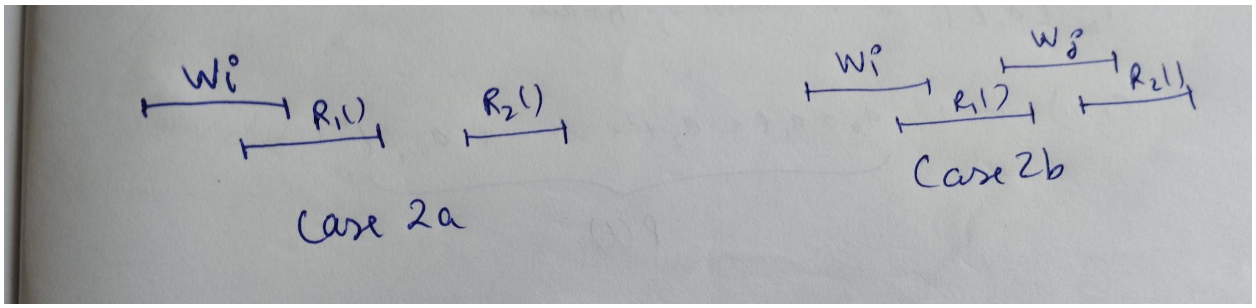
Here the second reader can read different value as read by first if there is a writer after first reader or first reader overlap with some writer.

Case 1:- Reader 1 does not overlap with the writer



Case 1a:- If no writer between Reader 2 and Reader 1, then Reader 2 would read the same value as Reader 1.

Case 1b:- If there is a writer between Reader 1 and Reader 2, based on the condition it overlaps or does not overlap with Reader 2, Reader 2 will read either value read by Readen 1 or written by the writer.
Case 2:- Reader 1 overlaps with writer



Case 2a:- If no reader between Reader 1 and Reader 2, then Reader 1 will read the value written by the writer or the previous written value and Reader 2 will read the value written by the writer.
Case 2b:- If there is a writer between Reader 1 and Reader 2.
As underlying atomic boolean SRSW registers, if R1 reads i, then R2 can read i or j.
If R1 reads j then R2 will read j.
So, Condition (4.1.3) holds.

Condition for M-valued:- The read() call in the construction always returns a value corresponding to a bit in 0..M set by some write() call.
Proof: -
If the reader thread is reading r_bit[j], then some bit at index j or higher, written by a write() call, is set to 1.
When the register is initialized, there are no readers; the constructor sets r_bit[0] to true. Assume a reader is reading r_bit[j], and that r_bit[k] is true for k ≥ j .

- If the reader advances from j to j + 1, then r_bit[j] is false, so k > j (i.e., a bit greater than or equal to j + 1 is true).
- The writer clears r_bit[k] only if it has set a higher r_bit[l] to true for l>k.

Hence proved.
Therefore given proposal is correct and will work for atomic M-valued boolean SRSW register