e)
For speeding up right shifting without wasting extra space for a shadow buffer, we can use shuffle_sync primitives. With the “shuffle_up_sync” method we store the value at the current thread.idx in the input, (in the single warp case this = lane within the warp), into a value (x for instance) which is stored in the register, updating our own value within the input array to the value stored in the previous threads register. This allows us to avoid race conditions even without a sync_threads as shuffle_up_sync is a warp level primitive that synchronizes this operations for all participating lanes in the warp.

Total time (shadow buffer method): 2.731 ms
Total time (warp shuffle_up_sync): 2.729 ms

f)
For an array size of 1024 there will be 32 warps (1024/32 = 32, 1 block) necessary for the computation. Warps at the boundaries (warp 0 shares a right boundary with warp 1) need to synchronize right shifting as the shuffle_sync primitive is not on the block level. To do this we need a shared memory buffer of size 32-1, the number of shared boundaries. Each right-most thread within a warp (lane 31) will write its current value into shared memory at its warp index. A sync_threads is needed after this to ensure the shared memory buffer contains all boundary values. After this the same warp-level primitive “shuffle_up_sync” can be used to right shift lanes 1-31, and the final lane will either be set to 0 if global thread index is 0 or be pulled from shared memory that contains the last warp’s final lane’s value.

Total time (shadow buffer method): 2.698 ms
Total time (warp shuffle_up_sync): 2.649 ms

g)
For an array size lf 4096 there will be 4 blocks with 32 warps each necessary for the computation. For cross-block communication we pass in a smaller buffer that will be used in a similar fashion to the shared memory buffer used for cross-warp communication, i.e holding the last value in the block to pass to the first thread of the right/adjacent block. To synchronize the write to the global buffer we use cooperative groups. Cooperative groups allows us to run grid.sync() which waits until the entire grid of blocks in the kernel have written their final value into the global buffer before continuing. After this we use the same method as f to perform cross warp synchronization as well as continuing use of the shuffle_up_sync method.

Total time (shadow buffer method): 2.694 ms
Total time (warp shuffle_up_sync): 2.797 ms

Results: Not much of a difference