King Saud University
College of Computer and Information Sciences
Department of Computer Science
CSC453 – Parallel Processing – Tutorial No 8 – Autumn 2022

Question

Let’s consider the following code of a parallel reduction that calculates the sum of an array of integers.

```
__global__ void reduce(int *g_idata, int *g_odata) {
extern __shared__ int sdata[];
// each thread loads one element from global to shared mem
unsigned int tid = threadIdx.x;
unsigned int i = blockIdx.x*blockDim.x + threadIdx.x;
sdata[tid] = g_idata[i];
__syncthreads();
// do reduction in shared mem
        for(unsigned int stride=1; stride < blockDim.x; stride *= 2) {
                if (tid % (2* stride) == 0) {
                        sdata[tid] += sdata[tid + stride];
                }
                __syncthreads();
        }
        // write result for this block to global mem
        if (tid == 0)
                g_odata[blockIdx.x] = sdata[0];
}
```

1. What is reduction.   reduce set of input elements to a single value output element
2. Why the threads are doing the reduction in shared memory.
3. This code is suffering from what is called interleaved addressing.
    a. What is interleaved addressing. Show it with a figure.
    b. Why warps in this code are not efficient.
    c. How can we fix that?

2- to accelerate access to memory and access to shared memory is fast
(briefly shared memory is fast to access)

3 - a - the thread not access to sequential address - go back to page 8 for figure do not forget to draw it in exam

3 - b - because there are a lot of not working thread in warp and this this thread working without any benefits

3 - c - by let the thread is sequential access to index and let all thread working in first time only and
last 64 index run it out of loop