Concurrent Computing

# Functionality & Design

Our program accepts images up to 1,800,000 pixels which is approximately a 1,400x1,400 pgm image. These are loaded in with a modified image loading system to save memory on the explorer tiles. To begin loading in the image the user clicks Button SW1. While loading of the image the main LED lights green. Once loading is finished the smaller separate LED flashes once per processing round and processing can be paused by tilting the board on its x-axis. Doing so will produce a status report listing the current round, number of live cells and total processing time. Processing can be resumed by tilting the board back to a horizontal orientation.

To export press Button SW2. Doing so will immediately begin exporting the most recently completed round. The blue LED will be lit while exporting is taking place.

The image is internally double buffered which allows us to immediately begin exporting a complete, correct image whenever required. Rows of the image are distributed to worker processes to complete by passing references to pointers. The workers will always be working on either one of the buffers at any time, but never both at the same time. At the end of each round pointers tracking the two buffers are swapped and the next round of processing commences. By using pointers to describe data for workers rather than segments of memory we greatly reduce communication costs and can process at great speed. The downside of the double buffered approach is memory overhead that holding two of the images introduces.

To counteract this we optimized other parts of the program for memory such as the loading and export stages. We also internally store the pgm image as bits of an array of uchars rather than the uchars initially read in.

# Tests & Experiments