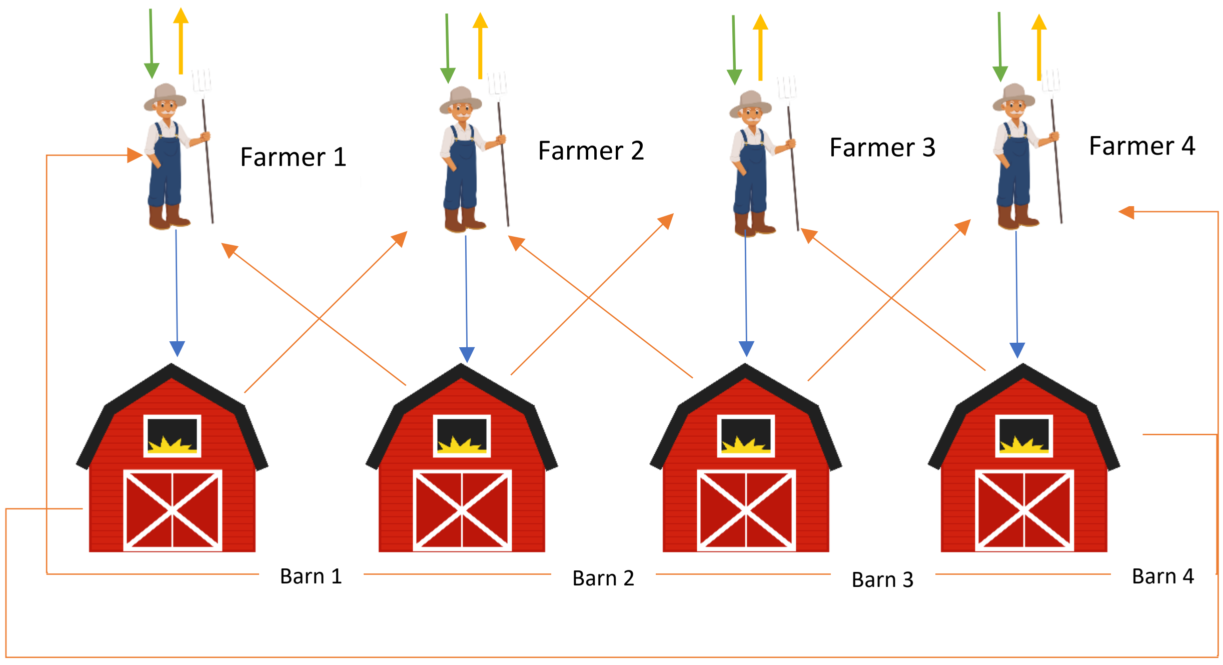
Juan Quintero
jq14772@my.bristol.ac.uk
David Shin
js16397@my.bristol.ac.uk
Group 6 (Concurrent Computing)

Introduction

Image processing is one of the areas that parallel computing is the most needed for. It is highly inefficient to process images without dividing them into smaller sections of itself. A perfect example of this, is the game of life in which single pixels change, potentially every round, according to the rules of the game. For this reason, the xCore 200 board was used to create a model of the game and run simulations on it over different inputs.

Functionality and Design

The parallel implementation was one of the most challenging parts of this project since there are so many various ways to tackle it. The following diagram showcases the approach taken to tackle the parallelization task.



The green and yellow lines represent the input and output respectively. The input was originally coming from the distributor, where the bit packing was being done. However, for large images this didn't work since the system was allocating memory that was far beyond the capacity of the hardware. To fix this, the input is done directly from the DataInStream function. On the same manner, the output is send directly to the DataOutStream function.

Since in The Game of Life the edges of the images must communicate with themselves, a channel of communication was created using barns. The barns receive the edges of the image from the farmer (blue line) and then distribute the respective edges to the farmers concerned. This process must be done every round.

Juan Quintero
jq14772@my.bristol.ac.uk
David Shin
js16397@my.bristol.ac.uk
Group 6 (Concurrent Computing)

This approach is almost deadlock-free which was one of the biggest concerns with other approaches. Moreover, we assure that all the farmers are performing the same task all the time since they are all created as equals, their only difference is the channels.

As part of the task, the implementation of hardware interaction was also required. The buttons, lights and tilting were successfully implemented. The system waits until the S1 button is pressed, it pauses when tilted and outputs when pressing the S2 button, all this being represented by the different LED lights. It is mentioned above how the approach taken was almost deadlock-free. This is because to pause the processing, when tilting, a deadlock is forced and when tilting back to original position it would unlock.

As mentioned briefly, bit packing was done to deal with the memory constraints of the system. Each pixel was packed into an 8-bit character. This size was chosen mainly because when communicating the edges, it is more efficient to send a single 8-bit character than a 32-bit int. The bit-packing is not done for the 16x16 image since it is such a small image, if bit-packed it could only be done into two chars and it would force the system to only use two farmers, reducing the speed greatly.

Finally, timers were also implemented and will be mentioned below, in the Test and Experiments section.

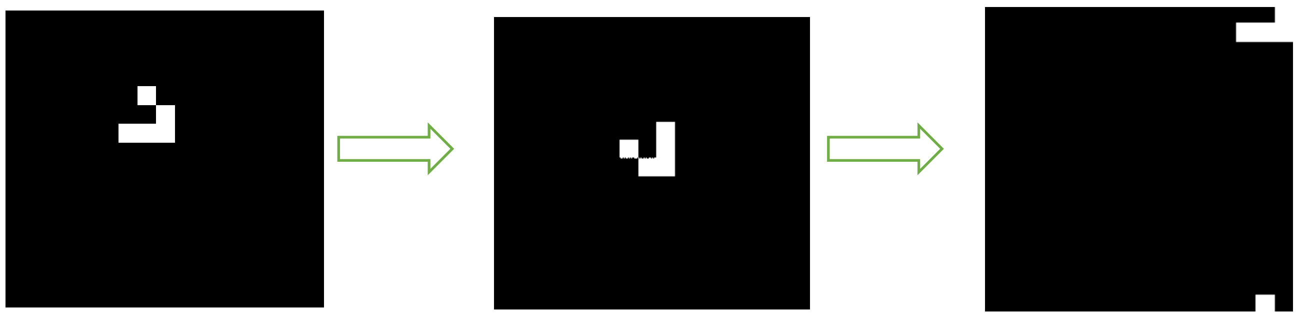
Tests and Experiments

After implementing the timer, some tests took place to compare the different approaches that could be taken for this task. The following table shows the processing time in seconds for 1000 rounds of the game of life.

Image size	Sequential	Parallel	Parallel bit-packed
16x16	1.59 s.	0.65 s.	0.41 s.
64x64	24.44 s.	10.33 s.	6.24 s.
128x128	97.15 s.	40.18 s.	24.40 s.
256x256	379.56 s.	--	97.03 s.
512x512	--	--	389.36 s.
1024x1024	--	--	1553.12 s.

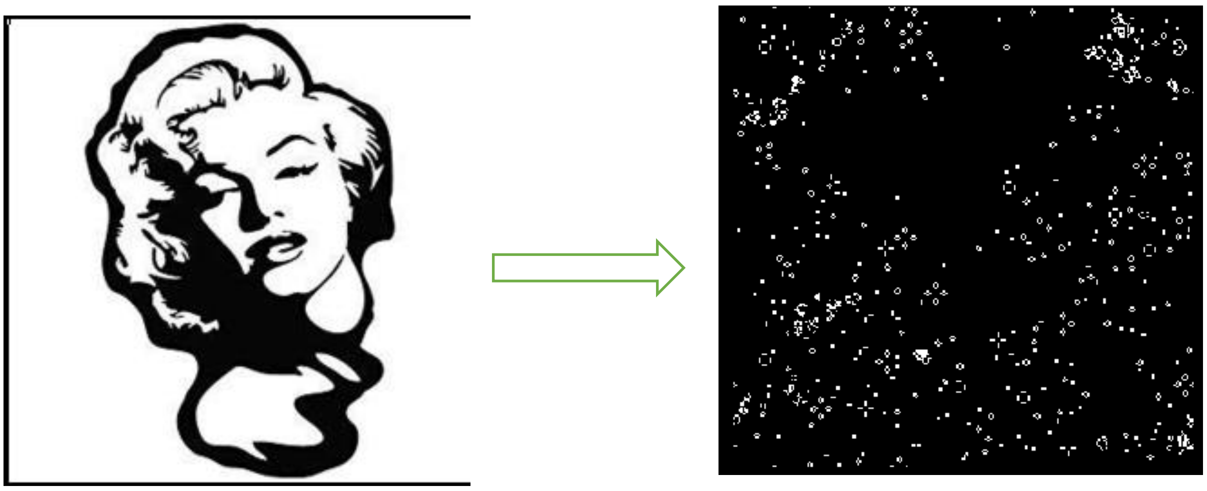
Table 1. Time after 1000 iterations

The following sequence of images showcase the 16x16 image being processed after 2 times and then 1000 times.



Juan Quintero
jq14772@my.bristol.ac.uk
David Shin
js16397@my.bristol.ac.uk
Group 6 (Concurrent Computing)

The following sequence showcases an image of Marilyn Monroe and then the same picture after 1000 iterations in the game of life.



Virtues and Limitations

While the greatest virtue of this implementation is the processing speed, it can also be a limitation when interacting with the user. Due to the hardware constraints (i.e. number of buttons), it is very hard to stop at the iteration that one wants exactly, the processing is going so fast that it is practically impossible to stop it with the 16x16 since in just 0.4 seconds, it will iterate 1000 times.

The system with the barns and the farmers is what causes the speed to be so low. It sacrifices thread space since there is one barn for each farmer, but it eliminates bottlenecks that could be originated if more than two farmers where trying to communicate through a barn.

The approach for the output aims to tackle this problem partially for other images, where the processing time takes longer. First, the image must be paused, by tilting the module and then ask it to output by pressing the S2 button. When tilting it, a message with the current round and time is printed, to let the user know in what state the program is.

The biggest constraint of this implementation is the fact that it can only process images that are divisible by 32. This is because the bit-packing divides the image by 8 and then having it distributed into 4 workers divides it by 4.

The maximum size image this implementation can process is 1312x1312.

Critical Analysis

As it can be seen in Table 1, the sequential approach is, as expected, the slower when processing all the images. However, it can also be observed how the sequential is able to process the 256x256 image whereas the parallel only approach is unable to do so. This is because for parallel, there exists multiple copies of the image that when summed up, exceed the capacity of the board. The bit-packing was done to deal with larger images. However, it encountered a similar problem when trying to process

Juan Quintero
jq14772@my.bristol.ac.uk
David Shin
js16397@my.bristol.ac.uk
Group 6 (Concurrent Computing)

the 1024x1024 image. To solve it, the distributor was removed since it was creating an unnecessary copy of the picture and then the farmers were given channels directly to input and output.

By removing the distributor, for parallel bit packed, it decreases the processing time greatly since the image is being send directly from the input and not as the parallel implementation in which the distributor was acting as a channel.

Finally, an upgrade for this implementation could be the option to add or remove workers depending on the size of the image. This would allow to process images that are not divisible by 32 since the possibility to add two workers would also allow images divisible by 48. This requires extra thread space that the current system lacks since there are certain hardware implementations that were required by the task.