Bio-inspired Computing

Z. Brock, N. Karst, M. Siripong

15 December 2005

Introduction

The goal of this project was to investigate and summarize the many ways in which cellular automata can deepen our understanding of the world around us. Cellular automata are extremely valuable because their behavior on micro- and macro-scopic levels is much like the behavior of the universe. Small, individually functioning units, from hydrogen atoms to plants in a jungle, interact with their neighbors to create a final emergent structure that is greater than the sum of its parts. Our general focus in this project was on systems that cellular automata can model easily and effectively. However we also found links in the other direction, where cellular automata like systems in nature had inspired and influenced computing. In order to investigate a wide variety of these links, the project was broken up into five parts: flocking, epidemiology, crystal formation, genetic algorithms and evolvable hardware, and pattern formation.

Flocking

The motion of a flock of birds flying south or of hundreds of fish in a school is completely unique to nature. This phenomena, where animals of a certain species group together and travel in high-density packs, is known in the computer science world as flocking. It is a particularly interesting phenomena because it is impossible for each entity in a flock to know exactly where it is headed - schools of fish have been recorded as being up to 17 miles long[1]. The implication here is some semblance of localized direction. If each fish takes into account information about the other fish surrounding it, it may set its own course accordingly. Similar to cellular automata, entities within a flock may employ simple rules to govern their speed and direction. This behavioral model for movement within a flock is a useful way to simulate the motion of animals in movies, games, or for scientific research.

Prior to 1987 when Craig W. Reynolds first described the behavioral model as a means for modelling flocks of birds[2], simulated flocks had to be created by giving each individual bird a predefined path. This is both tedious and dramatically limited, with the amount of input required exponentiated by the number of entities within the flock. But by having each entity follow simple rules rather than a set path, they may fly freely and infinitely while still exhibiting pseudo-natural behavior.

The rules originally described for flocking by Reynolds have held true, with some modification with time, as new programmers deem necessary. The three basic rules are as follows:

1. Collision Avoidance: Steer to avoid obstacles and crowding local flockmates
2. Alignment: Steer towards the average heading of local flockmates
3. Cohesion: Steer to move toward the average position of local flockmates

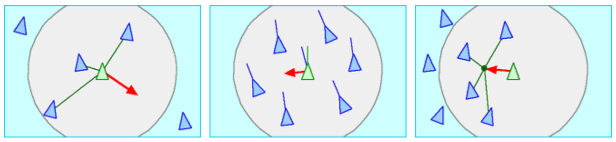

Figure 1: An illustrated explanation of the three primary rules for behavioral modelling of flocks. On the left, a boid considers a vector which prevents crowding its neighbors. In the center, a boid directs its motion in the average direction of its neighboring boids. Finally on the right a boid steers toward the average location of its neighboring flockmates. Image credits http://www.red3d.com

Simulations of birds, known as "boids," are able to fly independent of outside control using these basic rules and information it gathers from its "neighborhood." A standard interpretation of this neighborhood is a set distance and angle, where the angle describes the boid's line of sight. This is not to say that birds and boids can't see past this distance, but

[1]Reynolds, C.W.,1987. Flocks, herds and schools:a distributed behavioral model. Computer Graphics21:25-33.

[2]Reynolds, C.W.,1987. Flocks, herds and schools:a distributed behavioral model. Computer Graphics21:25-33.

rather that other boids within this sphere will influence its decisions. Many programmers have added additional rules to suit their needs, such as a low priority destination goal, a boid's desire to have a clear line of sight, resulting in the well known "V" formation[3], and many others[4].

Behavioral animation such as boids has been used frequently since its introduction in 1987. The army of penguins marching in Tim Burton's classic *Batman Returns* used a modified version of the original boid control code. The wildebeest stampede in the animated picture *The Lion King* also used behavioral animation. Behavoiral rule sets have also been used to model migration patters of certain species of fish, showing that individual input may lead to better living conditions for the entre school[5].

This concept of emergence - larger movements created by many smaller ones, occurs everywhere in nature, and its applications to computer science stretch far beyond animations. The same theory can be used to model other ntural phenomena such as fluid flow. Rather than birds and boids, each particle in the flow can be considered separately, with its own actions being governed by basic rules of physics and motion dependant on its local surroundings. While seemingly chaotic on the particle-level, these motions demonstrate chaotic emergence, and can be used to model flow on a larger scale.

Epidemiology

In recent years, modelling disease transmission using cellular automata has received much attention in the academic community. Whether describing common influenza[6], the recent spread of bird flu in Southeast Asia[7] or bubonic plague in medieval Europe[8], cellular automata have much information and insight to provide to biologists. In the following paragraphs, we will discuss the relevant parameters in disease transmission, the way in which these qualities translate to properties of cellular automata and the results and conclusions of experimentation.

Each disease has a unique mode of operation. Some spread incredibly quickly through a host population but do little lasting damage. Others spread slowly but have serious medical consequences for the infected. The dynamics of disease transmission are governed by four key quantities: infectivity, latency, duration and mortality rate.

As members of a susceptible population come in contact with a diseased individual, there is some probability that the sickness will spread. We define the probability of transmission under a given set of interaction conditions (e.g., casual, close and intimate interactions) to be the infectivity I of the disease High infectivities lead to a large percentage of the population contracting the disease. In cellular automata, infectivity represents the probability that a cell will contract the disease given that another cell in its neighborhood is infected. In actual disease transmission, the infectivity of a disease is actually a probability distribution based on characteristics of the susceptible individual. Age, sex and race can all play a part in contracting a disease. In most cellular automata models, these inconsistencies are ignored; infectivity is identical for all cells in the simulation.

After an individual has contracted a disease, the bacteria or virus lies dormant in the host. For some latency period λ, the host is not contagious. After a critical mass of pathogens has been replicated, the individual becomes contagious and begins to spread to the infection to others based on the infectivity probability I. In the vast majority of diseases, it is only after becoming contagious that the individual begins to show symptoms of the disease. This makes treatment and containment of diseases substantially more difficult. The concept of quarantine was developed as a countermeasures against this latency delay. By

[3] $http://mitpress.mit.edu/books/FLAOH/cbnhtml/home.html$

[4] $http://www.navgen.com/3d_boids/$

[5] $http://mywebpages.comcast.net/kils/pitcher.htm$

[6] Beauchemin, et al. "A simple cellular automaton model for influenza A viral infections", Journal of Theoretical Biology, August 2004, pgs. 223-234.

[7] Situngkir, Hokky. "Epdiemiology through Cellular Automata, Case of Study: Avian Influenza in Indonesia", submitted to Bandung Fe Board of Science, January 2004.

[8] Keeling and Gilligan, "Bubonic plague: a metapopulation model of a zoonosis", Proceedings of the Royal Society B, August 2000, pgs. 2219-2230.

segregating seemingly-healthy individuals who have had close contact with infected individuals, authorities hope to contain persons in the latency period. Without this kind of control the disease can spread unchecked. In cellular automata, latent cells behave in the same fashion as ordinary cells. As they are not contagious until the end of the latency period, they cannot spread the disease to their neighbors. And, as each latent cell has already contracted the disease, they are also unable to be reinfected. Thus, latent cells bide their time until becoming contagious.

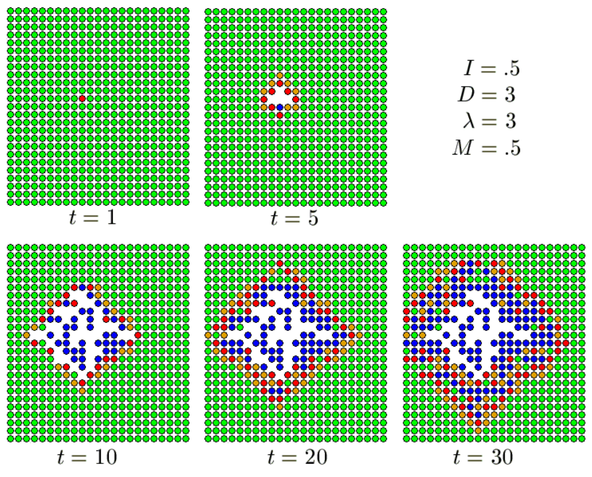

Figure 2: Above we see a disease transmission simulation given $I = .5, D = 3, \lambda = 3, M = .5$. Green dots indicate susceptible individuals; orange dots represent latent carriers; red dots represent contagious persons; blue dots denote recoverees; white space indicates a death. While not necessarily physically descriptive, the values used allow us to clearly see the dynamics of the system. Beginning with a single infected individual, we proceed to take snapshots over time. We can see that given stagnant neighborhoods, infections spreads out symmetrically from the point of origin. Any asymmetries are caused by interference with a disease-free boundary condition necessary for computation.

Every disease has a characteristic timescale. For instance, both the common cold and influenza last seven to ten days. We define the total period of time that an individual will host the bacteria or virus as the infection duration D. As with the latency delay, infection duration is normally a complex function of attributes such as age, sex, race and medical history. In modelling disease transmission, we ignore differences between individuals and claim that the disease has a characteristic duration. For this length of time, the infected cell will be contagious, spreading the disease to neighboring cells based on infectivity. At the end of the infection duration, the individual will either recover or die.

The fate of the infected individual is a function of the mortality rate M of the disease. This is perhaps the most crucial parameter in terms of physical relevance. Diseases such as common influenza that spread relatively easily through susceptible populations but rarely kill are far less concerning than diseases such as ebola that spread slower but kill nearly every person infected. As with the other parameters in our simulation, mortality rate depends on many attributes of the infected individual. We will disregard these dependencies and state that the mortality rate of a given disease is constant over the population.

Cellular automata models of disease transmission differ from normal cellular networks in several key ways. First, the transmission process is nondeterministic. That is, every simulation will be slightly different than any other experiment. In most automata, a set of deterministic conditions maps out the exact path to be taken by the system. In effect, a denumerable number of paths can be defined based on the size of the rule set. In systems involving probabilistic variability, the number of possible configurations is infinite. While producing rich dynamics, this indeterminism also makes it much more difficult to make definitive statements about the system. We are forced to speak in generalities rather than truths.

While certainly limiting in their scope, cellular automata models of disease transmission give researchers the traction needed to begin tackling the tough questions facing epidemiologists. How will factors such as latency and mortality effect propagation of a disease? In what way will quarantines, vaccinations and naturally resistant individuals change the course of an epidemic? Cellular automata provide a vital stepping stone to more realistic and complete models that may one day provide answers to these questions.

Crystal Formation

With nearest neighbor interactions dominate local dynamics, atoms or molecules in a crystalline lattice can be reasonably modelled as cellular automata[9]. Cells communicate through primarily through electromagnetic forces, though in some cases heat transfer and similar signals can play a significant role in crystal formation. A notable distinction between these crystalline cellular automata and many other cell networks used for computation and modelling is the domain over which the lattice is defined. This underlying structure can take any of a number of shapes depending on the physical nature of the atoms and molecules involved. An amethyst or diamond crystal, for instance, would lie in the typical rectangular domain. A water crystal (i.e., ice), however, would lie on a hexagonal domain. Each cell now has six neighbors to consider instead of the typical four. The ways in which these crystalline structures grow is directly dependent on both the lattice on which it will grow and the rule set governing the addition of new cells to the crystal.

One thoroughly researched cellular automata crystal-growth model is that of freezing water. The amazing variety, complexity and symmetry inherent in snowflakes was first earnestly research by Wilson Bentley. Indeed, even today, Bentley's collection[10] of snowflake images is second to none. Moreover, Bentley was one of the first physicists investigate the physical reactions causing the formation of snowflakes in a variety of different types of cloud.

In a natural continuation, Steven Wolfram defined a elegant and functional model for snowflake growth using cellular automata[11] . Defined over a hexagonal lattice, the flake will grow based on the simplest imaginable rule: if a single neighbor is on in the previous iteration, a cell will turn on. For convenience, we denote the number of active neighbors to be η.

[9]Wolfram, Steven *A New Kind of Science.* pgs. 371-373

[10]http://library.ssec.wisc.edu/bentley/copyright.html

[11]Wolfram, Steven. *A New Kind of Science.*

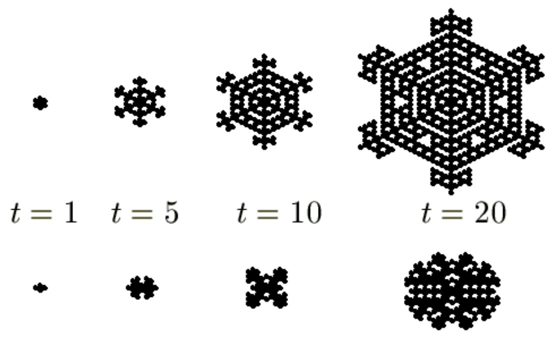

Figure 3: Above we see two examples of crystals grown on a hexagonal lattice. Black cells indicate areas of lattice that are participating in the crystal. The upper series features the classic "snowflake" simulation. Surprisingly, the rich structures observed come from the simplest of governing rules: if a single neighbor is part of the crystal, a cell will incorporate itself in to the crystal. That is, if a single neighbor is black, turn black. In the lower example, we see an equivalent simulation in which two neighbors must be in the crystal for the a cell to participate.

Thus, Wolfram's snowflake is governed by $\eta = 1$. In the spirit of crystal growth, all cells are state holding. That is to say, once a section of lattice is occupied with a chunk of frozen water, it will always contain this ice. In this way, the crystal grows out over time. We see can example of this type of growth in Figure 3. An analogous example with $\eta = 2$. We note the decreased structural complexity. Wolfram physically justifies his choice of the $\eta = 1$ governance. As water freezes, a fixed amount of heat is given off. Two or more of these heat quanta, he argues, will prohibit a nearby cells from freezing. Thus, only cells with a single neighbor present in the previous iteration may participate in the crystal.

Also of practical interest are crystals defined over a rectangular domain. One such crystal can be seen in Figure 4. While snowflakes display an elegant hexagonal symmetry, rectangular crystal have different, but equally intriguing properties. We can see in our example that the pattern created by the crystal

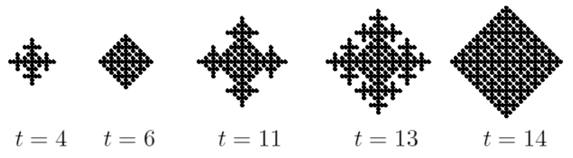

Figure 4: In this figure we see a crystal grown on a rectangular domain. Like their hexagonal counterparts, rectangular crystals have many interesting properties. For instance, in the example shown above we see that as the crystal grows, there is a certain periodicity in the forms the crystal will assume. That is, the crystal at time $t = 6$ is of the same functional form as at time $t = 14$. The crystal has simply scaled over time. Mathematically speaking, this result is extremely interesting as it indicates underlying periodic dynamics in crystalline cellular automata.

formation repeats over time. That is, the crystal assumes the same functional form at different points in time. In this particular case, the repeated form is a fully filled diamond. This periodic quality is apparent in a number of the unfeatured examples we have created. This result is intriguing for a number of reasons. First, the periodic nature of the growth sequence is quite mathematically remarkable. The presence of periodic dynamics in a system with such simple governing rules indicates a high degree of underlying complexity. Indeed, this periodicity creates a fractal structure that will govern the relation of one layer to the next. Second, the physical interpretation of this periodicity leads us to the concept that we can make "perfect" crystals of arbitrary size. Thus, any asymmetries observed are a result of stochastic or applied environmental factors. This may seem readily apparent but without theoretical confirmation, it would be very difficult to definitively state the configuration preferred by the crystal.

As with many cellular automata systems, the simplicity of the rule set and iteration scheme does not prohibit startlingly accurate models of natural phenomena. This is especially true for crystal growth, where physical dynamics are dictated in much the same way as cellular automata. By carefully considering the local physics governing crystal formation, we can define cellular automata rules that have a solid basis in accepted physics. Our result, then, is two-fold. We arrive at a physically accurate representation of the modelled system as well as gain general acceptance for cellular automata as efficient and effective modelling tools.

Pattern Formation

Patterns are present everywhere in nature. Many of the creatures on land and sea exhibit beautiful patterning on their fur, skin or shells. For whatever purpose, evolution gave these animals their patterns, and be it camouflage or for attracting prospective mates, the animals themselves have no control over their coloration.

It is theorized that perhaps each of the pigmentation cells is capable of choosing for itself what color it should be. In *The New Kind of Science*, Stephen Wolfram suggests[12] that perhaps these cells follow basic rules which govern their coloration. Some shell patterns are quite simple, stripes or spots, yet others develop very complicated patterns. These more complex patterns, as pictured in Figure 5, are very similar to patterns generated by cellular automata. To think that perhaps these shells are created in a similar fashion to cellular automata does not seem far from the truth.

Mollusc shells grow in a fashion similar to the growth of a fingernail - new material is created by a lip of soft tissue and grows outward. That is to say, a 'row' of cells are constructed at the same time, and perhaps may be colored in relation to their nearest neighbors. The cells in the soft tissue may inspect the nearby shell cells, and choose the color for the new piece of cell based on rules taking those nearby cells into account. One-dimensional cellular automata function in much the same way, and their outputs, when binary, are strikingly similar.

[12]Wolfram, Steven. *A New Kind of Science*

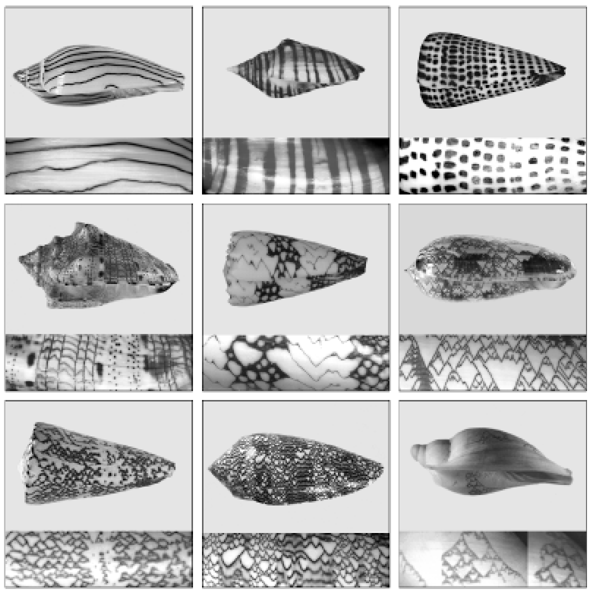

Figure 5: The patterns displayed on these seashells are not uncommon in nature, nor are they uncommon in one-dimensional cellular automata. Stephen Wolfram believes this is not a coincidence.

Genetic Algorithms

A way in which biology has inspired computing is through the use of Genetic Algorithms (GA). A Genetic Algorithm essentially mimics evolution by forcing a system adapt itself to a presented problem[13]. The basic operation of a GA is to take a sample population, evaluate their performance (or "fitness"), select pairs of high-ranking members to reproduce (via "crossover") and then apply random mutations. This cycle is repeated hundreds or thousands of times until there is little to no improvement in the fitness of the resulting population. The members of a population are often represented as binary strings, but they can also be indexes in a lookup table, items in an array or nearly any other data structure depending on how the problem is designed. During each cycle a fitness test is applied to each member (or "chromosome") to determine how well it has performed the task it for which is being evolved. These tests are the core of genetic algorithms. They guide the development of the populations and determine when a suitable solution has been found. An interesting application of GAs is to circuit design in a field called Evolvable Hardware. Until recently there was no easy way to evolve hardware, but with the proliferation of Field Programmable Gate Arrays (FPGAs) it is possible to rapidly test a multitude of hardware configurations, opening the door to genetic design of circuits. FPGAs are especially interesting because they provide essentially a hardware representation of cellular automata. Each logic cell in an FPGA is laid out in a grid and connected to four neighbors. By changing the internal truth table for the FPGA one is basically defining the logic inherent to a cell in a CA.

The idea of evolvable hardware was first demonstrated by Adrian Thompson in 1996 when he evolved a tone discriminator using an early version of an FPGA[14]. The design requirement for the circuit was to be able to tell the different between square wave signals of 1 kHz and 10 kHz by outputting +5V for one and 0v for the other. The problem was left purposefully vague to see what sort of solutions were arrived at. The circuit was constrained to only use a 10x10 array of logic cells in the FPGA with no external connections except the input and output pins. No clock was provided, forcing the circuit to evolve a continuous time solution, not a trivial problem even for an experienced designer. The circuit was evolved in the same basic method used by all GAs. Populations consisted of 1800 bit strings. The bit strings represented the interconnections between adjacent units by defining the truth table each logic cell would apply to the four inputs it received from its neighbors. This also determined what sort of logical computation each cell would perform.

The circuit as described was run for several thousand generations to generate a solution that met the design specifications. The best performing circuit from generation 5000 was chosen as the "final" circuit. This circuit made use of all 100 logic cells but

[13]Holland, John H (1975), "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor

[14]A. Thompson. "An evolved circuit, intrinsic in silicon, entwined with physics", 1997

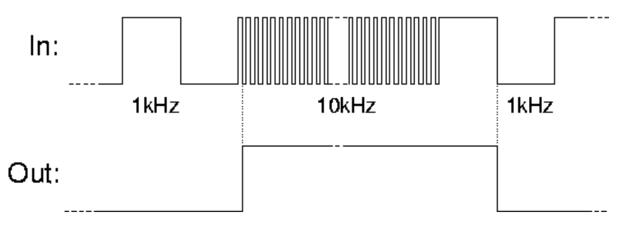

Figure 6: The actual input and output of the final circuit.

not all of them were necessary. Through iteration and experimentation most of the cells were eliminated as being unnecessary to final performance. This resulted in only 32 final cells needed to successfully discriminate between the two input signals. The experiment was successful in its attempts to evolve a circuit to perform this function. However this was not the most interesting result of the experiment. In the final circuit there were 5 cells that were not directly hooked into the path the signal took in the chip. However suppressing any of the cells degraded the performance of the system even though they did not directly affect the circuit.

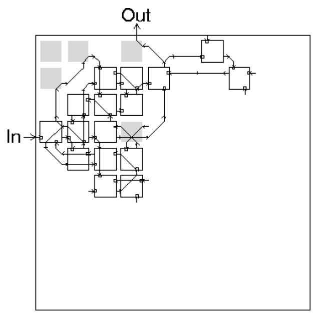

Figure 7: The final logic block configuration of the FPGA. Grey cells are those that cannot be suppressed but are not actually part of the signal path.

The only possible explanation for this behavior is that the GA made use of the inherent properties of the silicon on which the chip is printed, perhaps generating a magnetic field or exploiting some other subtle interaction to create its solution circuit. This is a solution that no human could have generated without extensive and precise modelling of every component in the chip down to the molecular level. In this way the GA can be a more efficient designer than a human by moving past the simplifications and assumptions a person must make in order to work in the domain of individual electrons. Extensive study of the circuit as generated has yielded many questions and fewer answers [15]. While a design of the circuit can be extracted from the chromosome of the final circuit, simulations and even CMOS implementations of this circuit are non functional. The subtle timing and logical loops present in the final circuit must make use of inherent characteristics of the chip on which it was developed and not more general principles. In addition, changing which one hundred logic cells on the FPGA are used or the temperature at which it operates can degrade the performance on a spectrum from slightly to completely. However the circuit can often recover within a few hundred generations to ideal operation under the new conditions, proving the robustness of the GA. Several GAs can even be run in parallel on different chips and with differing external conditions to apply evolutionary pressure on each generation to perform in a wide range of conditions[16]

Evolvable hardware and genetic algorithms also much promise for devising solutions on the fly to hardware failure from short circuits and breaks in a motor control circuit[17] to the disabling of three legs on a quadruped robot[18]. These examples make use of the same general format of algorithm as seen before in the frequency discriminator. They create chromosomes, run fitness tests, and perform crossover and mutation in a repeating cycle . From these and many other examples it can be seen that Genetic Algorithms as applied to hardware can be a very powerful way to develop specific and highly efficient solutions easily.

[15]A. Thompson and P. Layzell. "Unconventional Evolved Electronics", Communications of the ACM, vol. 42, 1999

[16]A. Thompson and P. Layzell and R. S. Zebulum. "Explorations in Design Space: Unconventional electronics design through artificial evolution", IEEE Trans. Evol. Comp, vol. 3, 1999.

[17]$http://ic.arc.nasa.gov/projects/eh2005/slides/gwaltney_eh05.pdf$

[18]$http://ic.arc.nasa.gov/projects/eh2005/slides/berenson/index.htm$

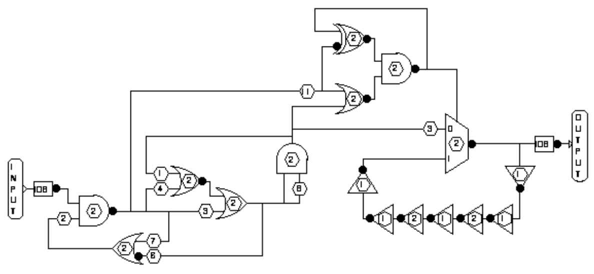

Figure 8: Drawing of the final circuit. This design works in the specific location on the specific FPGA it was designed on in continuous time and with varying levels of efficacy under other conditions.

Conclusion

Despite the relatively straightforward rules governing the cellular automata and biologically inspired systems discussed above, complex and emergent dynamics abound in each. Whether a flock of a hundred boids deciding on a common course, a crystal forming from a single, simple rule or a evolving circuit capitalizing on imperfections in a silicon substrate, biologically inspired computing shows incredible promise. In the years to come, we hope to see this amazing field grow and yield results that will help researchers not only understand natural phenomena but also also bring bio-inspired innovation to more unconventional applications.