Algorithmic Self-Assembly of Circuits

Michael deLorimier Alexandre Mathy Dustin Reishus Rolfe Schmidt
Bilal Shaw Li Chin Wong

August 24, 2002

Abstract

1 Introduction

The order we see in nature is self-assembled: atoms, crystals, cells, animals, solar systems, and galaxies are just a few examples. While we have been able to build phenomenal devices with controlled assembly techniques, we still cannot engineer machines at the scale of the ribosome. Can we understand the mechanism behind natural self-assembly and harness it to fabricate new devices? A systematic attack on this problem as been initiated [5, 1], and in this paper we use these techniques to explore how self-assembly could be used to fabricate nanoscale electrical circuits.

We start with a simple model, where molecules are thought of as two dimensional square tiles [Wang] having different glue types on north, east, south and west sides. These glues have strengths associated with them which will be important later on. Now tiles that have matching glues stick to each other. The model that we have considered is an irreversible one, which means that once tiles stick together with a certain bond strength they do not come apart.

In reality, molecular bonds are reversible. But some bonds are stronger than others, and if a molecule is attached in more than one place it is less likely to fall off. Analysis of the thermodynamics of these interactions shows that for some temperatures our model that "tiles with matching glues stick" is actually fairly accurate [7]. This is called the $T = 1$ model: the temperature T is such that only one matching bond is needed for two tiles to stick to each other. The behavior is different at other temperatures. As the temperature rises above $T = 1$, eventually it reaches a point where in equilibrium, a tile sticks to other tiles only if it can find two bonds (or one double strength bond). This gives us a new model of self assembly which seems more powerful than the first. Not surprisingly, it is called the $T = 2$ model.

In this paper our purpose is to assemble some interesting circuit with the $T = 2$ model of self assembly. One could imagine fabricating tiles with nano dimensions which have logical gates embedded on their surface. Then one need only program these gates in the appropriate way to assemble the tiles into meaningful circuit. For our presentation we chose to assemble the Fast Fourier Transform (FFT), but the ideas we use can be easily modified to assemble sorting networks, power-law crossbars, and various decoders.

Designing a tile system to assemble a network is a programming task, but the program is massively parallel and not at all like programs we are used to writing for machines with limited numbers of processing nodes and high context switch overhead. However, when writing a program for a multi-processor machine it is possible (and sometimes best) to just use one processor. Similarly, when designing a tile system we can make it emulate a "serial" program to simplify our task. We do this by taking advantage of the fact that every bounded CA rule with the Margolus neighborhood can easily translated into a $T = 2$ tile system. The glues in this tile system correspond to symbols in the CA rule, and tiles correspond to rules. When this tile system is seeded with "input data", it will assemble the trace of the CA's evolution on this input.

CA are easy to think about, and we take advantage of this to design a CA that produces the recursive shape we need for the FFT network. Once this is done, we translate the CA rules into a tile system that uniquely assembles our network.

The sequel proceeds as follows. In Section 2 we provide a brief overview of the uses and implementation of the Fourier transform. Section 3 describes how we designed CA rules to produce the recursive FFT shape, and follows with a discussion of the actual $T = 2$ tile system. Space does not allow a full presentation of the rules, but this is available online [4]. With the tile set in hand, we analyze the (in)tractability of implementing this system with DNA DX molecule tiles in Section 4. Finally, in Section 5 we analyze the error rates of systems like ours and propose a way to reduce the rate of failure do to tile misincorporation.

2 The Fast Fourier Transform

For all Abelian groups G, the Fourier Transform is the unitary change of basis on $L^2(G)$ that simultaneously diagonalizes all translation invariant operators. This transform plays a pivotal rôle in pure mathematics. For continuous groups, differential operators are translation invariant, and differential equations become algebra problems. Convolutions are just translation invariant "moving averages", and become pointwise multiplications. The applications are not limited to differential and integral equations: when $G = (\mathbb{R}^+, \cdot)$, and $\omega = \sum_{\mathbb{N}} \delta_n$ is the Dirac comb along the natural numbers, the Fourier transform of ω is just the Riemann zeta function. Understanding the zeroes of this function would allow us to place strong bounds on the distribution of the prime numbers.

As important as the Fourier transform is in pure Mathematics, most people who use it do so for very practical reasons: there is a fast divide-and-conquer algorithm, called the Fast Fourier Transform [FFT], for computing the transform. This makes it feasible to evaluate convolutions and solve many differential equations very quickly by transforming the problem to the Fourier domain, solving the easier problem, then performing the inverse FFT to bring the solution back. This approach is used widely, and in applications like signal processing speed is so crucial that is economical to implement the FFT in hardware.

How does the FFT work? On the group $G = \mathbb{Z}^N$ the Fourier transform of a function f can be written

$$\hat{f}(\xi) = \sum_{x \in \mathbb{Z}^N} f(x) \exp[-2\pi i x \xi / N]$$

Notice that

$$\hat{f}(\xi) = \hat{f}_{\text{even}}(\xi) + e^{-2\pi i \xi / N} \hat{f}_{\text{odd}}(\xi)$$

when $\xi < N/2$ and

$$\hat{f}(\xi) = \hat{f}_{\text{even}}(\xi - N/2) - e^{-2\pi i \xi / N} \hat{f}_{\text{odd}}(\xi - N/2)$$

when $\xi >= N/2$.

So we can evaluate this recursively. By the Master theorem [2], the running time of this algorithm is

$$T(N) = N + 2T(N/2) = \Theta(N \lg N)$$

which is much faster than the naive $\Theta(N^2)$ approach.

This algorithm can be implemented as a network of phase-shifters and adders. If we were given "magic boxes" that compute $\hat{f}_{\text{even}}$ and $\hat{f}_{\text{odd}}$, then it would be easy to build a network for the full Fourier transform (see Figure 1).

But we do not need magic- each of these boxes is simply implementing a smaller FFT, so we can continue building the network as seen in Figure 2.

There is one problem with this network as presented. The FFT requires us to keep splitting the array into even and odd subarrays, but our network splits it into high and low subarrays. In other words, we are checking the high bit of the array offset when we should be checking the low bit. We can fix this by performing a bit reversal permutation on the array before passing the array to the FFT network.

3 CA Rules for the FFT Network

To self-assemble a circuit that performs the fast fourier transform, we envision self-assembly from

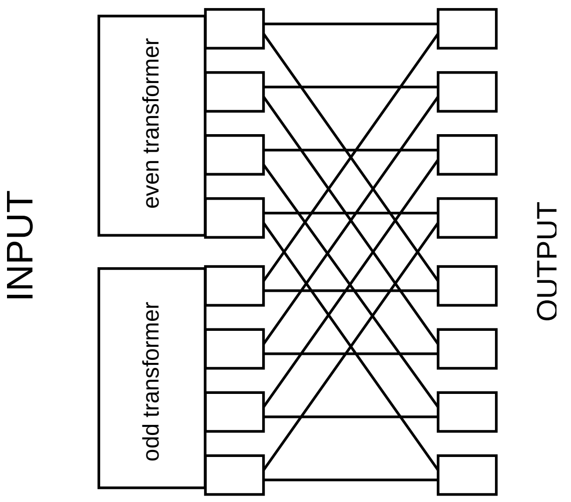

Figure 1: Implementing an FFT network with magic boxes

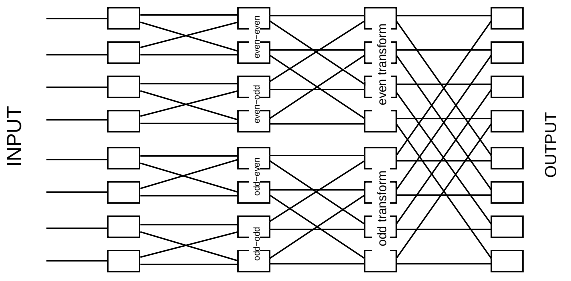

Figure 2: The FFT network

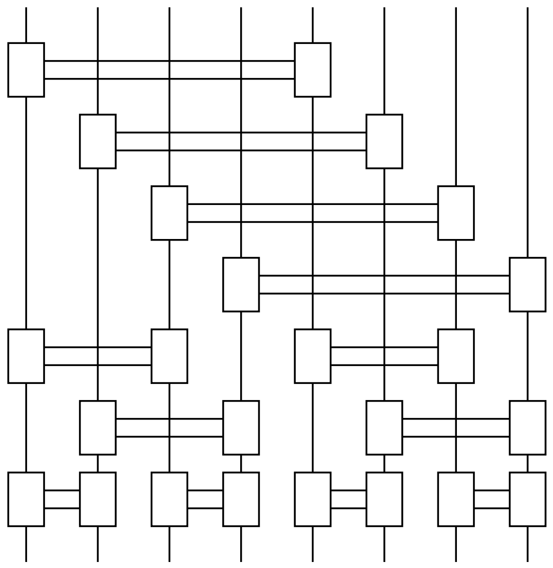

Figure 3: Self-Assembled FFT Network

a cellular automata perspective. Briefly, the T=2 model of self assembly can be thought of as a 1-dimensional cellular automata using the Margolus neighborhood. The specifics of how to transform 1D CA rules into a T=2 tile set will be covered in more detail in the next section. Thinking of self-assembly from a CA standpoint allows us to design self-assembled circuits in terms of particles and collisions, which are CA symbols and interactions between symbols, respectively.

An FFT network consists of many small logic blocks interconnected in a very specific way. The logic blocks perform mulitplications, and the interconnects tell them which signals to multiply. The structure of the circuit is highly self-similar; it is this recursive structure that led us to believe that the FFT network was a good candidate for production by self-assembly. In our design of the completed circuit, we have interconnects travelling horizontally and vertically through every tile. At certain locations along diagnals, the horizontal and vertical busses are connected at a logic block. See Figure 3 below for a schematic diagram of the logic blocks and interconnects.

Our self-assembled circuit consists of tiles that line up the interconnects and places the logic blocks in the correct positions. To locate the places where a logic block should be, we use a system of particles and collisions. As we mentioned before, the particles are symbols in a 1D CA using the Margolus neighborhood. The Margolus neighborhood is a way of implementing reversible CA rules. It consists of updating the CA in two phases: In the first phase, an even numbered cell looks at its neighbor to the left, an odd numbered cell looks at its neighbor to the right, and every cell updates based on its symbol and its neighbor's symbol. In the second phase, each cell looks at its other neighbor.

We created CA rules for four kinds of particle movement: stationary, left and right unit speed, and left moving half speed. Stationary particles are called μ, left moving particles are called λ, right moving particles are called ρ, and left moving half speed particles are called γ. The CA rules to implement stationary particles and particles that move with speed one are trivial. A half speed left moving particle's CA rules are slightly more complex. It completes one cycle every four phases of the CA and moves two cells per cycle. The first two phases it moves with speed one, and the next two phases it is stationary, after which it repeats this procedure. The movement must be two cells every four phases, because if it moved only one cell in two phases it would be in the wrong neighborhood and unable to move the following phase.

As you can see from the collision map in Figure 4, the FFT circuit has a recursive structure. At the top (time t_0), the only particles are the left and right boundry particles and a stationary particle in the middle. These immediatly eject ρ, λ, and γ particles. When the ρ and λ collide in the center, they create a μ and keep travelling. The λ collides with the left boundry, the ρ collides with the right boundry, and the γ collides with the μ at time t_1. This completes the first section of the FFT network. Here the recursion enters: The right and left boundries eject ρ and λ particles and the μ ejects ρ, λ and γ particles and serves as both the left boundry and the right boundry for the two halves of the circuit. This process continues until time $t_{\log(n)}$, when every particle is a μ. If logic blocks are placed on every ρ tile, the FFT network is complete.

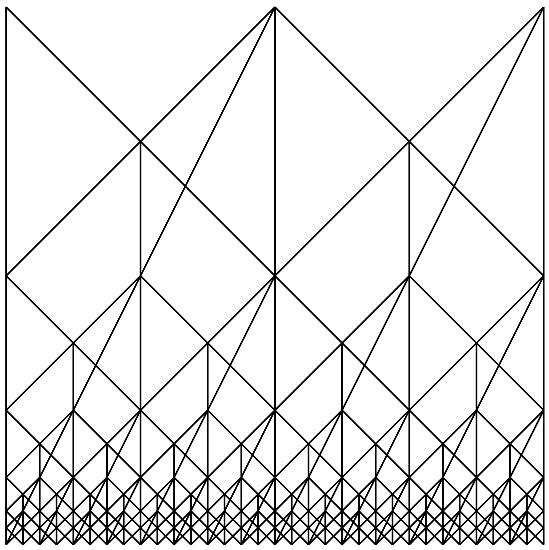
Figure 4: Collision Map

There are several difficulties when creating the rules for the CA with the Margolus neighborhood. First, the λ and ρ cannot start in the same phase. The neighborhood adds constraints so that the particles cannot always move in a particular direction; sometimes they must wait one phase before moving. An additional difficulty with creating CA rules for particle movement is that the number of CA symbols required typically grows as a power of the number of logical particles. This is compounded by the fact that the number of explicit rules can grow as a power of the number of rules.

This leads to a large number of explicit rules for a relatively small number of logical particles. This ends up being very bad for the implementation of any self-assembled circuit, because the each rule corresponds to exactly one tile. This is how the CA model, in which we designed our FFT network, can get translated to a $T = 2$ tile set. Along the boundries and on the input row, tiles with double bonds are used. For every rule in the CA, one tile is created in the tile set.

4 Implementation Issues

We explored the feasibility of experimentally building the self-assembled Fast Fourier Transform using DNA tiles. Suppose we use the double crossover (DX) molecule [6] to implement our one tile-one CA rule and one bond-one CA tile self-assembly system. The four sticky ends of the DX molecule will represent the north, south, east and west glues of the DNA tile. The south and west glue will represent the input and the north and east glue will represent the output.

To design the sequences of the unique sticky ends, we need to consider the optimal length and the corresponding sequence complexity. The number of tiles required to build the network, excluding the seed tiles, will be 73. If glue A exists in the north position, the complementary sequence will represent the A-south glue. A total of 2 different sequences will be required to represent this glue. Otherwise, if a glue occurs in both the north/south and east-west position, a total of 4 different sequences will be required to represent the glue. An analysis of the CA rules that showed that there are 45 symbols that occur exclusively in the north/south or east/ west position, and 24 symbols occur in both north/south and east/west position. Therefore a total of 69 different glues and the corresponding 69 complementary strands are required to build the experimental tile set. Assuming that we use the same sequence for all the non-sticky end portions of the DX molecule, we will be synthesizing 140 (2 + 69 + 69) different strands of DNA.

Assuming that we use sticky-ends which are 6nt long. The sequence space that can possibly be explored is 4096. However, in an attempt to control the hybridization between strands, the GC-content of each strand is kept constant at 50%. This reduces the sequence space to 1280. The sequence design should also attempt to avoid formation of undesired secondary structures and to maximize the interstrand interactions to form a stable double crossover structure. The 6nt sticky-ends should also not be complementary to the tile core.

Potential technical problems that will be faced include the problem of stoichiometry. According to the $T = 2$ model, concentration of different strands will be kept constant during self-assembly. However, as self-assembly occurs experimentally, the concentration of molecules that are being self-assembled decreases. This problem is even more pronounced if one sticky-end is used heavily compared to other sticky ends. Another example of a technical problem will be finding the optimal temperature for optimal self-assembly. Different sticky-ends might interact optimally at different temperatures. This problem is partially solved by keeping the GC-content constant.

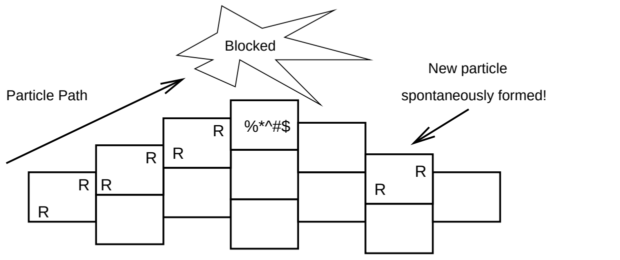

Figure 5: Errors that can occur in CA rule assembly

As shown in the analysis, implementing the self-assembly of the FFT circuit using DNA tiles is not an easy task, although not entirely impossible.

5 Error Correction

Our tile sets depend critically on particle scattering CA rules. Unfortunately, assembly errors do occur and the tile sets we have designed are particularly sensitive to these errors:

- If a particle tile gets misincorporated it will propogate and be locked in quickly.
- If a misincorporation occurs on a particle path it destroys the particle.

To see this more quantitatively, consider a tile system that implements a CA rule with one moving particle in open space that has a misincorporation rate of ϵ. It is straightforward to show that in an assembly of area A

- the expected number of spontaneous wires is $\epsilon A + o(\epsilon)$

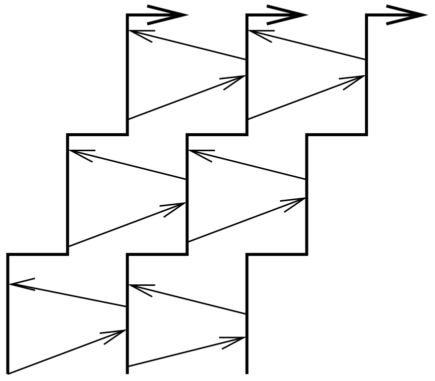

Figure 6: A self monitoring particle team

- the expected path length is $\frac{1-\epsilon}{\epsilon}$

Thus if 1 out of every tiles is a misincorporation (which would be an excellent error rate in a system with 73 tiles), we can only expect particle paths of length 99.

We would like to replace this with something more robust, so we propose using a "team" of particles that monitors and repairs itself. Figure 6 shows a schematic view of how such a system could work-three particles move in synchrony, sending signals to their neighbors. A set of CA rules that implements this strategy to repair all single errors in the signal path and prevent all single errors from creating a spontaneous particle is available at http://www-scf.usc.edu/ rolfesch/teamspirit.html. This set of rules fully repairs all spatially isolated errors within a constant number of time steps, but may fail to correct multiple errors. Thus it improves the expected particle path length to $\Omega(\epsilon^{-2})$ and reduces the number of spontaneous particles in an assembly of area A to $O(\epsilon^2)$.

However, this set of rules is enormous compared to the initial two free-particle rules, particularly if wildcards need to be expanded. We note that the rules were designed every symbol except for 0 has a unique left and right neighbor, so a misincorporation cannot lead to a valid rule application at the next step. This prevents erroneous tiles from being locked in quickly, and may allow us to implement our wildcard rules with tiles that simply have no glue on the wildcard side. These wildcard tiles will have no particular advantage over competing tiles when attaching to an erroneous assembly, but once attached they can be locked in by valid rule applications immediately.

In principle, this mechanism can be extended to teams of k particles that either simulate an error-correcting CA rule like GKL [3], or implement a voting protocol to determine whether the path should continue or terminate. The number of tiles required for such a system would be constant for the GKL rule, and polynomial in k for the voting rule. This sounds reasonable, but in practice the tile sets are unreasonably large. As we noted above, asymptotics can be falsely reassuring when designing DNA tiles for self-assembly. The constants are critical, and the absolute number of tiles must be small.

6 Conclusion

Our work has shown us that it is fairly simple to design tile systems that assemble common networks with $O(1)$ tiles. Unfortunately we also realize that it is very difficult to assemble common networks with a reasonable number of tiles. We relied heavily on CA rules to design our tile systems, and it would be interesting to know how much efficiency we give up with this approach. The CA rule approach also led us to a simple error correction scheme, which seems appealing in principle, but uses far too many tiles to be practical. In short, we have answered the querstions we set out to answer, but the standards we were using to measure success seem too liberal. We now realize that the real problems are much more challenging.

References

[1] L. Adleman. Toward a mathematical theory of self-assembly, January 2000.

[2] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2002.

[3] P. Gacs, G. Kurdiumov, and L. Levin. One-dimensional homogeneous media dissolving fi-

nite islands. *Problemy Peredachi Informatsii*, 14(3):223–226, 1978.

[4] http://www-scf.usc.edu/ rolfesch/cbsss.html.

[5] P. Rothemund and E. Winfree. Theprogram size complexity of self-assembled squares. In *STOC*, 2000.

[6] E. Winfree. Ph.d. Thesis, Caltech, 1998.

[7] E. Winfree. Simulations of computing by self-assembly. In *Proceedings of the 4th DIMACS Meeting on DNA Based Computers*, June 1998.