# Making DNA Self-Assembly Error-Proof: Attaining Small Growth Error Rates Through Embedded Information Redundancy

Saturnino Garcia and Alex Orailoğlu
Department of Computer Science and Engineering
University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92037
{sat, alex}@cs.ucsd.edu

***Abstract*—DNA self-assembly is emerging as the most promising technique for nanoscale self-assembly as it uses the simple, yet precise rules of DNA binding to create macroscale assemblies from nanoscale components. However, DNA self-assembly is also highly error-prone and requires the use of error-resilience techniques in order to unlock its potential. In this paper we propose a technique for error-resilience that is based on information redundancy but, in contrast to previous information redundancy schemes, can achieve much higher resilience to growth errors. By expanding the neighborhood from which redundant information is taken, we can extend the distance that errors are propagated and therefore increase the likelihood of the error being reversed. Given a growth error rate of $\epsilon$, we show that with a neighborhood of only 2 we can reduce the error rate to $\epsilon^{3.64}$ for arbitrary functions (as compared to $\epsilon^{2.33}$ previously achieved). Compared with spatial redundancy approaches, our technique allows for higher density nanostructures and has a greatly reduced assembly time.**

## I. Introduction

As CMOS has continued to scale further into the nanometer range, addressing reliability has become a central concern for circuit design. Process variation and signal integrity issues are among the problems that have arisen from the use of nanometer technologies. Addressing these problems has necessitated new approaches in the CAD community [1]. Unfortunately, the problem is even more pronounced in nanoelectronic devices such as carbon nanotubes that aim to continue Moore's Law past the end of CMOS scaling [2]. The extremely small feature sizes of nanoelectronic devices require a bottom-up self-assembly approach that is highly prone to defects. To address this issue, we will look into systematically improving the self-assembly process in order to reduce the number of defects that must be dealt with at later design stages.

DNA self-assembly is one of the most promising approaches that has been investigated. DNA self-assembly relies on the precise binding rules of DNA to assemble specific patterns. DNA molecules are created with complementary base-pair sequences so that they come together like a jigsaw puzzle to form larger structures. Le et al showed how self-assembled DNA may be used as a lattice upon which nanoelectronic devices may be precisely placed [3]. Using these techniques, it would be possible to form the nanoelectronic crossbar structure required by many proposed nanoelectronic architectures [4], [5].

The process of DNA self-assembly is not perfect though. It has been estimated that the error rate ($\epsilon$) in DNA self-assembly could be as high as 10% [6]. This high error rate has limited the construction of error-free structures to those of a very small size. There have been a number of works that have searched for techniques to limit the occurrence of errors without changing the basic DNA self-assembly process [6], [7]. While they have succeeded in reducing the error rate (some to $\epsilon^k$ for arbitrary $k$), the improvements in error rate have mostly come at the cost of greatly increasing both the size and the growth time of the assembly.

In this paper, we present a technique for error-resilience that can achieve a large reduction of growth errors. Like the work in [8], our method does *not* increase the size of the final DNA assembly. Our technique also outperforms previous techniques in that it forces more "double mismatches," thus further decreasing the possibility of error. Our technique relies upon adding redundant information into the connections between DNA tiles. This redundant data is propagated throughout a neighborhood in the DNA assembly and checked multiple times to ensure that the data is correct. In this way, a single error forces the appearance of multiple errors, an event that is much less likely to occur.

Our paper will proceed as follows. In Section II we introduce the concepts underlying algorithmic DNA self-assembly, also discussing previous work in dealing with errors. We present our technique, 2-ENR, in Section III before presenting simulation results in Section IV and concluding in Section V.

## II. Algorithmic DNA Self-Assembly

In order to model the process of DNA self-assembly, Winfree developed the abstract tile assembly model (aTAM) [9]. In the aTAM, structures are self-assembled from basic units called tiles. Figure 1a gives an example of one of these tiles. Each tile has four sides, two of which are considered inputs and two of which are outputs. In general, the outputs may be any function of the inputs. It is assumed that the tile

978-3-9810801-5-5/DATE09 © 2009 EDAA

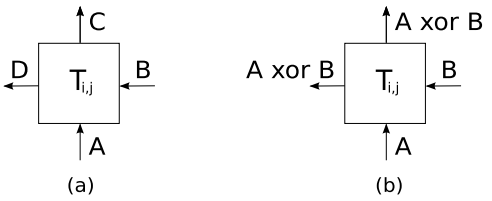


Fig. 1. (a) A basic tile. Outputs may be any function of the inputs. (b) Sierpinski tile. Outputs are the XOR of the inputs.

will never rotate so that the top will always be the top and so on. Each side has a bond type and strength associated with it. The bond type will determine which other tiles can attach to it. The bond strength (ranging from 0 to 2) determines how strong the attachment between two tiles will be. A tile attaches to a growing DNA crystal only when the bond strengths of its matching sides exceed 2. This is accomplished either by matching two strength 1 inputs or one strength 2 input. The growth of the DNA crystal is started by a special tile called the seed tile. This seed tile has two strength 2 outputs where boundary tile can attach. Boundary tiles form the periphery of the assembly. All other tiles are referred to as rule tiles. One example of a widely used rule tile is the Sierpinski tile shown in Figure 1b. In this tile, both outputs are the XOR of the inputs.

Unfortunately, the aTAM is an idealized model that does not take into account the physical environment in which a crystal will be built. To address this shortcoming, Winfree introduced the kinetic tile assembly model (kTAM) [9]. In the kTAM, any tile may attach at an open spot in the crystal. This introduces the possibility of a *mismatch* between tiles when the bond types of adjacent edges do not match. The occurrence of a mismatch during assembly is referred to as a *growth error*. The presence of only a single growth error will lead to the rest of the assembly being incorrect because all growth will happen relative to this erroneous tile. In the kTAM, tiles can also detach from the DNA crystal at a rate ($r_{off,b}$) that depends on the number of matching sides, $b$. Luckily, this makes it much more likely that an incorrect tile will detach. Unfortunately, experimental results have shown that the error rate for DNA self-assembly is somewhere between 1 and 10% [6]. At an error rate of 10%, only a handful of tiles may be reliably assembled before irreversible errors occur. In order to make DNA self-assembly viable, a method for handling growth errors is needed.

In order to deal with growth errors, two approaches may be taken: error recovery or error avoidance. In error recovery techniques such as the one presented in [10], errors are "tagged" so that they may be undone later. Unfortunately, undoing these errors requires external processes to be applied to the growing crystal and are therefore not ideal. For error avoidance, several approaches have been proposed [6], [7]. These approaches rely on replacing a tile by a $k \times k$ supertile. If an error occurs in one part of the supertile, $k$ additional errors are forced and the error rate is reduced to $\epsilon^k$. The drawback of these techniques is that they use spatial redundancy to reduce

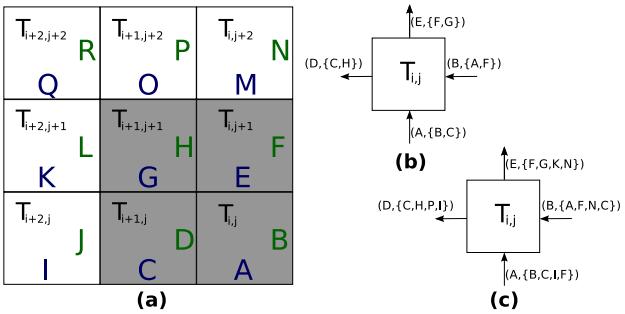


Fig. 2. (a) Neighborhood for redundancy used for (b) A tile in [8] and (c) A tile in 2-ENR.

errors. Using these techniques, the overall size of the assembly is increased by $k^2$-fold. This results in a much longer time needed to assemble and a reduction in the minimum feature size that is available.

Another approach, proposed by Sahu and Reif [8] uses information redundancy to avoid errors and therefore does not increase the final size or building time of the assembly. The basic idea behind their approach is to precompute the identity of neighboring tiles and pass those values along. If a mismatch occurs, then the error will propagate to the neighboring tiles where it can be identified and reversed. However, their approach was limited in the error rate it could achieve. Our technique, which is also based on information redundancy is able to achieve a much larger reduction in the error rate. In the following section, we will describe our technique as well as how it differs from the approach of Sahu and Reif.

## III. Extended Neighborhood Redundancy

As we alluded to earlier, the appearance of even a single growth error will prove disastrous. To understand why this is the case, we can refer to the basic tile in Figure 1. If a mismatch occurs in either input, the output may also change and affect the tiles that would use that output as their input. There is simply no method to check whether one's inputs are the correct ones. To remedy this situation, we may add redundancy to tile edges so that two versions of the same information may be compared. In order to perform this comparison both versions must be available at a single tile and be present on different inputs to that tile. Our proposed technique, *Extended Neighborhood Redundancy* (ENR), builds upon the redundancy technique of Sahu and Reif [8] by providing a general model for extracting and using redundant information. This redundant information is gathered from an expanded neighborhood around a tile and can further reduce the occurrence of growth errors. Before we discuss our technique, we will briefly review the technique introduced by Sahu and Reif in [8].

Figure 2 shows both the tile setup (part b) and the neighborhood from which redundant information is extracted (shaded region of part a). Note that the labels used in this figure

are simply variable names and are not unique bond types. Compared to the file in Figure 1, there is also additional information on each tile edge. Because each input shows up on exactly one other edge, a mismatch on that input will force one additional mismatch. As a result, the growth error rate is reduced from $\epsilon$ to $\epsilon^2$. The downside of Sahu and Reif's approach is that it increases the number of bond types and tiles. This is a result of needing a different tile for each possible set of redundant input values. However, other proposed techniques [6], [7] also require an increase in the number of tiles. Furthermore, it is not clear that a small increase in the number of tiles (in this case, from 4 to 16) is harmful. Restricting themselves to the immediate neighborhood of a block, Sahu and Reif showed that, outside of a very restricted class of functions, no further improvement in growth error rate is possible.

### A. Expansion to 2-Neighbors (2-ENR)

We do not need to restrict ourselves to the neighborhood presented in the shaded region of Figure 2a. In this section, we will show what benefits are to be had from using redundant information from blocks that are two away (the 2-ENR case).

Figure 2 shows both the extended neighborhood (all blocks in part a) as well as a detailed view of the inputs to tile $T_{i,j}$ (part c) of the 2-ENR setup. Because we are dealing with a superset of those in the shaded region, we can include all the same redundant information in our setup. There are two main differences between parts b and c of Figures 2: the inclusion of two new inputs ($I$ and $N$) as well as the addition of new redundant copies of $C$ and $F$. Redundant copies of $I$ are available at $T_{i+1,j}$ (as the bottom $C$) and $T_{i+2,j}$ (as the bottom $A$), allowing us to check for correctness at two locations. With the addition of $C$ (similarly, $F$), we can now check for correctness at $T_{i,j}$ and $T_{i+1,j}$. We should note that in Sahu and Reif's construction, the extra copies of $C$ and $F$ are not possible because they require the presence of $I$ and $N$. Similarly to Sahu and Reif's technique, the cost of this added redundancy is in the increased number of tiles. As we mentioned earlier though, a moderate increase in the number of tiles may not be harmful.

Having now seen the setup of the tiles, we will analyze the scheme to determine the overall growth error rate. In the bottom edge of $T_{i,j}$, we have the following data in common with the previous approach: $A$, $B$, and $C$. The analysis of mismatch on $A$ and $B$ is the same as previously described (i.e. $\epsilon_A = \epsilon_B = 2$). The analysis of a mismatch on $C$ is similar but with one important difference: in our 2-ENR setup, $C$ is included in the right edge. As a result, a mismatch in $C$ will result not only in a further mismatch of an edge in $T_{i+1,j}$ but also a mismatch on the right edge of $T_{i,j}$. We therefore have 2 additional mismatches forced by a mismatch on $C$ and an error rate of $\epsilon_C = \epsilon^3$. The $F$ input is similar to that of $C$ and therefore has an error rate of $\epsilon_F = \epsilon^3$.

We are now left to analyze the case of a mismatch on $I$. Similar to the $C$ case, because $I$ is part of the redundant part of the left edge, a mismatch will propagate to block $T_{i+1,j}$ and cause one additional mismatch. However, the $I$ on the left edge of $T_{i,j}$ corresponds to the $C$ value of $T_{i+1,j}$. Unfortunately, there is no guarantee that the error will propagate further. If it occurs that the match happens on the bottom of $T_{i+1,j}$, then the correct value of $C$ will go to $T_{i+2,j}$ and the error will not propagate. Conversely, a match on the right side of $T_{i+1,j}$ will propagate the error. Figure 3 illustrates these two options. Because both cases are equally likely, half the time there are three mismatches and half the time there are two. This leads to an error rate of $\epsilon^{2.5}$ for an error on $I$ (similarly for $N$).

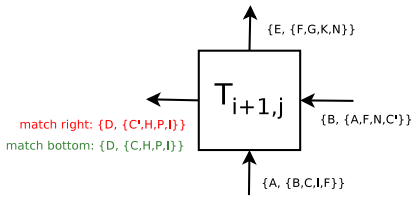


Fig. 3. Propagation of error on $I$ in $T_{i,j}$.

Before our analysis is complete, we must take into account one more issue. Once again we will examine Figure 3, which shows the propagation option of an error on $I$ in $T_{i,j}$. By design, only the right side of the tile includes $N$ while only the bottom contains $I$. Depending on which side of the tile is matched, either $N$ (if the bottom is matched) or $I$ (right match) will be an unbound variable (i.e. it can take on any value). In a binary tile set, only 50% of the time will this unbound variable be correct. In the case where it is incorrect, a new error is introduced which will now be propagated on. Luckily, this is good news as the increased number of mismatches will further drive down our error rate.[1]

To quantify the impact of unbound variables, we will determine the average number of additional errors introduced. At each step of propagation, there is a 50% chance of introducing a new error. With 2-ENR, the average distance an error on $I$ propagates is 1.5; as such, the chance of introducing a new error before propagation ends is 62.5%. From this, it follows that the probability of introducing $m$ new errors is $0.625^m$. For a mismatch on $C$, the case is similar except there is only a 50% chance of generating the first new error so the probability of $m$ errors becomes $0.5 * 0.625^{m-1}$. From these probabilities we find that, on average, 1.6 additional mismatches are forced for each mismatch on $C$ and 2.5 additional mismatches for $I$. We can now calculate the overall growth error rate: $\epsilon_{2-ENR} = \frac{1}{5}(2 * 2 + 2 * (3 + 1.6) + (2.5 + 2.5)) = 3.64$.

Our technique has another important benefit: it increases the rate of double mismatches where both inputs to a tile have a mismatch. Double mismatches are important because they have an exponentially larger probability of being corrected. In our setup, 80% ($A$,$B$,$C$, and $F$ but not $I$) will be double mismatches while only 66% ($A$ and $B$ but not $C$) of Sahu and Reif's will cause a double mismatch.

[1] [8] did not consider this in their analysis. Taking this into account for their scheme, they would achieve an error rate of $\epsilon^{2.33}$ instead of the reported $\epsilon^2$.

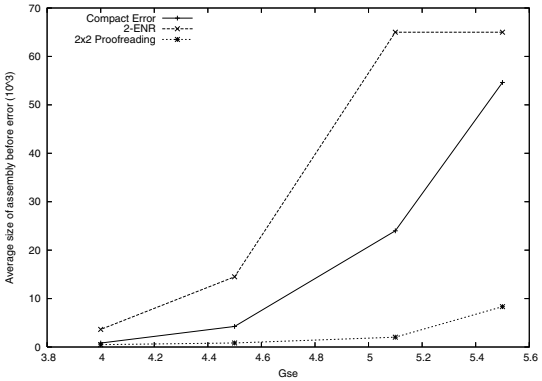


Fig. 4. $G_{se}$ vs average error-free aggregate size for various error-resilience techniques.

## IV. Simulation Results

In order to test the effectiveness of our technique, we employed the xgrow simulator [11]. In xgrow we can set the parameters $G_{mc}$ and $G_{se}$ to carefully control the simulation. $G_{mc}$ is a unitless parameter that corresponds to the concentration of tiles and dictates the growth rate of the assembly. Reducing $G_{mc}$ increases the rate of growth in the assembly but also negatively impacts the error rate. $G_{se}$ is a parameter relating to the free-energy required to break a strength 1 bond and therefore plays an important role in setting the error rate of the system. Our simulations made use of the Sierpinski tile set, a popular binary tile set (see Fig. 1b), so that we could compare it to the results of previous techniques. As part of our experimental setup, we aimed to assemble a 256x256 (approximately 65,000) tile aggregate. As suggested in [6], we have set the border tile concentration to half that of the rule tiles in order to discourage rampant border-growth and minimize the number of facet-errors that occur while not affecting the growth error rate.

Figure 4 shows the average size of assembly that can be grown before permanent errors are introduced. We varied the value of $G_{se}$ in order to measure several error rates. As was noted in [6], the optimal error rate for growth occurs when $G_{mc} \approx 2G_{se}$ so we followed this rule in order to obtain a value for $G_{mc}$ for a given $G_{se}$. At this ratio of $G_{mc}$ the error rate ($\epsilon$) is approximately $2e^{-G_{se}}$. For each value of $G_{se}$, we measured the average (over several runs of the simulation) size of the aggregate before a permanent error occurred. In addition to our 2-ENR technique, we also simulated Sahu and Reif's technique [8] as well as the 2x2 proofreading technique in [6]. Note that because of the $k^2$ increase in size in proofreading techniques, the 2x2 technique only produces $\frac{1}{4}$ of the number of original tiles for any given aggregate size. To account for this expansion, we normalize the simulation results of the 2x2 proofreading technique by dividing by 4.

As we can see from the results, at $G_{se}$ = 5.1 ($\epsilon \approx 0.012$), the 2-ENR technique can reliably build the 256x256 aggregate while the compact error-resilient technique in [8] fails to build half the assembly before encountering an error. For smaller $G_{se}$, the rate of growth is faster and the error rate increases. For the case of $G_{se}$ = 4.5 ($\epsilon \approx 0.0222$), the size of aggregate we can reliably build with 2-ENR drops to around 15K while the other techniques fail at less than 4K. While the size of assembly is limited at smaller values of $G_{se}$, because $G_{mc}$ is proportional to $G_{se}$, the speed of assembly at smaller values of $G_{se}$ is faster. This opens the possibility of quickly assembling crossbars of sufficient size to create future nanoarchitectures.

## V. Conclusion

In this paper, we have presented a new technique, *Extended Neighborhood Redundancy*, for error-resilience in DNA self-assembly. Our technique uses information redundancy to provide error checking of a tile's output. Because it does not require tile expansion as spatial redundancy based techniques do, it does not increase the overall size of the final DNA assembly.

As we have shown with our analysis, we can achieve a reduction in the growth error rate from $\epsilon$ to $\epsilon^{3.64}$ for arbitrary boolean functions. The previous error-resilience technique based on information redundancy [8] was only able to achieve a reduction to $\epsilon^{2.33}$. Simulation results have shown that our technique outperforms previous techniques and allows for reliable construction of aggregates of size at least 256x256 when the growth error rate is approximately 1.3%. Compared with the previous compact error-resilience technique in [8], 2-ENR would be able to assemble nanoscale crossbars that are twice as large. Conversely, where smaller assemblies are needed our technique allows much faster assembly, reducing the cost of manufacturing nanoelectronic circuits.

## References

[1] J. Kong, "CAD for nanometer silicon design challenges and success," *IEEE Transactions on VLSI Systems*, vol. 12, no. 11, pp. 1132–1147, 2004.

[2] M. Mishra and S. C. Goldstein, "Defect tolerance at the end of the roadmap," *ITC '03: Proceedings of the International Test Conference*, pp. 1201–1210, 2003.

[3] J. Le, Y. Pinto, N. Seeman, K. Musier-Forsyth, T. Taton, and R. Kiehl, "DNA-Templated self-assembly of metallic nanocomponent arrays on a surface," *Nano Letters*, vol. 4, no. 12, pp. 2343–2347, 2004.

[4] A. DeHon, "Array-based architecture for FET-based, nanoscale electronics," *IEEE Transactions on Nanotechnology*, vol. 2, no. 1, pp. 23–32, Mar. 2003.

[5] J. P. Patwardhan, V. Johri, C. Dwyer, and A. R. Lebeck, "A defect tolerant self-organizing nanoscale SIMD architecture," in *ASPLOS-XII: Proceedings of the 12th international conference on Architectural Support for Programming Languages and Operating Systems*, 2006, pp. 241–251.

[6] E. Winfree and R. Bekbolatov, "Proofreading tile sets: Error correction for algorithmic self-assembly," *DNA9: 9th International Workshop on DNA Based Computers*, 2004.

[7] H. Chen and A. Goel, "Error free self-assembly using error prone tiles," *DNA10: 10th International Workshop on DNA Based Computers*, 2005.

[8] S. Sahu and J. H. Reif, "Capabilities and limits of compact error resilience methods for algorithmic self-assembly in two and three dimensions," *DNA12: 12th International Meeting on DNA Based Computers*, pp. 5–9, 2006.

[9] E. Winfree, "Simulations of computing by self-assembly," *DIMACS: DNA-Based Computers*, 1998.

[10] S. Frechette and F. Lombardi, "Error detection/correction in DNA algorithmic self-assembly," in *DATE '08: Proceedings of the Conference on Design, Automation and Test in Europe*, 2008, pp. 1079–1082.

[11] E. Winfree, "The xgrow simulator," http://dna.caltech.edu/Xgrow/.