Rational Design of DNA Sequences with Non-Orthogonal Binding Interactions

Joseph Don Berleant[1]
Department of Biological Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

Molecular computation involving promiscuous, or non-orthogonal, binding interactions between system components is found commonly in natural biological systems, as well as some proposed human-made molecular computers. Such systems are characterized by the fact that each computational unit, such as a domain within a DNA strand, may bind to several different partners with distinct, prescribed binding strengths. Unfortunately, implementing systems of molecular computation that incorporate non-orthogonal binding is difficult, because researchers lack a robust, general-purpose method for designing molecules with this type of behavior. In this work, we describe and demonstrate a process for the rational design of DNA sequences with prescribed non-orthogonal binding behavior. This process makes use of a model that represents large sets of non-orthogonal DNA sequences using fixed-length binary strings, and estimates the differential binding affinity between pairs of sequences through the Hamming distance between their corresponding binary strings. The real-world applicability of this model is supported by simulations and some experimental data. We then select two previously described systems of molecular computation involving non-orthogonal interactions, and apply our sequence design process to implement them using DNA strand displacement. Our simulated results on these two systems demonstrate both digital and analog computation. We hope that this work motivates the development and implementation of new computational paradigms based on non-orthogonal binding.

2012 ACM Subject Classification Hardware → Emerging architectures; Hardware → Biology-related information processing

Keywords and phrases DNA sequence design, binding networks, promiscuous binding, non-orthogonal binding, isometric graph embeddings

Digital Object Identifier 10.4230/LIPIcs.DNA.29.4

Funding Research supported in part by a National Science Foundation Graduate Research Fellowship (Grant No. 1122374). Research supported in part by the Office of Naval Research (N00014-21-1-4013), the Army Research Office (ICB Subaward W911NF-19-2-0026), and the National Science Foundation (CBET-1729397, OAC-1940231, CCF-1956054).

1 Introduction

The vast majority of prior work with DNA-based molecular computation has dealt with the design and use of orthogonal DNA domains, which are intended to bind only to their perfect complements and to exhibit minimal cross-talk interactions with other domains in a system. This eases the design and analysis of large DNA-based molecular circuits, because it reduces the number of interactions that must be considered, allowing the scalable implementation of circuits with many more components [14, 20, 23]. Comparatively little research has attempted to solve the problem of designing non-orthogonal DNA sequences, that is, sets of DNA domains such that each sequence can bind to several partners with prescribed binding affinities. This gap in our abilities exists despite several well-documented examples of naturally occurring biological networks that make use of non-orthogonal binding

[1] Corresponding author. E-mail address: jberlean@mit.edu

behavior [1,15], computing problems that would benefit from non-orthogonal sequence design like similarity search in large databases [2,17], and nanostructure fabrication strategies that rely on promiscuous binding between structural subunits [16,30].

Very recently, some researchers have attempted to use machine learning for the design of non-orthogonal sequence designs, for example for DNA memory storage [3,27], while others have used exhaustive searches or evolutionary approaches to design non-orthogonal DNA sequences for digital logic and analog computing circuits [18]. While promising, these attempts have been characterized by design inaccuracies and, in some cases, the need for extensive experimental trial and error. Further, some preliminary evidence suggests that computational predictions of strand binding thermodynamics may be less accurate for sequences that are highly non-complementary, although additional data to support this observation is warranted [18]. Currently, the field of molecular computing lacks a general-purpose and efficient method for non-orthogonal DNA sequence design, capable of handling the variety of potential use cases in DNA-based molecular circuits, nanostructure fabrication, and data storage.

In this work, we describe a method for the rational design of non-orthogonal DNA sequences, which is based on two main contributions: the identification of a subset of DNA sequence space for which a simple model accurately predicts binding affinities between pairs of DNA sequences, and the application of a process called isometric graph embedding to the sequence design process. In Sections 3 and 4, we demonstrate our design process in the context of two previously described molecular computation systems involving non-orthogonal interactions [1,18]. Through simulations of these systems, we demonstrate both digital and analog computation using our sequence designs.[2]

2 Rational design of non-orthogonal DNA domains

Our ultimate goal will be to design for the differential binding affinities between pairs of DNA sequences, that is, we will consider the quantities $\Delta G(x, y^*) - \Delta G(w, z^*)$ for DNA sequences x, y, w, z. In Section 2.1, we define a model for the binding affinity between sequences, which will be applicable to a well-defined subset of the overall DNA sequence space. In Section 2.2, we show how this model enables rational design of DNA sequences via isometric graph embeddings.

2.1 Model of binding between non-orthogonal domains

In general, accurate estimation the binding affinity between two arbitrary DNA strands requires involved computation, and our goal is not to address this task for any two DNA strands. Instead, we first focus on defining a subset D of sequence space with properties that are amenable to efficient and accurate DNA sequence design for prescribed binding affinities. The main idea behind defining D will be to identify a set of substitution mutations that may be applied in any combination to some sequence, and whose cumulative effect on binding affinity is approximately additive. This will justify the model that we introduce at the end of this subsection.

Let $\mathcal{D}_n = \{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}^n$ be the set of all n-nt DNA sequences using only canonical nucleotides, where all sequences are written from 5′ to 3′. For an n-nt DNA sequence $x \in \mathcal{D}_n$, its perfect complement is denoted x^*. The nucleotides are referenced by subscripts, so that

[2] This work incorporates material from my doctoral thesis [4], which describes the sequence design process (Section 2) and one of the analog computing gates (part of Section 3).

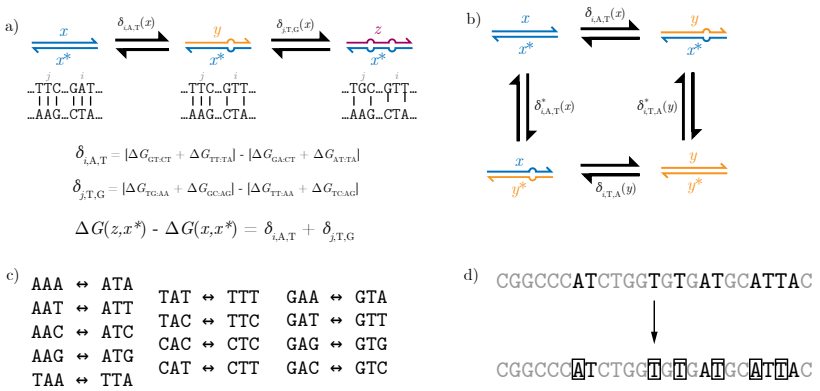

Figure 1 a) The binding affinity of a DNA duplex can be estimated by summing the contributions of each pair adjacent of base pairs. A single point substitution affects the contributions from two of these terms, reflecting the decreased favorability of the two stacking bonds next to a mismatch. A subsequent substitution at a nonadjacent nucleotide will affect two additional terms in the sum. However, under the nearest neighbor model, its effect will be independent of the presence or absence of the first substitution. b) To aid design, we proposed restricting the substitutions to those for which various metrics of its effect on binding affinity agree. For example, mutating the top strand ($\delta_{i,A,T}(x)$) or bottom strand ($\delta^*_{i,A,T}(x)$) should have a similar effect. c) Because the effect of a substitution is determined only by local interactions, we can enumerate all combinations of nucleotides that satisfy our constraints. This makes it very easy to identify sites on a candidate DNA sequence that may be mutated. d) Candidate substitution sites on a DNA site must be filtered to satisfy non-adjacency and interiority requirements.

$x_i \in \{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}$ is the i-th nucleotide of x. For $1 \leq i \leq n$ and $a, a' \in \{\mathrm{A}, \mathrm{T}, \mathrm{C}, \mathrm{G}\}$, we define a *substitution* $\sigma_{i,a,a'} : \mathcal{D}_n \to \mathcal{D}_n$ as a function such that $\sigma_{i,a,a'}(x)$ is the sequence generated by substituting the nucleotide a at position i in x with the nucleotide a'.

Fix an $x \in \mathcal{D}_n$, and consider the binding affinity of x to x^*. Under the nearest-neighbor model for DNA strand binding [22], this binding affinity may be estimated from the additive contributions of all pairs of neighboring base pairs in the duplex.[3] For a substitution $\sigma_{i,a,a'}$, $1 < i < n$ and $x_i = a$, let $y = \sigma_{i,a,a'}(x)$. Note that the constraint $1 < i < n$ implies that the substitution occurs at an interior position of x. Under the nearest-neighbor model, the binding affinity of y to x^* differs from that of x to x^* due to the contributions of exactly two affected pairs of adjacent base pairs (Figure 1a).[4] With these assumptions the change in binding affinity $\Delta G(y, x^*) - \Delta G(x, x^*)$ depends only on the values of a, a', x_{i-1}, and x_{i+1}. We will write $\delta_{i,a,a'}(x) := \Delta G(y, x^*) - \Delta G(x, x^*)$, the change in binding affinity from mutating x and leaving x^* unchanged. The corresponding change in binding affinity with x^* mutated and x unchanged will be written $\delta^*_{i,a,a'}(x) := \Delta G(x, y^*) - \Delta G(x, x^*)$. Note that $\delta^*_{i,a,a'}(x) = \delta_{n-i+1,a^*,a'^*}(x^*)$.

For a second substitution $\sigma_{j,b,b'}$, $1 < j < n$ and $x_j = b$, let $z = \sigma_{j,b,b'}(y)$. $\delta_{j,b,b'}(y) = \Delta G(z, y^*) - \Delta G(y, y^*)$ is determined only by the two affected pairs of base pairs. Furthermore, if $|i - j| > 1$, then the base pairs affected by $\sigma_{i,a,a'}$ are disjoint from those affected by $\sigma_{j,b,b'}$, and we may estimate $\Delta G(z, x^*) - \Delta G(x, x^*) = \delta_{i,a,a'} + \delta_{j,b,b'}$. More generally, given a set

[3] This model is biochemically justified by the fact that the principal contributor to binding affinity is the formation of a π-π stacking bond between two adjacent base pairs, and the contribution of each so-called "stack" is determined by the identities of the neighboring nucleotides [22].

[4] This calculation assumes that the substitution occurs in the interior of the sequence, and that "shifted" binding conformations of y to x^* may be neglected. This is likely valid when shifted conformations of x to x^* are unlikely, and the number of substitutions applied is small.

of substitutions occurring at non-adjacent interior positions within x, the effect on binding affinity of applying any subset of these substitutions is the sum of the effects of applying each substitution individually (Figure 1a).

We are now prepared to construct a subset of sequence space $D \in \mathcal{D}_n$. Given $x \in \mathcal{D}_n$, we propose constructing D based on a set of substitutions S satisfying the following requirements:

1. (interiority) For each $\sigma_{i,a,a'} \in S$, $k+1 \leq i \leq n-k$ and $x_i = a \neq a'$.
2. (non-adjacency) For any two $\sigma_{i,a,a'}, \sigma_{j,b,b'} \in S$, $|i-j| > l \geq 1$.
3. (symmetry) For each $\sigma_{i,a,a'} \in S$, $y = \sigma_{i,a,a'}(x)$, the four values $\delta_{i,a,a'}(x)$, $\delta_{i,a',a}(y)$, $\delta^*_{i,a,a'}(x)$, $\delta^*_{i,a',a}(y)$ are within the range $(\delta_{\min}, \delta_{\max})$.

For the interiority requirement, we choose $k = 2$ because of the observation that substitutions at terminal or penultimate strand positions have different effects on binding affinity [19]. For other the non-adjacency requirement, we use $l = 1$ as the minimal requirement for achieving additivity in the effects of each substitution. The symmetry requirement is included to aid subsequent sequence design, and its purpose will become clear by the end of this subsection. Informally, it implies that the effect of a mutation is consistent across different measures of its effect on binding affinity (Figure 1b). Because the effect of mutating a nucleotide depends only on that nucleotide and its neighbors, there are exactly 192 possible substitutions, and we may enumerate the list of those that satisfy condition (3) (Figure 1c). We use the range $(11.0, 15.3)$ kJ/mol, with all binding affinities estimated by published nearest neighbor parameters [22]. In principle, greater design robustness could be achieved by increasing k, increasing l, or decreasing the range $(\delta_{\min}, \delta_{\max})$, at the cost of reducing the size of the set S.

Given a set of substitutions S (Figure 1d), let D be the set of sequences generated by applying a subset $S' \in S$ to x, and D^* be the set of perfect complements to sequences in D. Each element of D is associated with a binary string of length $m := |S|$ as follows. Number the elements of S from 1 to m. Assume without loss of generality that the j-th element of S is a substitution at position i of x. Define $f : D \to \{0,1\}^m$ such that, for $y \in D$, $f(y)$ is the binary string with a j-th bit of 0 if $x_i = y_i$ and 1 otherwise. In other words, a bit of $f(y)$ is 1 if and only if the corresponding substitution in S was applied to generate y from x.

We may now model the differential binding affinity between two pairs of sequences taken from D and D^*. For sequences y, z, u, v from D, we model the differential binding affinity as

$$\Delta G(y, z^*) - \Delta G(u, v^*) \approx \delta_{\text{mut}} \times [d_H(f(y), f(z)) - d_H(f(u), f(v))] \tag{1}$$

where δ_{mut} is a proportionality constant approximating the change in binding affinity due to a single mutation, and d_H is the Hamming distance between two binary strings. This approximation is justified by the fact that the set of substitutions S was chosen so that the effect of each mutation would be approximately additive, and the number of substitutions that differentiates x from y equals the Hamming distance between $f(x)$ and $f(y)$. Note that Equation (1) relates the Hamming distance between $f(x)$ and $f(y)$ to the binding affinities both of x to y^* and of y to x^*, which is ensured by the symmetry requirement.

In this way, we model the binding affinity of any sequence of D with any sequence of D^*. In principle, the number of sequences may be very large (e.g., later we find sets S of 9 substitutions within 25-nt sequences, which generate sets D of $2^9 = 512$ sequences). However, the distribution of binding affinities is constrained by the corresponding distribution of Hamming distances, so only a subset of these sequences would likely be used for a particular application. Note that these sequences contrast with those generated via random mutations by Nikitin [18] for their designs, because we constrain our substitutions to enable an additive model. While Nikitin observed that NUPACK predictions were not accurate enough to avoid significant experimental debugging, we posit that the highly constrained nature of our substitutions may improve the reliability of the non-orthogonal designs. This claim is supported by our experimental results, although further validation is warranted.

2.2 Design process with isometric graph embeddings

This model establishes a relationship between the binding affinities between D and D^* and the Hamming distances between binary strings. However, it does not solve the problem of designing the sequences themselves given desired binding behavior. To accomplish this, we proposed designing the binary strings for a system directly, and afterwards transforming these into the sequences themselves. This latter step is trivial given binary strings, an initial sequence, and an associated set of substitutions.

A specification of the binding affinities between pairs of sequences can be represented in a matrix $A = (a_{ij})$, where a_{ij} represents the binding affinity between sequences x_i and x_j^* in some units. In analogy to Equation (1), our goal will be to design the differential binding affinities $a_{ij} - a_{i'j'}$ such that

$$a_{ij} - a_{i'j'} \propto \Delta G(x_i, x_j^*) - \Delta G(x_{i'}, x_{j'}^*). \tag{2}$$

To ease design, we chose to consider those design matrices that may be represented in an undirected graph G such that there is some set of vertices $u_i \in V(G)$ where $V(G)$ denotes the vertex set of G. Specifically, we require the existence of a graph such that

$$a_{ij} - a_{i'j'} \propto d_G(u_i, u_j) - d_G(u_{i'}, u_{j'}) \tag{3}$$

where d_G is the shortest path metric on pairs of vertices of G. From this equation, we may immediately conclude the following:

$$a_{ij} - a_{ji} \propto d_G(u_i, u_j) - d_G(u_j, u_i) = 0 \implies a_{ij} = a_{ji} \tag{4}$$

$$a_{ii} - a_{jj} \propto d_G(u_i, u_i) - d_G(u_j, u_j) = 0 \implies a_{ii} = a_{jj} \tag{5}$$

Thus, representation in an undirected graph implies that the design matrix must be symmetric with all diagonal entries identical. In addition, because the shortest path metric is a metric space, the design matrix must also represent a metric space (i.e., satisfy the triangle inequality).

Next, we proposed using a mapping between graphs called an isometric graph embedding, which is a mapping $\phi : V(G) \to V(G')$ between the vertex sets of two graphs that preserves the distances between vertices. That is, for all $u, v \in V(G)$,

$$d_G(u, v) = d_{G'}(\phi(u), \phi(v)). \tag{6}$$

We are particularly interested in isometric graph embeddings into hypercube graphs, or hypercube embeddings. A hypercube graph of dimension m is a graph of 2^m vertices, with each vertex associated with a binary string and two vertices adjacent if and only if their binary strings differ at a single position. These graphs are of interest to us because they naturally represent the Hamming distance between two binary strings in the graph distance between the corresponding hypercube vertices.

For any graph G, finding a hypercube embedding $h : V(G) \to V(H)$ into m-dimensional hypercube graph H immediately implies an assignment of binary strings to each vertex of the graph such that the distance between vertices equals the Hamming distance between binary strings. Given the results of Section 2.1, this completes our stated design goal, assuming we are able to find an initial sequence and a set of substitutions of size at least m:

$$a_{ij} - a_{i'j'} \propto d_G(u_i, u_j) - d_G(u_{i'}, u_{j'}) \tag{7}$$

$$= d_H(h(u_i), h(u_j)) - d_H(h(u_{i'}), h(u_{j'})) \tag{8}$$

$$= \delta_{\text{mut}} \times [\Delta G(x_i, x_j^*) - \Delta G(x_{i'}, x_{j'}^*)] \tag{9}$$

where in the final equality, x_i is the sequence associated with binary string $h(u_i)$, so $f(x_i) = h(u_i)$.

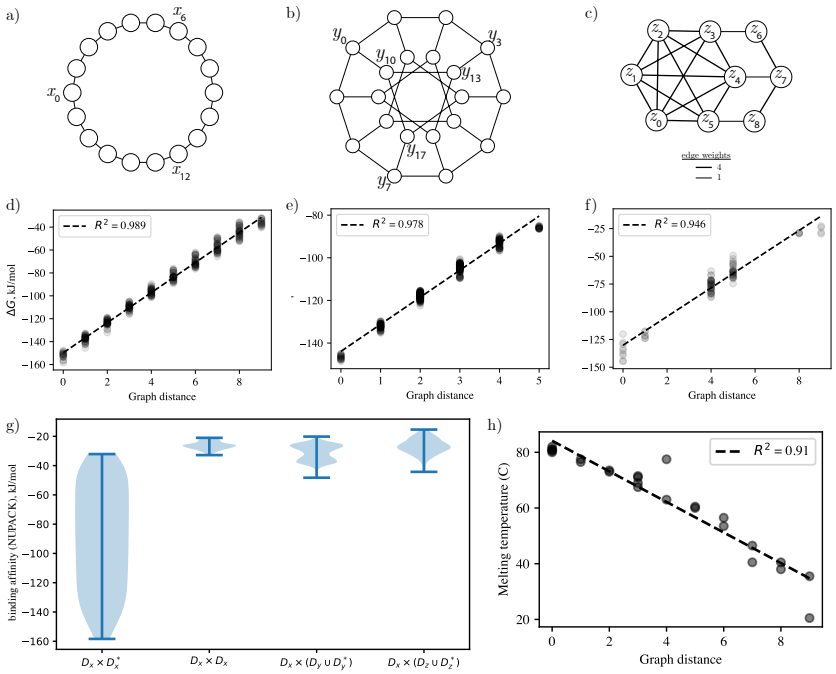

Figure 2 a) A cyclic graph on 18 nodes. b) The Desargues graph. c) A hypercube embeddable weighted graph. d-f) NUPACK simulations of designed sequences show a strong linear relationship between the graph distance and binding affinities of every pair of sequences from D and D^*. g) Distributions of binding affinities between D_x and other sets of sequences. Binding affinites for sequences that should not interact are unfavorable, comparable to the weakest binding affinites observed among the correct pairs of sequences. h) Experimental testing of a subset of the sequences in D_x and D_x^* (Technical Appendix) shows a linear relationship between melting temperature and graph distance. Melting temperature is used here as a proxy for binding affinity.

Given a graph G, the difficulty of finding a hypercube embedding varies depending on the properties of G. For unweighted, undirected graphs, hypercube embeddings were first characterized by Djoković [8], with several important results later established by Winkler and Graham [11, 28]. If an unweighted, undirected graph G permits a hypercube embedding it is called a partial cube, and determining whether a particular graph is a partial cube may be performed in $O(v^2)$ time [9], for a graph with v nodes. In addition, all hypercube embeddings of G are equivalent up to symmetries of the hypercube graph [28]. While not all graphs are partial cubes, many important classes of graphs do permit hypercube embeddings. Two examples are a cyclic graph of $2m$, which is embeddable into an m-dimensional hypercube, and the Desargues graph on 20 nodes, which is embeddable into a 5-dimensional hypercube (Figure 2ab).

To give an intuition for how such embeddings are constructed, we briefly describe the algorithm proposed by Graham and Winkler [11], noting that more efficient algorithms exist. First, a relation θ is defined on the edge set $E(G)$ of the graph G, where $uv\theta u'v'$ if and only if the quantity $[d(u,u') - d(u,v')] - [d(v,u') - d(v,v')]$ is nonzero. The transitive closure $\hat{\theta}$ of θ is an equivalence relation, and the equivalence classes of $\hat{\theta}$ partition $E(G)$ into sets $E_1, \ldots, E_m$. For set E_i, let G_i be the graph with $V(G_i) = V(G)$ and $E(G_i) = E(G) \setminus E_i$ (i.e., formed by removing the edges in E_i from G). If every G_i has exactly two connected components, then G is hypercube embeddable. In this case, a hypercube embedding may be constructed by assigning binary strings to each vertex of G such that the i-th bit of a vertex is 0 if it is one connected component of G_i and 1 if it is in the other.

When a graph G is weighted, the task of finding a hypercube embedding is NP-hard [7], so an efficient and generally-applicable solution to the problem is unlikely to exist. However, methods for efficiently embedding some classes of weighted G exist (Figure 2c) [5, 24, 25]. Characterizing additional classes of weighted graphs for which hypercube embeddings can be efficiently found remains an open area of research in graph theory, and advances in this area would improve the utility of this design process.

2.3 Validation via simulation and experiment

We consider three graphs of varying complexity (Figure 2a-c), and construct hypercube embeddings and sequence designs for each. For the three designs, we denote the designed sequences as D_x, D_y, and D_z, and the final binary strings and sequences are given in the Technical Appendix. To show the robustness of our design, each design used a distinct initial sequence taken from a previously designed set of 240,000 25-nt PCR primers known to be orthogonal to one another [29]. In each case, the pairwise binding affinities between the corresponding set of sequences D_i and the set of complementary sequences D_i^*, $i \in \{x, y, z\}$ were simulated with NUPACK at 25 °C in an aqueous solution of 1M NaCl. The simulations show a strong linear correlation between the binding affinities and the original graph distances (Figure 2d-f), with correlation coefficients of $R^2 = 0.99$, 0.98, and 0.95, respectively for the three designs.

To test the orthogonality of separately designed sets of non-orthogonal sequences, we computed binding affinities between sets of sequences that should not interact. Specifically, we tested the set D_x against the sets D_x, D_y, D_y^*, D_z, and D_z^*, and found weak binding affinities in all cases (Figure 2g). The binding affinities were comparable with the weakest binding observed among any of the expected pairs of interacting sequences.

Experimental tests of the first sequence design (D_x and D_x^*) support the validity of our model. As a proxy for binding affinity, we measured melting temperatures for 25 pairs of sequences (5 sequences of D_x and 5 sequences of D_x^*), covering a range of graph distances. Melting temperature was interpreted as a proxy for binding affinity, with higher melting temperature indicating stronger binding affinity. Melting temperatures plotted against the expected distance in the cyclic graph showed a strong linear correlation with a correlation coefficient of $R^2 = 0.91$ (Figure 2h).

3 Analog computation with promiscuous ligand-receptor networks

In naturally occurring biological networks, promiscuous binding between components (non-orthogonality) is common [1, 15]. Recently, researchers studied the computational abilities of promiscuous ligand-receptor networks, such as the BMP signaling pathway [1]. In their model of this pathway, trimers are formed from the binding of a ligand to two receptor halves, with each trimer having its associated free energy and signal output production rate. The ligand concentrations, which were assumed to be in excess, affect the formation of trimers and subsequent signal output, and these concentrations were the network inputs. The receptor concentrations, binding affinities, and trimer production rates constitute network parameters that determine the particular function computed by the system. The authors identified four archetypal response types characteristic of the promiscuous ligand-receptor network architecture, which they labeled “additive”, “ratiometric”, “balance”, and “imbalance” gates [1]. In this section, we propose and simulate a DNA strand displacement analogue to the promiscuous ligand-receptor network.

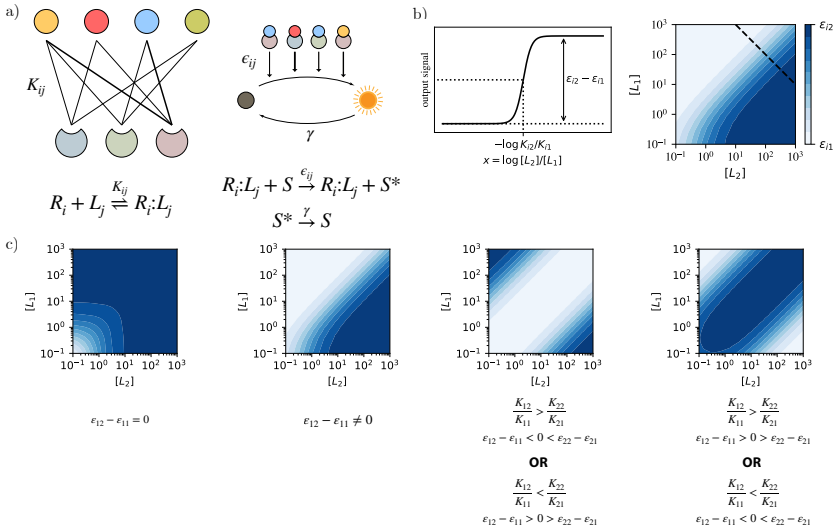

Figure 3 a) The dimer model for a promiscuous ligand-receptor network uses a set of K_{ij} binding affinities between each pair of ligands and receptors, and a set of ϵ_{ij} reaction rates to represent the signal activation rates associated with each ligand-receptor pair. b) The signal response of each receptor is independent of the other receptors, because we assume high ligand concentrations. For a two-ligand network, the response is sigmoidal in the log-ratio of the two ligand concentrations (left). This is also seen in a diagonal cross-section (dotted line) of a log-log heatmap (right). c) From left to right, an additive response, a ratiometric response, an imbalance response, and a balance response. Below each heatmap are conditions on K_{ij} and ε_{ij} to achieve each response.

3.1 Dimer model for the promiscuous ligand-receptor network

Here, we present a simplified model of a promiscuous ligand-receptor network, based only on dimer formation between ligands and receptors. While less biologically pertinent, the dimer model is capable of computing the same four archetypal response types. The dimer model considers a system of n_R receptors and n_L ligands, such that each ligand-receptor pair may bind and generate signal output with an arbitrary binding affinity and signal production rate (Figure 3a). For simplicity, we let $\varepsilon_{ij} = \frac{\epsilon_{ij}}{\gamma}$. Under the limit of high total ligand concentration, we may assume that all receptors are bound to some ligand, and the steady-state network output is given by

$$[S]_{\text{ss}} = \sum_{1 \leq i \leq n_R} [R_i]_0 \frac{\sum_{1 \leq j \leq n_L} \varepsilon_{ij} K_{ij} [L_j]_0}{\sum_{1 \leq j \leq n_L} K_{ij} [L_j]_0}, \tag{10}$$

where $[A]_0$ is the total concentration of species A, bound or unbound. A full mathematical analysis of the model is given in the Technical Appendix.

In our case, we are interested in a system of two ligands and two receptors, which is already capable of generating the four archetypal response types identified in the context of the BMP signaling pathway [1]. We consider the system response as a function of $x = \log \frac{[L_2]}{[L_1]}$, because in the limit of high ligand concentration, the signal output of a two-ligand system is a function only of the ratio of the two ligand concentrations. Consider the receptor R_1. As the x increases (L_2 dominates L_1), the receptor transitions from binding predominantly to L_1 to binding predominantly to L_2. This transition follows a sigmoidal function of x, and the signal output due to R_1 is a sigmoid whose minimum and maximum are determined by the output production rates of $R_1{:}L_1$ and $R_1{:}L_2$ (Figure 3b). Because ligand concentration is high, the binding of R_2 to L_1 and L_2 is not affected by that of R_1, so the total network output is a sum of the two sigmoidal responses generated by the two receptors.

Each of the four archetypal response types is generated from a linear combination of two sigmoids, and different responses may be chosen by transforming the sigmoidal response of each receptor. The sigmoidal response of a receptor can be shifted horizontally by adjusting the relative binding affinity of the receptor to each ligand. The response can be shifted and scaled vertically by adjusting the output production rates of its dimers. Parameter constraints and dimer model simulations for each response type are shown in Figure 3c. Note that the additive and ratiometric responses require only a single receptor.

3.2 DNA strand displacement-based implementation

We designed a set of DNA-based "ligands" and "receptors" to implement the behavior of the dimer model for a promiscuous ligand-receptor network. Each DNA-based ligand and receptor uses a combination of classical orthogonal domains along with additional domains taken from two sets of independently designed non-orthogonal sequences (Figure 4a). The signal output is generated by a fluorophore-quencher FRET pair, so that the fluorophore is quenched in certain ligand-receptor conformations. Changes in response type are possible simply by swapping out the choice of non-orthogonal domains on each DNA-based ligand and receptor.

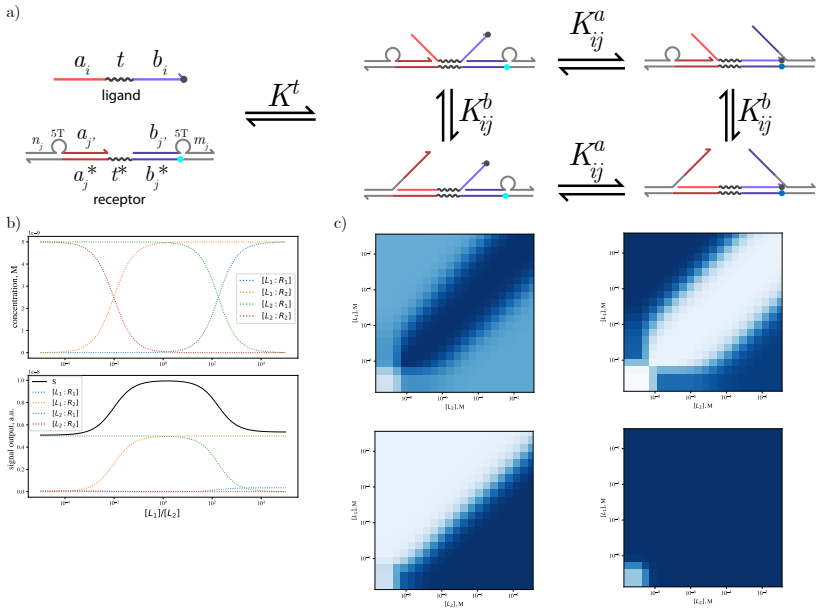

Figure 4 a) A proposed DNA strand displacement implementation of a promiscuous ligand-receptor network. b) For a balance gate, NUPACK simulations of the concentrations of the four ligand-receptor pairs as well as the signal response from each. The signal response is estimated from the probability that the 3′-most nucleotide of the ligand is bound to its corresponding nucleotide on the bottom strand of the receptor. c) Heatmaps from NUPACK simulations of our sequence designs for each of the four archetypal response types.

The steady-state output of this network is given by the following equation

$$\sum_{1\leq i\leq n_R, 1\leq j\leq n_L} [(R_i{:}L_j)^*] = \sum_{1\leq i\leq n_R} \frac{[R_i]_0 \sum_{j=1}^{n_L} K^t(1+K^a_{ij})[L_j]_0}{\sum_{j=1}^{n_L} K^t(1+K^a_{ij})(1+K^b_{ij})[L_j]_0}, \tag{11}$$

where $[(R_i{:}L_j)^*]$ is the concentration of the ligand-receptor pair in a fluorescent (i.e., not quenched) state (see the Technical Appendix for a full derivation). This equation can be made formally equivalent to Equation (10) via the following substitutions

$$K_{ij} = K^t(1 + K^a_{ij})(1 + K^b_{ij}) \tag{12}$$

$$\varepsilon_{ij} = \frac{1}{1 + K^b_{ij}} \tag{13}$$

which relate the parameters of this DNA strand displacement system (K^t, K^a_{ij}, and K^b_{ij}) to the parameters of the dimer model of the previous section (K_{ij} and ε_{ij}). In this way, it is possible in principle to implement a wide variety of dimer model parameters, limited mainly by a discretization of the energy landscape due to δ_{mut} (the change in binding affinity due to a single substitution mutation).

We describe our implementation of the balance gate in detail. Implementation of the other three gates follows similarly. For our design, we used the binary strings and corresponding sequences from the first two designs in Section 2.3, D_x and D_y. To achieve the network parameters for the balance gate (Figure 3c), we chose 2 binary strings from the D_x design and 4 binary strings from the D_y set of sequences. Sequences corresponding to each binary string were designed, and NUPACK simulation was used to estimate K_{ij} and ε_{ij} using these particular sequences and Equations (12) and (13), with K^t set to 1 as it did not affect the relative values of the K_{ij}) (Table 1). Complete lists of sequences are given in the Technical Appendix.

Table 1 Table of estimated K_{ij} and ε_{ij} values for the balance gate.

K_{ij}	L_1	L_2
R_1	4.0	1.5×10^5
R_2	1.8×10^4	4.0

ε_{ij}	L_1	L_2
R_1	0.5	1.0
R_2	1.0	0.5

NUPACK does not allow direct simulation of the fluorophore or quencher modifications, so the concentration of colocalized fluorophore-quencher pairs was estimated from the probability that the rightmost nucleotide of the y domain on a ligand would be bound to its corresponding nucleotide on a receptor. Note that hydrophobic interactions in an actual experiment could stabilize this conformation further. Using NUPACK, we simulated the concentration of each ligand-receptor pair for a fixed total ligand concentration $[L_1]_0 + [L_2]_0 = 2$ μM and varying ligand concentration ratio (Figure 4b, top). This data shows the two transitions corresponding to replacement of L_1 by L_2 on each receptor. The computed network output was also computed, as well as the output from each ligand-receptor complex, showing the expected balance gate behavior (Figure 4b, bottom). Finally, a heatmap was generated showing the network output with initial receptor concentrations $[R_1]_0 = [R_2]_0 = 5$ nM and ligand concentrations ranging from 10^{-9} to 10^{-1} M. This heatmap shows that at high ligand concentration the signal output is high when the ligand concentrations are similar and low otherwise. Note that at low ligand concentrations, receptors remain unbound and the signal output will be low regardless of the ligand concentration ratio.

For the other three archetypal response types, the binary strings were selected based on the constraints for each response type in Figure 3c, and DNA sequences and response heatmaps were computed using NUPACK simulation (Figure 4c). Results show successful implementation of each response type. Note that the additive and ratiometric gates required only a single receptor for their implementations because their response profile consisted of a single sigmoid.

4 Digital logic computation with non-orthogonal DNA hairpins

Next, we wished to demonstrate the utility of our design process for digital logic computation. We were motivated by recent work due to Nikitin illustrating digital logic gates based on the binding of short ssDNA strands optimized for particular binding affinities [18]. Notably, Nikitin used sequences that were freely mutated and subsequently tested with NUPACK to achieve particular binding affinities, and additional optimization of the affinities and strand concentrations was needed for successful circuit function. The author also found that NUPACK predictions for sequences mutated in this manner were not accurate enough to avoid significant experimental trial and error [18]. This contrasts with our approach of carefully selecting the locations of each mutation and the nucleotide identities in order to achieve consistent and predictable effects on binding affinity within the population of sequences of our design. In addition, strand concentrations were largely determined from the circuit connectivity, with only a single global working concentration that required optimization.

In Section 4.1, we begin by describing the basic structure of these logic gates, which is based off of a fan-in design for a NOR gate, and our hairpin-based implementation, which we call a "hairpin logic circuit". Section 4.2 describes the application of our design process to logic circuits of varying complexity. This process is based on the construction of an interaction graph for which a hypercube embedding may be found as in Section 2.2.

4.1 Structure of the hairpin logic circuit

The gates proposed by Nikitin [18] use sequences of NOR gates implemented by controlled binding between single strands of DNA. For each NOR gate, any one of a set of input strands may bind to the output strand, preventing it from binding to any downstream strands. In contrast to Nikitin's proposed design, our sequence designs are intended for control of the differential binding affinities of distinct double-stranded DNA complexes, rather than the change in free energy associated with two single strands forming a duplex. Thus, we proposed a hairpin-based circuit design in which each hairpin stem consists of two non-orthogonal domains that may either bind to each other (stem closed) or to another hairpin (stem open) (Figure 5a). For hairpin signal X, we denote the non-orthogonal stem domains with lowercase x and $\bar{x}^*$. The structure of our circuits is identical to those of Nikitin, except for the substitution of each single-stranded signal molecule with one of our hairpin molecules. For example, the NOT gate is simply a single-input NOR gate (Figure 5b), whose output signal is high when the input is low using a FRET pair to modulate fluorescence. Other examples of simple gates constructed from the NOR-gate primitive are shown in Figure 5c.

The favorability of two hairpins A and B opening and binding to each other is dependent on several factors. These include the degree of sequence complementarity in their stems, the favorability of opening up the hairpin loop, and the entropic penalty of replacing two freely moving molecules with a single molecule. Our sequence design process controls the first factor (i.e., the affinity of a to $\bar{b}^*$ and b to $\bar{a}^*$ relative to the affinity of a to $\bar{a}^*$ and b to $\bar{b}^*$). The latter two factors are dependent on system parameters, such as the strand concentrations and the design of the hairpin loop, and are expected to be approximately equal for all hairpin signal binding reactions.

For each circuit, we assume a working concentration c, which can be increased or decreased to tune the global favorability of a free (closed) hairpin state against the sequestered (open) hairpin state in which it is bound to another (open) hairpin signal. Because some hairpin signals must bind to multiple downstream signals as part of their function, the concentration of each signal was set to the total concentration of the downstream signals to which it must

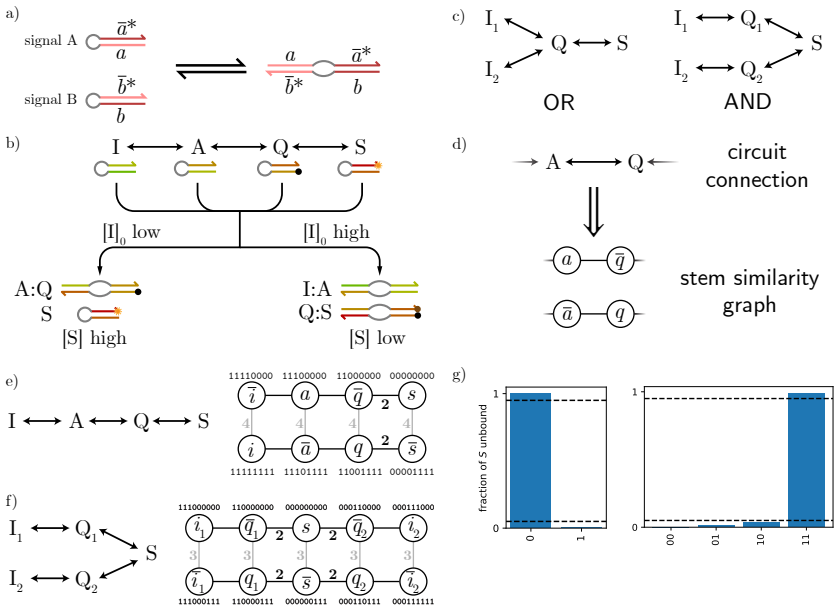

Figure 5 a) The hairpin logic circuit uses signal strands that form either a closed hairpin when isolated or an open conformation when bound to another signal. b) Example operation of a NOT gate. Each Q hairpin has a quencher label that reduces fluorescence from S when they are bound. c) Each layer of the circuit is a NOR gate, allowing the implementation of any other logic gate. Circuits for an OR and an AND gate are shown. d) To design the domains for each hairpin stem, a stem similarity graph is constructed that represents the desired binding relationships between each pair of stem domains. Each circuit connection corresponds to two edges in the stem similarity graph, to ensure that the domains on the two hairpins can bind to each other. e-f) Stem similarity graphs and hypercube embeddings for the NOT and AND gates. g) NUPACK simulations of circuit output for each gate. Performance is within 5% of the expected output (dotted lines) for all inputs.

bind. In principle, this means some hairpins will be at higher concentration and thus more likely to bind to other hairpins; however, in practice, this did not materially affect the circuit behavior (free energy of binding is affected by the logarithm of concentration, so orders of magnitude change would likely be required to affect the circuit).

4.2 Sequence design for the hairpin logic circuit

From the connectivity of a circuit diagram, we can graphically represent the desired similarity between hairpin stem domains. For a circuit C, the hairpin stem similarity graph $\mathcal{H}(C)$ represents each hairpin stem domain with a vertex (i.e., two vertices per hairpin signal). If two hairpin signals A and B are connected in the circuit diagram, then $a\bar{b}$ and $\bar{a}b$ are (unweighted, undirected) edges in $\mathcal{H}(C)$ (Figure 5d). This implies that a and $\bar{b}$ should differ by only a single mutation, and similarly for $\bar{a}$ and a. The stem similarity graph for a NOT gate is shown in Figure 5e. In this simple case, the stem similarity graph has two connected components, which means that the graph distance for the domains in each hairpin stem can be freely chosen (i.e., a to $\bar{a}$ for hairpin signal A). For the NOT gate, we assign a graph distance of 4 between every pair of domains on a single hairpin, which corresponds to 4 mutations within each hairpin stem. The implications of this choice are discussed further later.

When a circuit has at least 3 layers, it becomes necessary to adjust the stem similarity graph so that the output signals bind less strongly to their connected signals. This ensures that the output signal will only bind to any preceding signals if those signals have no other binding partners. To accomplish this with the AND gate, we apply a weight of 2 to each edge incident to a non-orthogonal domain of the output hairpin signal (Figure 5f).

Given a stem similarity graph, a hypercube embedding can be found in $O(|V|^2)$ time if one exists, for a graph with $|V|$ vertices. Hypercube embeddings for the NOT and AND gates are shown in Figure 5ef. A unitless binding affinity of two hairpin signals, neglecting factors other than stem loop complementarity, can be estimated with the equation

$$\delta(A, B) = d(a, \bar{b}) + d(b, \bar{a}) - d(a, \bar{a}) - d(b, \bar{b}) \tag{14}$$

Note that two hairpins that are intended to bind are guaranteed will always have the lowest binding affinities because each stem domain was separated by only a single edge in the stem similarity graph. This is true even for the output signal domains when they have incident edge weights of 2.

Corresponding sequences for the NOT and AND gate designs are taken from the sequence set D_y (Section 2.3; see the Technical Appendix for full set of sequences). When fewer than 9 substitutions were needed, such as with the NOT gate, a subset of the 9 substitutions available with this design was selected. In addition, the order of the substitutions was randomized, in order to distribute the mutations more evenly across the sequence. This randomization was performed for the other gates as well. Using NUPACK, the circuit output for high and low initial input concentration was computed as the concentration of the closed hairpin S (Figure 5g), showing performance within 5% of the optimal values.

To ensure that only desired hairpin pairs would bind, we modulated the working concentration for each circuit so that binding between undesired hairpin pairs was less favorable than the hairpins remaining free (closed). Previously, we chose a graph distances of 4 (NOT gate) or 3 (AND gate) between pairs of domains corresponding to the same hairpin signal, to demonstrate the effect of this design choice. With the NOT gate, a working concentration of 10^{-6} M was most effective, while with the AND gate, a significantly higher working concentration of 10^{-4} M was used. In general, this graph distance can be increased at the cost of additional bits in the hypercube embedding, in order to allow a lower working concentration to correctly balance the open and closed hairpin states. This flexibility can be useful, for instance, if the working concentration is fixed by other experimental factors and cannot be independently modulated.

For some circuits, the stem similarity graph constrains the graph distance between the two domains on each hairpin signal. The simplest feedforward circuit for which this occurs is the logically false circuit for $\mathrm{NOR}(A, \neg A)$, which can be implemented with 3 signals A, $B = \neg A$, and $C = \mathrm{NOR}(A, \neg A)$ (Figure 6a). In this case, the corresponding stem similarity graph constrains the distance between a and $\bar{a}$ to be 3 (similarly for B and C). Situations such as this occur whenever the circuit is not bipartite, when viewed as a graph on the hairpin signals themselves. This becomes increasingly common with larger circuits, and in such cases, the working concentration must be chosen based on the constraints implied by the stem similarity graph.

As a final example, we consider the square root circuit first considered by Qian and Winfree [20], which was also implemented by Nikitin [18] (Figure 6b). The stem similarity graph for this circuit is significantly more complex than the previous examples (Figure 6c), consisting of 30 vertices and 34 edges. As with the AND gate, a weight of 2 was assigned to the edges incident to the domains of the output signals S_1 and S_2. This stem similarity graph

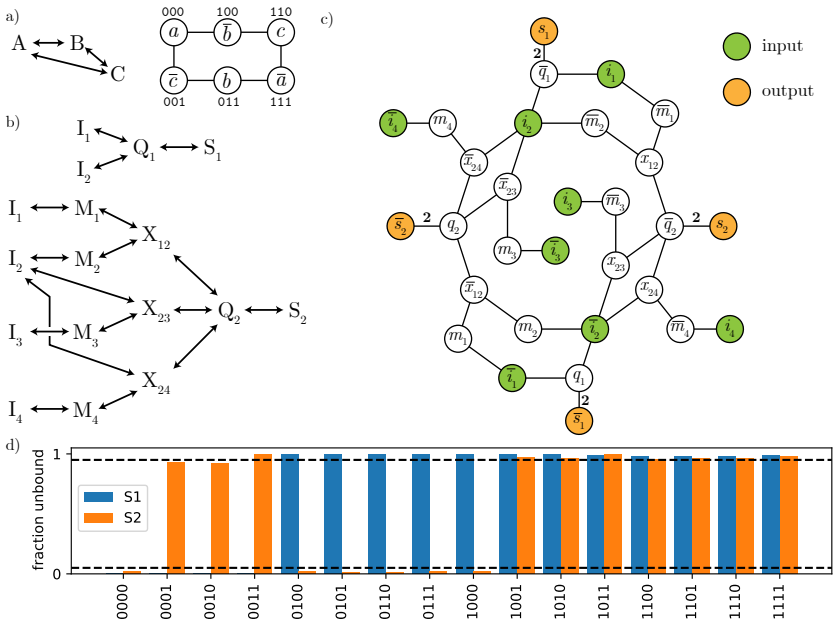

Figure 6 a) When a stem similarity graph is connected, the graph distance between domains on the same hairpin is constrained. This is a circuit for A NOR NOT A, which is logically false. b) Circuit connections for the square-root circuit. This is the same circuit that was constructed by Nikitin [18]. c) Stem similarity graph for the square root circuit. Domains on the input and output hairpins are colored for clarity. d) NUPACK simulations of circuit output for all 16 input combinations (inputs are listed in the order $I_1I_2I_3I_4$). Outputs remain within 10% of the correct values for all inputs, and within 5% (dotted lines) for all but 2 of the inputs (0001 and 0011).

is not exactly hypercube embeddable. For this case, we introduce the notion of a *k-hypercube embedding*, or an isometric embedding into a hypercube that preserves all distances up to a distance of k. Precisely, $\phi : V(G) \to V(H)$ is a k-hypercube embedding if for all $u, v \in V(G)$:

1. $d_G(u, v) = d_H(\phi(u), \phi(v))$ if $d_G(u, v) \leq k$
2. $d_G(u, v) > k$ if $d_G(u, v) > k$

This notion is a generalization of the k-snake, which is a k-hypercube embedding of a linear graph [13, 26].

The use of a k-hypercube embedding has an additional advantage, in that it may allow the researcher to embed into a hypercube of smaller dimension (i.e., requiring fewer bits). Unfortunately, an efficient algorithm for finding a k-hypercube embedding is not known, although for smaller circuits this is often possible to do by hand.

For the square root circuit, we constructed a 2-hypercube embedding of dimension 9, where $k = 2$ was chosen because all correct interactions were encoded in the stem similarity graph with a distance of at most 2 (see Technical Appendix). Any vertices at distance 3 or greater should not interact anyway, and the working concentration will be chosen to ensure that these interactions are not favorable. Our 2-hypercube embedding has the added advantage that for every hairpin, its two domains have the same Hamming distance from each other. This makes it easier to choose a working concentration that correctly tunes the propensity of each hairpin signal to choose an open conformation (bound to another hairpin) over a closed one.

To simulate this circuit, sequences corresponding to each binary string were used from the set D_y from Section 2.3. A working concentration of 10^{-11} M was used; all signals were initially present at this concentration except for I_1 and I_2, which used concentrations of

2×10^{-11} and 4×10^{-11}, respectively. NUPACK simulations were performed over all 16 possible 4-bit inputs, which showed that, for all input combinations, the 2-bit output was within 10% of the correct value (Figure 6d). All but two inputs (0001 and 0011) achieved outputs within 5% of the correct values. It is possible that additional circuit optimization could improve this further.

We make one final comments regarding this application of our sequence design procedure to digital logic circuits. When some hairpin signals have stems that bind more strongly than others, choosing a global working concentration can be difficult. In the square root circuit, we circumvented this problem in the way we compressed our hypercube embedding; however, this may not always be possible. Handling this scenario remains another open question that could make it possible to apply our design process to additional circuits.

5 Discussion and Conclusions

In this work, we have presented an effective rational design approach for non-orthogonal DNA domains, that may be combined with orthogonal domains for complex DNA strand displacement circuitry. We applied this approach to two previously proposed molecular computational systems based on non-orthogonal interactions, through which we demonstrated *in silico* analog and digital logic computation. This design approach eases the use of non-orthogonal domains within DNA strand displacement cascades, because it allows researchers to perform initial designs of non-orthogonal domains using only binary strings. Experimental implementation requires finding an appropriate starting DNA sequence with enough substitutions; heuristically, a starting sequence with little secondary structure and that binds to its perfect complement primarily in a single conformation works well. Once found, generating the corresponding DNA sequence variants is straightforward, allowing researchers to quickly design candidate non-orthogonal domains. Thus, a larger portion of the design process can be done prior to ordering materials. Further experimental testing is warranted to quantify the accuracy of our design method.

Several areas for further improvements to our method exist. For example, the use of k-hypercube embeddings was necessary for the more complex square root circuit. Additional work could address algorithms for efficiently constructing k-hypercube embeddings, which would also help with compressing the embeddings so they can be implemented on shorter DNA strands.

The sequence design process we describe here has the potential to be applied in other contexts. For example, previous DNA strand displacement-based implementations of neural networks have used "weight complexes," dedicated auxiliary DNA molecules whose concentrations encode the weights between various nodes [6, 21]. However, as the size of a neural network grows, the number of weights that must be encoded grows quadratically in the number of nodes, so that the number of distinct DNA molecules that must be designed and synthesized grows quickly. Using non-orthogonal sequence design, it may be possible to encode weights in the binding affinities between molecules, which would significantly reduce the number of system components required, increasing the size of the neural networks that can feasibly be implemented experimentally.

Recently, researchers have shown that non-orthogonal interactions can also be applied to nanostructure fabrication, allowing the design of a pool of structural subunits capable of creating any one of several possible multi-subunit assemblies. The creation of a particular assembly can be triggered by the presence of a nucleation seed [16] or by the concentrations of the various subunits [30]. An experimental demonstration of this used a set of 917 unique

DNA tiles to construct one or more of three target assemblies [10]. This demonstration used orthogonal interactions between tiles; however, our sequence design method could in principle be used to allow the construction of a larger number of target assemblies closer to the theoretical limit [16].

The problem of non-orthogonal sequence design is a complex task, with different approaches likely to be best suited to different applications. However, there are many potential use cases for a non-orthogonal sequence design approach that can be applied to a variety of DNA strand displacement systems. We hope that our non-orthogonal sequence design method spurs new innovation in both DNA sequence design and the computational uses of non-orthogonality, and that future improvements to non-orthogonal design techniques will open the doors to more complex DNA-based computers and new advancements in molecular computing.

References

1 Yaron E Antebi, James M Linton, Heidi Klumpe, Bogdan Bintu, Mengsha Gong, Christina Su, Reed McCardell, and Michael B Elowitz. Combinatorial signal perception in the BMP pathway. *Cell*, 170(6):1184–1196, 2017.

2 Eric B Baum. Building an associative memory vastly larger than the brain. *Science*, 268(5210):583–585, 1995.

3 Callista Bee, Yuan-Jyue Chen, Melissa Queen, David Ward, Xiaomeng Liu, Lee Organick, Georg Seelig, Karin Strauss, and Luis Ceze. Molecular-level similarity search brings computing to DNA data storage. *Nature communications*, 12(1):1–9, 2021.

4 Joseph Berleant. *DNA sequence design of non-orthogonal binding networks, and application to DNA data storage.* PhD thesis, Massachusetts Institute of Technology, 2023.

5 Joseph Berleant, Kristin Sheridan, Anne Condon, Virginia Vassilevska Williams, and Mark Bathe. Isometric Hamming embeddings of weighted graphs. *Discrete Applied Mathematics*, 332:119–128, 2023.

6 Kevin M Cherry and Lulu Qian. Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks. *Nature*, 559(7714):370–376, 2018.

7 Vašek Chvátal. Recognizing intersection patterns. *Annals of Discrete Mathematics*, 8:249–252, 1980. doi:10.1002/net.3230210602.

8 Dragomir Ž Djoković. Distance-preserving subgraphs of hypercubes. *Journal of Combinatorial Theory, Series B*, 14(3):263–267, 1973.

9 David Eppstein. Recognizing partial cubes in quadratic time. *Journal of Graph Algorithms and Applications*, 15(2):269–293, 2011.

10 Constantine Glen Evans, Jackson O'Brien, Erik Winfree, and Arvind Murugan. Pattern recognition in the nucleation kinetics of non-equilibrium self-assembly. *arXiv preprint arXiv:2207.06399*, 2022.

11 Ronald L Graham and Peter M Winkler. On isometric embeddings of graphs. *Transactions of the American Mathematical Society*, 288(2):527–536, 1985.

12 Haukur Gudnason, Martin Dufva, Dang Duong Bang, and Anders Wolff. Comparison of multiple DNA dyes for real-time pcr: effects of dye concentration and sequence composition on DNA amplification and melting temperature. *Nucleic acids research*, 35(19):e127, 2007.

13 Simon Hood, Daniel Recoskie, Joe Sawada, and Dennis Wong. Snakes, coils, and single-track circuit codes with spread k. *Journal of Combinatorial Optimization*, 30:42–62, 2015.

14 Matthew R Lakin, Simon Youssef, Filippo Polo, Stephen Emmott, and Andrew Phillips. Visual dsd: a design and analysis tool for dna strand displacement systems. *Bioinformatics*, 27(22):3211–3213, 2011.

15 Tomas Malinauskas and E Yvonne Jones. Extracellular modulators of Wnt signalling. *Current opinion in structural biology*, 29:77–84, 2014.

16 Arvind Murugan, Zorana Zeravcic, Michael P Brenner, and Stanislas Leibler. Multifarious assembly mixtures: Systems allowing retrieval of diverse stored structures. *Proceedings of the National Academy of Sciences*, 112(1):54–59, 2015.

17 Andrew Neel and Max Garzon. Semantic retrieval in DNA-based memories with Gibbs energy models. *Biotechnology progress*, 22(1):86–90, 2006.

18 Maxim P Nikitin. Non-complementary strand commutation as a fundamental alternative for information processing by DNA and gene regulation. *Nature Chemistry*, pages 1–13, 2023.

19 Nicolas Peyret, P Ananda Seneviratne, Hatim T Allawi, and John SantaLucia. Nearest-neighbor thermodynamics and NMR of DNA sequences with internal A⊙A, C⊙C, G⊙G, and T⊙T mismatches. *Biochemistry*, 38(12):3468–3477, 1999.

20 Lulu Qian and Erik Winfree. Scaling up digital circuit computation with DNA strand displacement cascades. *Science*, 332(6034):1196–1201, 2011.

21 Lulu Qian, Erik Winfree, and Jehoshua Bruck. Neural network computation with DNA strand displacement cascades. *Nature*, 475:368–72, July 2011. doi:10.1038/nature10262.

22 John SantaLucia Jr. and Donald Hicks. The thermodynamics of DNA structural motifs. *Annual Review of Biophysics and Biomolecular Structure*, 33(1):415–440, 2004. doi:10.1146/annurev.biophys.32.110601.141800.

23 Georg Seelig, David Soloveichik, David Yu Zhang, and Erik Winfree. Enzyme-free nucleic acid logic circuits. *science*, 314(5805):1585–1588, 2006.

24 Kristin Sheridan, Joseph Berleant, Mark Bathe, Anne Condon, and Virginia Vassilevska Williams. Factorization and pseudofactorization of weighted graphs. *Discrete Applied Mathematics*, 337:81–105, 2023. doi:10.1016/j.dam.2023.04.019.

25 Sergey V. Shpectorov. On scale embeddings of graphs into hypercubes. *Eur. J. Comb.*, 14(2):117–130, March 1993. doi:10.1006/eujc.1993.1016.

26 Richard C Singleton. Generalized snake-in-the-box codes. *IEEE Transactions on Electronic Computers*, pages 596–602, 1966.

27 Kyle J Tomek, Kevin Volkel, Elaine W Indermaur, James M Tuck, and Albert J Keung. Promiscuous molecules for smarter file operations in DNA-based data storage. *Nature Communications*, 12(1):3518, 2021.

28 Peter M Winkler. Isometric embedding in products of complete graphs. *Discrete Applied Mathematics*, 7(2):221–225, 1984.

29 Qikai Xu, Michael R Schlabach, Gregory J Hannon, and Stephen J Elledge. Design of 240,000 orthogonal 25mer DNA barcode probes. *Proceedings of the National Academy of Sciences*, 106(7):2289–2294, 2009.

30 Weishun Zhong, David J Schwab, and Arvind Murugan. Associative pattern recognition through macro-molecular self-assembly. *Journal of Statistical Physics*, 167:806–826, 2017.

A Methods

A.1 NUPACK simulation parameters

All NUPACK simulations were performed at 25 °C, with 1 M NaCl. NUPACK simulations for Sections 2 and 3 used material "dna04-nupack3" and ensemble "some-nupack3". NUPACK simulations for Section 4 used material "dna" and ensemble "stacking".

A.2 DNA melting assay

The sequences x_0, x_1, x_3, x_6, and x_{10} and their complements were tested. Because intercalating dyes such as SYBR Green often affect the binding affinity of DNA strands or have strong variations in fluorescence with temperature [12], we used a FRET assay with strands labeled either with cyanine 3 (Cy3) or cyanine 5 (Cy5) dyes. Cy3- or Cy5-labeled strands were ordered from Integrated DNA Technologies (IDT). Strands were mixed to a concentration

of 0.25 μM per strand in a solution of 1× TAE and 1 M NaCl. Fluorescence of Cy3 was measured on a qPCR machine (QuantStudio Flex 6, Applied Biosystems) over a temperature ramp from 95 °C down to 10 °C and then back to 95 °C, at a rate of 0.5 °C/min. This was repeated three times. Each curve was corrected with the Cy3 fluorescence measured from a sample of the single strand without any binding partner. Data from the upward ramp was averaged, and the average rate of change in fluorescence with temperature was computed. The temperature with highest rate of change was taken as the melting temperature.

B Thermodynamic analysis of the promiscuous ligand-receptor system

B.1 Analysis of the dimer model

The dimer model uses a set of n_R receptors R_i and n_L ligands L_j and is capable of generating the four archetypal response types. This requires fewer components than the trimer model used in by Antebi et al. in their analysis of the BMP signaling pathway [1].

$$R_i + L_j \underset{}{\overset{K_{ij}}{\rightleftharpoons}} D_{ij} \tag{15}$$

$$D_{ij} \xrightarrow{\epsilon_{ijk}} D_{ij} + S \tag{16}$$

$$S \xrightarrow{\gamma} \emptyset \tag{17}$$

Let $\varepsilon_{ij} = \frac{\epsilon_{ij}}{\gamma}$. Given these reactions, the steady-state level of S is

$$[S]_{\text{ss}} = \sum_{1\le i\le n_R, 1\le j\le n_L} \varepsilon_{ij}[D_{ij}]_{\text{ss}} \tag{18}$$

$$= \sum_{1\le i\le n_R} [R_i]_0 \frac{\sum_{1\le j\le n_L} \varepsilon_{ij} K_{ij}[L_j]_{\text{ss}}}{1 + \sum_{1\le j\le n_L} K_{ij}[L_j]_{\text{ss}}} \tag{19}$$

$$\approx \sum_{1\le i\le n_R} [R_i]_0 \frac{\sum_{1\le j\le n_L} \varepsilon_{ij} K_{ij}[L_j]_0}{\sum_{1\le j\le n_L} K_{ij}[L_j]_0} \tag{20}$$

where $[A]_0$ is the total concentration of species A bound or unbound and the final approximation is achieved by assuming the total ligand concentration is high.

When there are exactly two ligands, each term of the summation can be individually expressed as a sigmoid curve:

$$[S]_{\text{ss}} \approx \sum_{1\le i\le n_R} [R_i]_0 \frac{\varepsilon_{i1} K_{i1}[L_1]_0 + \varepsilon_{i2} K_{i2}[L_2]_0}{K_{i1}[L_1]_0 + K_{i2}[L_2]_0} \tag{21}$$

$$= [R_1]_0 \left[\varepsilon_{11} + \frac{\varepsilon_{12} - \varepsilon_{11}}{\frac{K_{11}}{K_{12}} e^x + 1}\right] \tag{22}$$

where $x = \log\left(\frac{[L_1]_0}{[L_2]_0}\right)$. This describes a sigmoid curve that transitions from $[R_i]_0\varepsilon_{i2}$ to $[R_i]_0\varepsilon_{i1}$ with a midpoint at $-\log\frac{K_{i1}}{K_{i2}}$. The "height" of the sigmoid is given by $[R_i]_0(\varepsilon_{i2} - \varepsilon_{i1})$.

Thus, in a two ligand, two receptor system, the signal response is the sum of two sigmoids, each of which may be independently shifted along the x-axis by changing $\frac{K_{i1}}{K_{i2}}$, and shifted and/or stretched along the vertical axis by modifying ε_{i1} and ε_{i2}. Note that the steepness of the sigmoid (i.e. horizontal stretching) is not adjustable under this model; the same is true of the trimer model presented by Antebi et al. [1].

The additive and ratiometric responses may be generated by a single receptor, while the balance and imbalance responses require two receptors.

B.2 DNA strand displacement implementation

Consider the ligands and receptors shown in Figure 4a. Each ligand has two non-orthogonal domains (a_i and b_i) as well as a toehold that allows it to bind to each receptor. Once bound, the two non-orthogonal domains on the receptor that flank its toehold binding site may be displaced with some free energy change based on the relative affinities of the incumbent domains (a_j and b_j) and intruder domains to the bottom domains (a_k and b_k). A FRET-based readout is implemented by the addition of a fluorophore-quencher pair, which is activated only when the right-hand non-orthogonal domain is displaced. The full reaction diagram is enumerated below:

$$R_i + L_j \overset{K^t}{\rightleftharpoons} R_i{:}L_j \tag{23}$$

$$R_i{:}L_j \overset{K^a_{ij}}{\rightleftharpoons} (R_i{:}L_j)_a \tag{24}$$

$$R_i{:}L_j \overset{K^b_{ij}}{\rightleftharpoons} (R_i{:}L_j)_b \tag{25}$$

$$(R_i{:}L_j)_a \overset{K^b_{ij}}{\rightleftharpoons} (R_i{:}L_j)_{ab} \tag{26}$$

$$(R_i{:}L_j)_b \overset{K^a_{ij}}{\rightleftharpoons} (R_i{:}L_j)_{ab} \tag{27}$$

Thermodynamic analysis of this system is straightforward, as long as we neglect any second-order effects of adjacent domains (note that these effects can in principle be significant). The presence of the 5T loop on either side of the non-orthogonal domains on the receptor is intended to reduce the significance of this. The following equations hold at equilibrium

$$K^t = \frac{[R_i{:}L_j]}{[R_i][L_j]} \tag{28}$$

$$K^a_{ij} = \frac{[(R_i{:}L_j)_a]}{[R_i{:}L_j]} = \frac{[(R_i{:}L_j)_{ab}]}{[(R_i{:}L_j)_b]} \tag{29}$$

$$K^b_{ij} = \frac{[(R_i{:}L_j)_b]}{(R_i{:}L_j]} = \frac{[(R_i{:}L_j)_{ab}]}{[(R_i{:}L_j)_a]} \tag{30}$$

and some algebra yields the following:

$$\sum_{1\le i\le n_R, 1\le j\le n_L} [(R_i{:}L_j)^*] = \sum_{1\le i\le n_R} \frac{[R_i]_0 \sum_{j=1}^{n_L} K^t(1+K^a_{ij})[L_j]}{1+\sum_{j=1}^{n_L} K^t(1+K^a_{ij})(1+K^b_{ij})[L_j]} \tag{31}$$

$$\approx \sum_{1\le i\le n_R} \frac{[R_i]_0 \sum_{j=1}^{n_L} K^t(1+K^a_{ij})[L_j]_0}{\sum_{j=1}^{n_L} K^t(1+K^a_{ij})(1+K^b_{ij})[L_j]_0} \tag{32}$$

where $[(R_i{:}L_j)^*] = [R_i{:}L_j] + [(R_i{:}L_j)_a]$ is the concentration of fluorescing receptor, noting that $[R_i] \approx 0$ because ligands are in excess so that essentially no receptor will be unbound.

Note the similarity to Equation (20). The two are formally equivalent if we make the following correspondences

$$K_{ij} = K^t(1+K^a_{ij})(1+K^b_{ij}) \tag{33}$$

$$\varepsilon_{ij} = \frac{1}{1+K^b_{ij}}. \tag{34}$$

Thus, this system is capable of the same signal responses as the ligand-receptor system of the previous section, and can in principle implement the four archetypal responses using two ligands and two receptors.

C Tables of sequences

Table 2 Base sequence and substitutions for the three designs D_x, D_y, and D_z.

cyclic graph	Base sequence	GCCTTGTATGTGAATATCCGTGTCA
	Fully mutated sequence	GCCATGAAAGAGTAAAACCGAGACA
Desargues graph	Base sequence	GGAGAATGATTAGCACGGAGAGTGG
	Fully mutated sequence	GGTGTAAGTTAAGCTCGGTGTGAGG
weighted graph	Base sequence	CGGTGTGCTTTTACTTAAGTAACCG
	Fully mutated sequence	CGGAGAGCATATTCATTAGAATCCG

Table 3 Sequences for the cyclic graph on 18 nodes (D_x), Desargues graph (D_y), and weighted graph (D_z).

Design	Name	Binary string	Sequence
cyclic	x_0	000000000	GCCTTGTATGTGAATATCCGTGTCA
	x_1	100000000	GCCATGTATGTGAATATCCGTGTCA
	x_2	110000000	GCCATGAATGTGAATATCCGTGTCA
	x_3	111000000	GCCATGAAAGTGAATATCCGTGTCA
	x_4	111100000	GCCATGAAAGAGAATATCCGTGTCA
	x_5	111110000	GCCATGAAAGAGTATATCCGTGTCA
	x_6	111111000	GCCATGAAAGAGTAAATCCGTGTCA
	x_7	111111100	GCCATGAAAGAGTAAAACCGTGTCA
	x_8	111111110	GCCATGAAAGAGTAAAACCGAGTCA
	x_9	111111111	GCCATGAAAGAGTAAAACCGAGACA
	x_{10}	011111111	GCCTTGAAAGAGTAAAACCGAGACA
	x_{11}	001111111	GCCTTGTAAGAGTAAAACCGAGACA
	x_{12}	000111111	GCCTTGTATGAGTAAAACCGAGACA
	x_{13}	000011111	GCCTTGTATGTGTAAAACCGAGACA
	x_{14}	000001111	GCCTTGTATGTGAAAAACCGAGACA
	x_{15}	000000111	GCCTTGTATGTGAATAACCGAGACA
	x_{16}	000000011	GCCTTGTATGTGAATATCCGAGACA
	x_{17}	000000001	GCCTTGTATGTGAATATCCGTGACA
Desargues	y_0	10100	GGTGAAAGATTAGCACGGAGAGTGG
	y_1	10110	GGTGAAAGTTTAGCACGGAGAGTGG
	y_2	10010	GGTGAATGTTTAGCACGGAGAGTGG
	y_3	11010	GGTGTATGTTTAGCACGGAGAGTGG
	y_4	01010	GGAGTATGTTTAGCACGGAGAGTGG
	y_5	01011	GGAGTATGTTAAGCACGGAGAGTGG
	y_6	01001	GGAGTATGATAAGCACGGAGAGTGG
	y_7	01101	GGAGTAAGATAAGCACGGAGAGTGG
	y_8	00101	GGAGAAAGATAAGCACGGAGAGTGG
	y_9	10101	GGTGAAAGATAAGCACGGAGAGTGG
	y_{10}	11100	GGTGTAAGATTAGCACGGAGAGTGG
	y_{11}	00110	GGAGAAAGTTTAGCACGGAGAGTGG
	y_{12}	10011	GGTGAATGTTAAGCACGGAGAGTGG
	y_{13}	11000	GGTGTATGATTAGCACGGAGAGTGG
	y_{14}	01110	GGAGTAAGTTTAGCACGGAGAGTGG
	y_{15}	00011	GGAGAATGTTAAGCACGGAGAGTGG
	y_{16}	11001	GGTGTATGATAAGCACGGAGAGTGG
	y_{17}	01100	GGAGTAAGATTAGCACGGAGAGTGG
	y_{18}	00111	GGAGAAAGTTAAGCACGGAGAGTGG
	y_{19}	10001	GGTGAATGATAAGCACGGAGAGTGG
weighted	z_0	000011110	CGGTGTGCTTTTTCATTAGAAACCG
	z_1	001100110	CGGTGTGCATATACTTTAGAAACCG
	z_2	010101010	CGGTGAGCTTATACATAAGAAACCG
	z_3	000000000	CGGTGTGCTTTTACTTAAGTAACCG
	z_4	011010010	CGGTGAGCATTTTCTTAAGAAACCG
	z_5	111111110	CGGAGAGCATATTCATTAGAAACCG
	z_6	000000001	CGGTGTGCTTTTACTTAAGTATCCG
	z_7	011010011	CGGTGAGCATTTTCTTAAGAATCCG
	z_8	111111111	CGGAGAGCATATTCATTAGAATCCG

Table 4 Shared sequences for the DNA strand displacement ligand-receptor networks.

Name	Binary string	Sequence
t		GAGAACT
n_{R_1}		GGTCTTGACAAACGTGTGCT
m_{R_1}		TATGAGGACGAATCTCCCGC
n_{R_2}		CCGATGTTGACGGACTAATC
m_{R_2}		GTTTATCGGGCGTGGTGCTC
a_{R_1}	111111111	GCCATGAAAGAGTAAAACCGAGACA
$a_{R'_1}$	111111100	GCCATGAAAGAGTAAAACCGTGTCA
a_{R_2}	111111100	GCCATGAAAGAGTAAAACCGTGTCA
$a_{R'_2}$	111111111	GCCATGAAAGAGTAAAACCGAGACA
b_{R_1}	000000000	GGAGAATGATTAGCACGGAGAGTGG
$b_{R'_1}$	100000000	GGTGAATGATTAGCACGGAGAGTGG
b_{R_2}	011111111	GGAGTAAGTTAAGCTCGGTGTGAGG
$b_{R'_2}$	111111111	GGTGTAAGTTAAGCTCGGTGTGAGG

Table 5 Gate-specific sequences for the DNA strand displacement ligand-receptor networks.

Gate	Name	Binary string	Sequence
Balance	a_{L_1}	111111100	GCCATGAAAGAGTAAAACCGTGTCA
	a_{L_2}	111111111	GCCATGAAAGAGTAAAACCGAGACA
	b_{L_1}	100000000	GGTGAATGATTAGCACGGAGAGTGG
	b_{L_2}	111111111	GGTGTAAGTTAAGCTCGGTGTGAGG
Imbalance	a_{L_1}	111111110	GCCATGAAAGAGTAAAACCGAGTCA
	a_{L_2}	111111110	GCCATGAAAGAGTAAAACCGAGTCA
	b_{L_1}	100000000	GGTGAATGATTAGCACGGAGAGTGG
	b_{L_2}	111111111	GGTGTAAGTTAAGCTCGGTGTGAGG
Ratiometric	a_{L_1}	111111110	GCCATGAAAGAGTAAAACCGAGTCA
	a_{L_2}	111111111	GCCATGAAAGAGTAAAACCGAGACA
	b_{L_1}	100000000	GGTGAATGATTAGCACGGAGAGTGG
	b_{L_2}	111111111	GGTGTAAGTTAAGCTCGGTGTGAGG
Additive	a_{L_1}	111111110	GCCATGAAAGAGTAAAACCGAGTCA
	a_{L_2}	111111110	GCCATGAAAGAGTAAAACCGAGTCA
	b_{L_1}	111111111	GGTGTAAGTTAAGCTCGGTGTGAGG
	b_{L_2}	111111111	GGTGTAAGTTAAGCTCGGTGTGAGG

Table 6 Sequences for the hairpin logic circuits. The full hairpin sequence is $al\bar{a}^*$ for signal A.

Gate	Name	Binary string	Sequence
	l		TTT
NOT	i	11111111	GGTGTAAGTTAAGCTCGGTGAGAGG
	$\bar{i}$	11110000	GGTGAAAGATTAGCACGGTGAGAGG
	a	11100000	GGTGAAAGATTAGCACGGAGAGAGG
	$\bar{a}$	11101111	GGTGTAAGTTAAGCTCGGAGAGAGG
	q	11001111	GGTGTAAGTTAAGCTCGGAGAGTGG
	$\bar{q}$	11000000	GGTGAAAGATTAGCACGGAGAGTGG
	s	00000000	GGAGAATGATTAGCACGGAGAGTGG
	$\bar{s}$	00001111	GGAGTATGTTAAGCTCGGAGAGTGG
AND	i_1	111000000	GGTGAAAGATTAGCACGGAGAGAGG
	$\bar{i}_1$	111000111	GGTGAAAGATAAGCTCGGAGTGAGG
	q_1	110000111	GGTGAAAGATAAGCTCGGAGTGTGG
	$\bar{q}_1$	110000000	GGTGAAAGATTAGCACGGAGAGTGG
	i_2	000111000	GGAGTATGTTTAGCACGGTGAGTGG
	$\bar{i}_2$	000111111	GGAGTATGTTAAGCTCGGTGTGTGG
	q_2	000110111	GGAGTATGATAAGCTCGGTGTGTGG
	$\bar{q}_2$	000110000	GGAGTATGATTAGCACGGTGAGTGG
	s	000000000	GGAGAATGATTAGCACGGAGAGTGG
	$\bar{s}$	000000111	GGAGAATGATAAGCTCGGAGTGTGG
Square Root	i_1	001010100	GGAGAAAGTTTAGCACGGAGAGAGG
	$\bar{i}_1$	110100100	GGAGTATGTTAAGCACGGAGTGTGG
	i_2	001110000	GGAGTAAGATTAGCACGGAGAGAGG
	$\bar{i}_2$	110000000	GGAGAATGATAAGCACGGAGTGTGG
	i_3	010001100	GGTGAATGTTAAGCACGGAGAGTGG
	$\bar{i}_3$	101111100	GGTGTAAGTTTAGCACGGAGTGAGG
	i_4	100001100	GGTGAATGTTTAGCACGGAGTGTGG
	$\bar{i}_4$	011111100	GGTGTAAGTTAAGCACGGAGAGAGG
	m_1	111100100	GGAGTATGTTAAGCACGGAGTGAGG
	$\bar{m}_1$	000010100	GGAGAAAGTTTAGCACGGAGAGTGG
	m_2	111000000	GGAGAATGATAAGCACGGAGTGAGG
	$\bar{m}_2$	000110000	GGAGTAAGATTAGCACGGAGAGTGG
	m_3	101111000	GGTGTAAGATTAGCACGGAGTGAGG
	$\bar{m}_3$	010001000	GGTGAATGATAAGCACGGAGAGTGG
	m_4	011111000	GGTGTAAGATAAGCACGGAGAGAGG
	$\bar{m}_4$	100001000	GGTGAATGATTAGCACGGAGTGTGG
	x_{12}	000010000	GGAGAAAGATTAGCACGGAGAGTGG
	$\bar{x}_{12}$	111100000	GGAGTATGATAAGCACGGAGTGAGG
	x_{23}	010000000	GGAGAATGATAAGCACGGAGAGTGG
	$\bar{x}_{23}$	101110000	GGAGTAAGATTAGCACGGAGTGAGG
	x_{24}	100000000	GGAGAATGATTAGCACGGAGTGTGG
	$\bar{x}_{24}$	011110000	GGAGTAAGATAAGCACGGAGAGAGG
	q_1	110000100	GGAGAATGTTAAGCACGGAGTGTGG
	$\bar{q}_1$	001110100	GGAGTAAGTTTAGCACGGAGAGAGG
	q_2	111110000	GGAGTAAGATAAGCACGGAGTGAGG
	$\bar{q}_2$	000000000	GGAGAATGATTAGCACGGAGAGTGG
	s_1	001100110	GGAGTATGTTTAGCACGGTGAGAGG
	$\bar{s}_1$	110010110	GGAGAAAGTTAAGCACGGTGTGTGG
	s_2	000000011	GGAGAATGATTAGCTCGGTGAGTGG
	$\bar{s}_2$	111110011	GGAGTAAGATAAGCTCGGTGTGAGG