

Assembling Nanoscale Circuits with Randomized Connections

Tad Hogg, Yong Chen and Philip J. Kuekes*

September 8, 2005

Abstract

Molecular electronics is difficult to fabricate with precise positioning of large numbers of devices and their connections. Self-assembly techniques can create such circuits but with some random variation in their connection locations and characteristics. Using simulations, we show how to produce reliable circuits in spite of this variation by adding enough redundant components to pass a sharp threshold in likely circuit correctness. As an example of this approach, we examine a demultiplexer circuit, useful for connecting nanoscale circuits with larger conventional circuits.

Keywords: molecular electronics, circuit reliability, nanotechnology, multiplexing, self-assembly

1 Introduction

Compared to microelectronics, nanoscale electronics is still in its infancy, as microelectronics was in the 1940's when the transistor was invented. The next major development in microelectronics was the integrated circuit, in the 1950's, which ultimately facilitated interconnecting millions of microelectronic devices in an economic, manufacturable, and scalable way for today's circuits. This interconnection technology, along with decreasing device sizes, led to ultra-large integration of microelectronics for the information revolution as we witness today. These circuits rely on the technological capability to precisely construct, place and connect the components.

Molecular electronics holds the promise for significantly denser circuits than is possible for current microelectronics. The research in nanoscale electronics has mainly focused on basic building blocks of the nanoscale circuits: nanoscale dots and wires, switches and transistors that can turn an electric current on or off as well as amplify signals. Examples include molecular switching devices [1], carbon nanotube transistors and logic [2, 3], and single electron transistors with nanoscale crystals [4]. So far the progress has been impressive. However, using these nanoscale devices for arithmetic or logic operations also requires interconnection technology, i.e. methods to connect many devices into circuits. Although we know the functions we need and the weakness of nanoscale devices that must be compensated for, we currently lack the technology to reliably fabricate and connect large numbers of nanoscale devices into precise circuits. While lithography can create microscale circuits with many components, high yield and low unit cost, this combination of capabilities is not yet possible for molecular-scale devices.

Instead, creating a large number of molecular devices must rely on appropriately designed self-assembly of the molecules. In principle, self-assembly can produce complex nanoscale structures as demonstrated by biological processes. Unfortunately, our current physical and chemical self-assembly technology at the nanoscale is limited to creating simple, regular or random structures. In either case, the assembled structures have low information content. Self-assembly [5] can create a wide range of structures at many length scales.

For nanoscale circuits, the challenge is to design components simple enough to fabricate by self-assembly that are nevertheless useful for constructing complex circuits. Arrays of parallel nanowires, microns long with nanometer pitch, are one such example [6, 7]. Two of these arrays, placed orthogonally with molecules between the crossed wires, can be configured to provide electronic memory and logic circuits [8, 9, 10, 11]. Configuring and using these circuits requires electrical connections to standard microelectronics so that the nanowires are individually addressable [12, 13]. The most direct approach spreads the nanowires far

*The authors are at HP Labs, 1501 Page Mill Road, Palo Alto, CA.
to appear in *IEEE Trans. on Nanotechnology*

enough apart to be connected using conventional lithography, and has been used to create a functioning demultiplexer [9]. However this greatly increases the area of the circuit, thereby negating the advantage of nanoscale circuitry when applied to a large number of components.

A better approach, used by all existing memory chips, is the demultiplexer circuit, in which $M = 2 \log_2 N$ external addressing wires can individually address N data wires. Since only the small number of address wires need to connect to the micron scale, using demultiplexers with nanoscale circuits retains the density advantages of the nanoelectronics. Unfortunately, demultiplexers require a complex pattern of connections, which can't be fabricated at the nanoscale by current technology.

This paper presents a theoretical study of a technique to fabricate demultiplexers at the nanoscale using random self-assembly. We consider a regular self-assembly process to create an array of many parallel nanowires. Ordinary micron-scale lithography makes the small number of micron-scale address wires. Connections between these sets of wires are formed with two-way AND logic elements at their cross points. These connections are formed by a feasible random physical process. This process only allows control over the density of connections but not their precise location, thereby precluding the complex connection pattern of a standard demultiplexer. Such random connections are much easier to fabricate than precisely specified connections, e.g., by self-assembly methods. On the other hand, the randomness leads to many defects and uncertainties in the circuits. In this paper, we use theoretical modeling to investigate this trade-off.

This work extends the range of nanoscale self-assembly techniques beyond the regular structures described above for a conventional demultiplexer circuit. It thus points the way to developing complex nanoscale circuits by combining regular and random structures. We show that in spite of the randomness, with a sufficient number of extra wires, the resulting circuit has a high probability to correctly function as a demultiplexer, i.e., we can achieve high yield of correct circuits. The ability to fabricate nanoscale circuits with high yield but without precise control of individual connections relies on the appearance of sharp thresholds in yield as a function of the number of extra components included in the circuit. That is, instead of a gradual improvement in yield with an increasing number of extra components, we see an abrupt change from yields near zero to values near one at a specific threshold. This behavior, exhibited by our simulation results, is a general property of many statistical systems, including physical systems [14, 15], mathematical structures [16], biology [17], computation [18] and engineered systems such as wireless networks [19]. Such thresholds in performance enable reliable operation by detecting and adjusting to faults so as to exceed the threshold [8, 20, 21].

2 Demultiplexer Circuits

Electronic circuits often involve large numbers of devices but have limited space for external communication, e.g., via a limited number of pins on an integrated circuit. Demultiplexer circuits are often used to solve this problem by allowing a small number of wires to selectively address a larger number of data wires [22, 23].

For an example, Fig. 1 shows a demultiplexer circuit, in which six addressing wires (marked A, \dots, F) address eight data wires (marked $1, \dots, 8$). Each dot on an intersection between an addressing and data wire represents a two-way AND logic element (as shown in the insert in Fig. 1). A crossing with no dot indicates no connection between the wires. The two-way AND logic element can be a direct connection, a diode, or a transistor between the two wires. Selecting the type of connecting device is a choice between ease of fabrication and operation efficiency. For instance, resistor connections are the easiest to fabricate but suffer from increased power use and lower discrimination between “on” and “off” states compared to more complex devices.

As an application of this circuit, a single additional wire (not shown) connected to all the data wires at the top of the circuit would set all data wires to 1 at the top. An address wire whose signal is zero will set all its connected data wires to zero. Thus a data wire will only be set to 1 if the signals on *all* the addressing wires it connects to are 1. The circuit could be used to introduce external signals to exactly one of the data wires, e.g., as part of a write/read operation for a molecular memory.

With the pattern of connections shown in Fig. 1, the addressing wires group into pairs whose signals are always complementary, i.e., either 0,1 or 1,0. These addressing signals are readily achieved by connecting each pair to a single external communication wire, which branches into two addressing wires, presenting both the original signal and its inverse. Presenting any set of binary values for the three communication

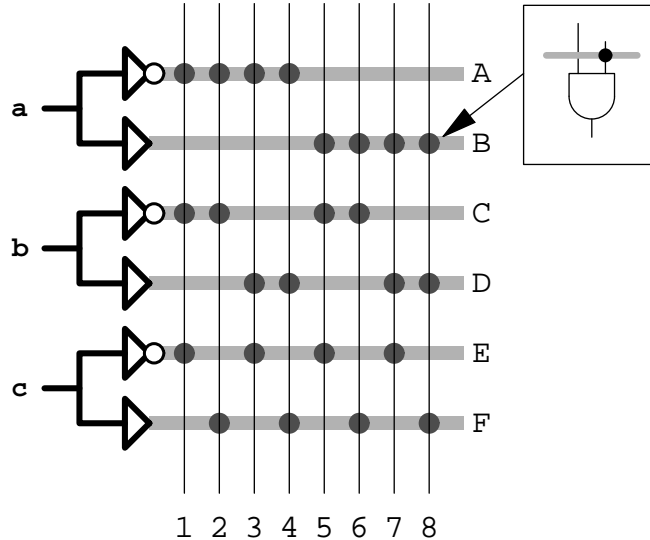


Figure 1: Demultiplexer circuit with 8 data wires ($1, \dots, 8$) and 6 addressing wires. The binary pattern of connections shown here has valid addresses consisting of alternating 0's and 1's, which can be provided externally by 3 wires (a, b, c), each connected to two addressing wires, one of which is via an inverter circuit. Inset: Logic element corresponding to a connection.

wires (marked as a, b , and c) addresses exactly one of the data wires.

For example, setting the signals 1, 1, 0 for a, b , and c , respectively, addresses the data wire 7. That is, only the signal on data wire 7 will appear to be 1 at the bottom of the circuit, while the signals from all other wires are blocked by one or more of the logic elements at the connections.

This small example, involving six addressing wires for 8 data wires gives only a small reduction in the number of wires. However, in general, such demultiplexer circuits allow M addressing wires to uniquely address $N = 2^{M/2}$ data wires. Thus we require only $M = 2 \log_2 N$ addressing wires, giving very large reductions as N increases. Thus demultiplexers provide an efficient interface to circuits with many components. By grouping pairs of addressing wires, the addressing signals can be delivered to the circuit with only $M/2$ external communication wires.

2.1 Fabricating Nanoscale Connections

While demultiplexers are commonly used for conventional microelectronic circuits, their use as an efficient interface for nanoscale circuits is more difficult. In this case, the addressing wires are most easily fabricated with conventional microelectronic patterning techniques, allowing them to easily connect with external circuits. This means the width of the addressing wires, e.g. at 100nm, is considerably larger than that of the nanoscale data wires, e.g. at 10nm or smaller. Thus the fabrication of the AND logic elements requires a precision in their size and location beyond the capabilities of conventional lithographic techniques. While nanoscale circuits can be fabricated on this small scale, it is not currently possible to create large numbers of such connections in the precise pattern required for the ideal demultiplexer illustrated in Fig. 1.

To illustrate this fabrication problem, Fig. 2 shows an expanded version of the demultiplexer of Fig. 1 with the addressing wires illustrated as having larger widths than the nanoscale data wires. To form valid addresses, the addressing wires are considered in pairs so that the signal on one wire in each pair is the opposite of that for the other wire in the pair. Note that the connections in the figure have sizes comparable to the spacing between the nanoscale data wires, and hence are too small to fabricate and position via lithographic techniques.

As a possible fabrication process to address this problem, we propose using randomly positioned connections within regions defined by larger masks or windows created lithographically [23]. Within the masked

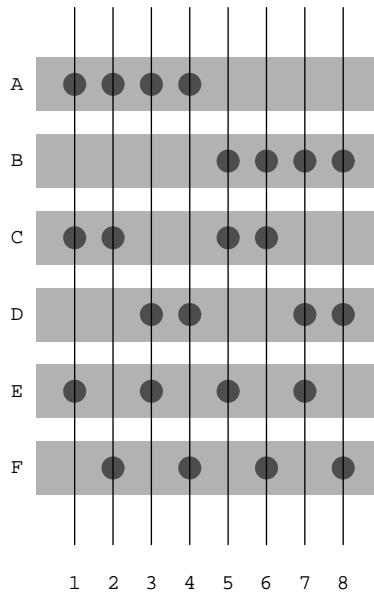


Figure 2: Ideal connection pattern for 8 data wires $(1, \dots, 8)$ and 6 addressing wires (A, \dots, F) .

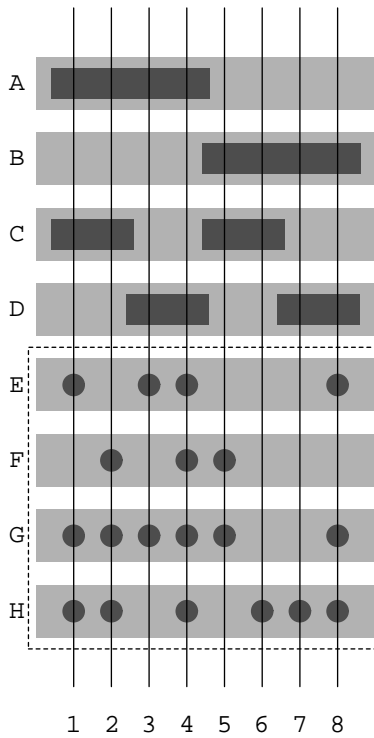


Figure 3: Schematic example of connections between 8 data wires $(1, \dots, 8)$ and 8 addressing wires (A, \dots, H) . Connections to the top four addressing wires are made in windows corresponding to the ideal connection pattern. For the lower four addressing wires, in the dashed box, random connections are made with probability $p = 1/2$. The components in this diagram are not to scale.

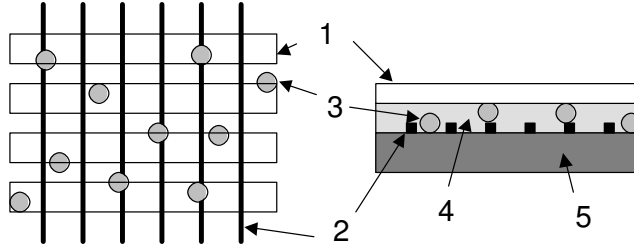


Figure 4: A schematic diagram (Left: top-view; Right: Cross-section) showing the random interconnections between addressing wires (marked as #1) and data wires (#2) by conductive nano-particles (#3) randomly distributed in insulative polymer thin film (#4) on a substrate (#5).

regions, this process self-assembles connections, with control over the density of connections but not their detailed placement. This results in two types of connected areas, as shown in Fig. 3. In this example, each wire has a distinct pattern of connections, but they are not all uniquely addressable. In particular, turning on data wire 4 (by setting address wires A, D and E to H to 1 and the rest to 0) also turns on data wire 3. Similarly, turning on data wire 8 also turns on wire 7.

In the masks used for the high order bits, we assume the boundaries of the masks are precise enough to fall between successive data wires, preventing any connections between a single data wire and two addressing wires part of a single pair (e.g., wires *A* and *B* in Fig. 3). If this level of precision is not possible for the masks, then data wires at the boundary between the masked regions could be connected to neither or both of the addressing wires in a pair. Such data wires will not be correctly addressable by the demultiplexer. For the most significant bits, the masking regions are quite large, so relatively few data wires will be near the boundaries. However, as the mask sizes decrease, more wires will be near the boundary of each masking region, leading to a larger proportion of potentially incorrect connections. If this is a problem, we can simply switch to using random connections for more of the addressing wires, eliminating the boundaries, and add additional addressing wires with random connections as necessary to achieve high probability of a functional circuit. As a further improvement, the mask regions can be expanded somewhat with suitable reordering of the desired connections, e.g., according to a Gray code [24] instead of the standard binary ordering shown in the figure.

Experimentally, it is much easier to establish random interconnections between data wires and addressing wires than accurately defined interconnections. The random connections can be done chemically using a chemical process which has a specific (e.g., 50%) chance of happening. One example of such a process is using a randomly-deposited, colloidal-sized (i.e., with a diameter of nanometers) dot to either make electrical contact or not between a nanowire and a microscopic address wire. These dots could consist of nanoscale conductive particles (such as gold or semiconductors). A set of particles with homogeneous nanoscale sizes can be prepared by colloidal self-assembly method [25]. Fig. 4 illustrates how these particles can be used to form the random connections. Specifically, these nano-particles can be mixed with an insulative polymer to form a thin film on the top of the data wires. The film thickness can be made comparable with the diameters of the nano-particles. When the addressing wires are fabricated on the top of the film, the data wires and the addressing wires will be electrically connected at their cross points where the nano-particles exist, otherwise, the data wires and the addressing wires will be electrically separated at their cross points by the insulative polymer. The ratio between the conductive to the insulative cross points can be controlled and adjusted by the density of the nano-particles in the polymer, although the location of the conductive cross-points can not be controlled, and therefore is randomly distributed.

Our analysis of the consequences of random assembly of nanostructures does not depend on the particular physical method of construction. For instance, DeHon et al. [12] have proposed an alternative physical mechanism based on random alignment of patterned nanowires.

To use a random assembly process, we simply introduce enough connections over the whole extent of the data wires so each data wire has probability p to connect to each of the addressing wires, where p can be selected with any value between 0 and 1. Thus we have control over the density of connections but not their

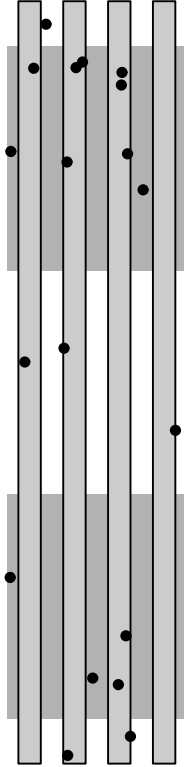


Figure 5: Connections between data and address wires shown with correct scale. The two horizontal gray areas are sections of two address wires, each of 100nm width and separated by 100nm. The vertical data wires have 10nm width and separation. Connections are shown with 5nm particles (black dots), corresponding to a connection probability of $p = 1/2$.

precise number or placement. Forming such random connections is a much simpler fabrication process than required to precisely position connections, and currently feasible for creating large numbers of nanoscale devices.

Fig. 5 illustrates the connections produced by this process using nominal sizes for the wires and connecting particles. A probability $p = 1/2$ of a connection means the contact area, A , between a data and address wire (which in this example is $10 \times 100 = 1000 \text{ nm}^2$) has probability 1/2 to have no nanoparticles. Considering the particles on a grid, this area could hold A/d^2 particles, where d is the diameter of the particles (5nm in this example). Thus if ρ is the probability a grid point is in fact occupied by a nanoparticle, we require $(1 - \rho)^{A/d^2} = 1/2$, giving $\rho = 0.017$. Thus the contact area will have, on average, $\rho A/d^2 = 0.69$ particles. This value is somewhat larger than 1/2 because we have an equal probability to have either no particles in the contact area or at least one particle, but possibly more. An equal weighting of these two possibilities results in an average value greater than 1/2. This density of particles corresponds to covering about 1% of the area: $0.69\pi(d/2)^2/A = 0.013$. Thus the relatively large width of the address wires implies a relatively sparse distribution of nanoparticles to achieve connectivity of $p = 1/2$.

This process will occasionally produce multiple connections between a data and address wire, in which case it is important that the connection device properties are such that one or more have the same behavior, e.g., a low enough resistance to switch signals. If this is not the case, the resulting variation in the connections alters the threshold locations as discussed in Section 3.3. Similarly, particles near the edge of a data wire could result in partial connections, adding to this variation.

For larger connection densities (i.e., p close to 1), the nanoparticle area coverage will eventually be large enough for the nanoparticles themselves to come into contact quite often, thereby forming additional

conducting paths. At high enough densities, these paths extend over long distances, forming additional conductive links between adjacent addressing wires. A few such connections would simply add a bit to the conducting paths between addressing wires already formed by their connections to the data wires, and thereby have limited effect on the functioning of the circuit (though would increase power drain). A large number of such connections would form a short between the address wires, preventing their independent use. Alternatively, because the gaps between address wires are large enough to access lithographically, conventional technology could be used to place additional barriers in the gaps to limit or remove conductive nanoparticle paths between address wires.

2.2 Using Randomly Placed Connections

Using random connections means the pattern of connections for the less significant bits in each address will not follow the ideal pattern illustrated in Fig. 1 for a demultiplexer design. Thus the addresses will not necessarily be correct. To compensate for this limitation, we introduce additional addressing wires, also with random connections. Unlike those involved in the ideal demultiplexer, the addressing wires corresponding to these random connections will not necessarily be used in such a way that successive pairs always use opposite polarities. Thus these addressing wires must be separately available for external connection rather than the situation shown in Fig. 1 in which they are grouped into pairs.

With this technique, the proportion p and number of addressing wires M should be appropriate to give functional circuits with high probability. In this context, a functional circuit is one in which it is possible to uniquely address enough data wires to perform the desired function for the circuit. From a statistical viewpoint, we can evaluate the resulting behavior from two viewpoints. First, we may require access to all the data wires, in which case the appropriate criterion is the probability the demultiplexer gives a unique address to each data wire. Second, the circuit may have some spare data wires, beyond the number actually needed to perform its function. In this case, we can examine the fraction of data wires with a unique address, with those wires without unique addresses removed from use when the circuit is tested after fabrication.

The high order address bits, which are correctly connected, give unique addresses to groups of wires. So if the patterning for only the last b bits is too small to create with the masks, we only need enough additional addressing wires to disambiguate among groups of 2^b nanowires rather than all of them. In the next section, we see that relatively few additional addressing wires can, with high probability, make fully unique addresses in spite of the random connections.

3 Thresholds with Random Connections

In this section we examine the requirements for forming functional circuits with random connections. We do so in two ways: via simulation using randomly generated samples, and via approximate analytic estimates of the behavior.

3.1 Correct Circuits

For the simulation with N data wires and M addressing wires, we generate a connectivity matrix in which each element is independently selected to be 1 with a specified probability p . We then compare the addresses of each pair of data wires (i.e., columns in the connectivity matrix) to see whether using one of the addresses would select both wires. If so, the address is not unique. Performing this check on each pair determines the number of uniquely addressable data wires for a particular connectivity matrix.

For example, a possible matrix with $N = 4$ and $M = 3$ is

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

The two 1's in the first row indicate the first addressing wire connects to the 2nd and 3rd data wires. Comparing the first and second columns, we see that turning on the connections for the second data wire also turns on those for the first. Similarly, the last column address also activates the first. Thus this set of connections only allows uniquely addressing two of the four data wires, namely the first and third.

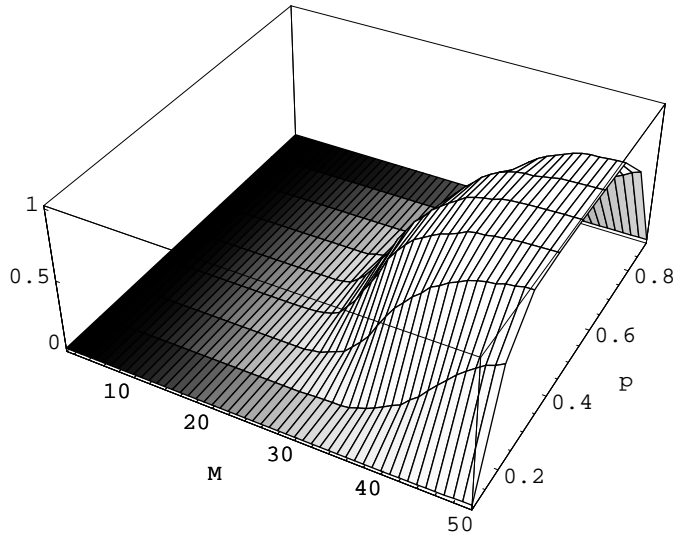


Figure 6: Probability all addresses are unique as a function of number of addressing wires M and random connection probability p for $N = 64$ data wires. The shading indicates the average fraction of wires with unique addresses, ranging from 0 (black) to 1 (white).

More generally, we can view each address, i.e., a column of the connection matrix, as specifying a set of connections. If column I is a superset of column J , i.e., column I has a value 1 at least at every position that J does, then addressing I will also address J , so I is not uniquely addressable. (An exception is if column J has no connections at all: but then wire J is not addressable so even in this case we do not have uniquely addressable wires.) As the number of addressing wires, M , increases, the chance two addresses conflict in this way decreases. Thus we can expect that sufficiently large M will give unique addresses with high probability. On the other hand, increasing the number of data wires, N , makes it more likely that at least two of them will conflict, in which case the system will not be fully addressable.

In fact, when dealing with a large number of data wires, the probability for unique addresses exhibits a sharp threshold with relatively few addressing wires. To see this, we created numerous randomly generated connectivity matrices and recorded the proportion with unique addresses for all wires.

As an example, Fig. 6 shows the threshold behavior for $N = 64$ data wires as a function of number of addressing wires and probability to make a connection, p . The vertical axis shows the probability for all data wires to have unique addresses, and the shading indicates the average fraction of wires with a unique address (i.e., the expected proportion of uniquely addressable wires).

The most important qualitative feature of the figure is the abrupt change between a region with very low probability for unique addresses and a second region with probability near one. Thus, if we select p and the number of addressing wires at least a bit above the threshold, we are very likely to create circuits with unique addresses for all wires. Since the probability is not quite equal to 1, there will remain a small fraction of circuits that will be defective. Thus for a given acceptable defect tolerance (i.e., proportion of circuits that are not fully addressable), this sampling procedure readily gives appropriate choices for M and p .

Another observation from Fig. 6 is how the threshold location varies with the probability of connections. We see poor performance for both small and large values of p . This is readily understood by considering the extreme cases: when $p = 0$ there are no connections at all, and when $p = 1$ all wires have the same address so none are uniquely addressable. The threshold is lowest for $p = 1/2$ (i.e., that probability requires the fewest addressing lines to obtain a high probability of unique addresses for all data wires). For p values near $1/2$, the threshold location grows quadratically, i.e., proportional to $(p - 1/2)^2$. Thus for p values near $1/2$, the threshold location is close to the best value. Hence the p value need not be very precise to get close to the lowest possible threshold, simplifying the construction of the circuit.

Finally, the shading in Fig. 6 shows a distinctly smaller number of address wires is sufficient to give

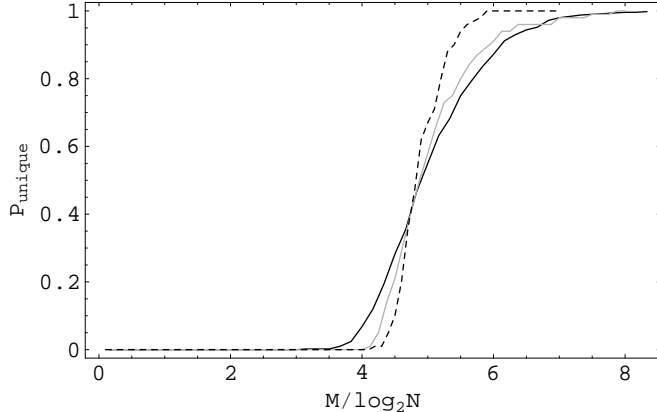


Figure 7: Threshold scaling. Probability all addresses are unique as a function of $M/\log_2 N$ for connection probability $p = 1/2$. The curves correspond to $N = 64, 256$ and 1024 for black, gray and dashed curves, respectively.

unique addresses to most of the data wires than is required to ensure *every* wire is unique. Moreover, even some cases with very small probability for all wires addressable still have a substantial fraction of uniquely addressable wires.

For this example, the number of addressing wires to reach the threshold is around 40, not much smaller than the number of data wires (64 in this case). Nevertheless, the demultiplexer design scales well: the number of addressing lines required to reach the threshold grows only logarithmically with N .

To see this scaling for $p = 1/2$, Fig. 7 shows the probability for all data lines to have unique addresses as a function of $M/\log_2 N$. With this scaling, the probability curves overlap near 5 giving the threshold at $M \sim 5 \log_2 N$.

To understand why probability of connections close to $1/2$ works best, we give an analytic estimate of how the threshold depends on p . When connections are completely random, the combinatorial expression for the probability for unique addresses is complicated. Nevertheless, we can estimate the location of the threshold by making the assumption that each pair of wires can be considered independently. Consider the connections on two data wires, i and j . For each of these to be separately addressable, there must be at least one addressing wire connecting to wire i but not wire j , and at least one other addressing wire with the opposite connections. The probability an addressing wire connects to wire i but not j is $\alpha = p(1 - p)$, and similarly for connecting to j but not i . Since the connections for each wire are made independently, for M addressing wires, the probability none connects to i but not j is $(1 - \alpha)^M$. Similarly for having none connecting to j but not i . Finally, the probability for neither type of connection is $(1 - 2\alpha)^M$. Thus the probability this pair, at least, is correctly connected is

$$P_{\text{pair}} = 1 - 2(1 - \alpha)^M + (1 - 2\alpha)^M \quad (1)$$

For all wires to be uniquely addressable, every pair must be correctly connected. Under the approximation assuming each pair's correctness is independent of the others, the overall probability for unique addresses for all data wires is $P_{\text{unique}} = P_{\text{pair}}^{\binom{N}{2}}$. For $0 < p < 1$, we have $0 < \alpha < 1/4$ with the maximum at $p = 1/2$. To identify the threshold under this approximation, we consider the behavior when M and N are large. In this case $P_{\text{unique}} \sim \exp(-\binom{N}{2} 2(1 - \alpha)^M)$. The threshold $P_{\text{unique}} = e^{-1}$ then corresponds to $N^2(1 - \alpha)^M \sim 1$ or

$$M \sim \frac{-2 \ln 2}{\ln(1 - \alpha)} \log_2 N \quad (2)$$

The coefficient is smallest when $p = 1/2$, in which case it becomes $M \sim \frac{-2 \ln 2}{\ln(3/4)} \log_2 N = 4.8 \log_2 N$. This expression is quite close to the observed threshold based on random samples in Fig. 7, so the independence assumption is fairly accurate in this case.

As an alternative showing how the precise nature of the distribution of random connections affects the threshold, consider the situation in which each data wire is connected to *exactly* h of the M addressing wires, but the choice of these connections is random. In this case, there are $T = \binom{M}{h}$ ways to pick the connections for a wire, each of which is equally likely. With N data wires, the addresses can be selected in T^N ways, but they are all unique in only $T(T-1)(T-N+1)$ ways. Thus the probability the addresses are unique is the ratio of these quantities:

$$P_{\text{unique}} = \left(1 - \frac{1}{T}\right) \dots \left(1 - \frac{N-1}{T}\right) \quad (3)$$

For a given number of wires N , the larger the number of choices T , the higher the probability all the addresses are unique. Since the binomial coefficients are largest in the middle of their range, T is maximum when $h = M/2$, i.e., when half the connections are made. This corresponds with the value $p = 1/2$ seen to give the lowest threshold in the fully random case.

This analysis also shows the basis for the logarithmic scaling. Specifically when the number of possible addresses is much larger than the number of data wires ($T \gg N$), $P_{\text{unique}} \sim e^{-N^2/(2T)}$. Defining the threshold location as the number of addressing wires for which $P_{\text{unique}} = e^{-1}$, the threshold corresponds to $T = \frac{1}{2}N^2$. Stirling's approximation for the binomial coefficient gives T growing as $T \sim e^{MH(h/M)}$ where $H(x) = -x \ln x - (1-x) \ln(1-x)$ is the entropy function. Thus the threshold condition becomes $M \sim \frac{2}{H(h/M)} \ln N$. The smallest threshold, corresponding to the largest value of the entropy, is when $h = M/2$, giving $M \sim \frac{2}{\ln 2} \ln N \sim 2 \log_2 N$.

This discussion not only shows why the number of addressing wires required to reach the threshold grows only logarithmically, but also indicates the difference between two methods producing random connections:

- random connections constrained to have exactly the same number of connections per data wire, and
- random connections made completely independently, i.e., with no constraint.

Examples of these constrained and independent random cases appear in references [12] and [23], respectively. The thresholds for these cases are at $2 \log_2 N$ and about $5 \log_2 N$, respectively. Thus, even though both methods give the same number of connections on average, the variation in the number of connections in the second case increases the likelihood of duplicate addresses. This difference shows the potential for reducing the number of addressing wires needed to pass the threshold by improved fabrication techniques that reduce the variation in number of connections made to each wire.

As an example of this difference, consider $p = 1/2$ and $N = M = 2$. In the first case, of constrained randomness, both addresses would have exactly $Mp = 1$ connection. Since the system's behavior is unchanged if the addressing wires are reordered, for simplicity we can consider only cases in which the connection to the first data wire is on the second addressing line so the possible connection matrices (each equally likely) are

$$\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

The first connection matrix does not have unique addresses while the second does. So the probability such a system is uniquely addressable is $1/2$.

By contrast, in the second case, of fully random connections, the data wires can have 0, 1 or 2 connections, with probabilities $1/4$, $1/2$ and $1/4$, respectively. When the first wire has no connections, it is not addressable so this case always fails to be uniquely addressable. Similarly, if the first wire has both connections, then all choices for the second wire will be a subset of that for the first: again the system is not uniquely addressable. Finally, if the first wire has a single connection, combining the cases that differ only by a permutation of the addressing wires gives the possible matrices as

$$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

each equally likely. As mentioned above, the second wire can have zero, one or two connections, as illustrated with these possible connection matrices. Of these, only the third gives unique addresses for both wires. Considering all these possibilities, the probability such random connections give unique addresses is only $\frac{1}{2} \times \frac{1}{3} = \frac{1}{6}$, considerably smaller than when each wire is constrained to have exactly one connection.

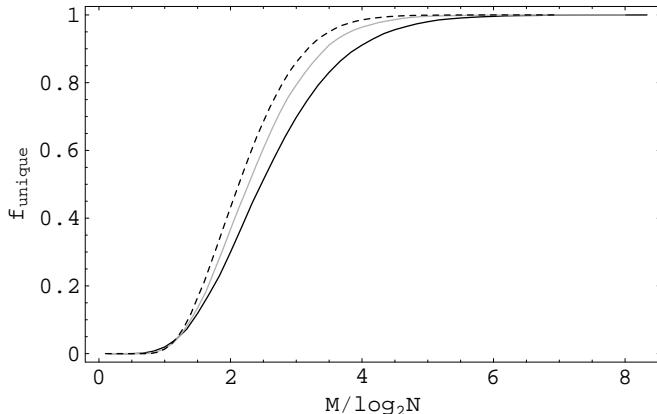


Figure 8: Fraction of uniquely addressable data wires as a function of $M/\log_2 N$ for connection probability $p = 1/2$. The curves correspond to $N = 64$, 256 and 1024 for black, gray and dashed curves, respectively.

3.2 Number of Functional Components

The previous discussion examined conditions under which all data wires in the circuit have unique addresses. In some applications it may be sufficient to allow a few wires that cannot be addressed. For such circuits, a fault-tolerant architecture [8] could identify the defective wires and then avoid using them. In this situation, we need to determine the likely number of working components in the circuits. This would allow making enough extra components so that, even after removing defects, there remain a desired number of functioning components. Depending on the fabrication technology and circuit application, this approach may be better than insisting on circuits with no defects, e.g., all data wires uniquely addressable by the demultiplexer.

With random connections, the fraction of uniquely addressable wires has a threshold at a somewhat lower value than that required for *all* addresses to be unique. For instance, Fig. 8 shows the fraction of uniquely addressable wires as a function of $M/\log_2 N$ when $p = 1/2$. The threshold becomes sharper as N increases. Thus, for example, to have about 80% of the wires uniquely addressable needs $M \sim 3 \log_2 N$.

The above approximate analysis for the threshold in P_{unique} applies in this case as well. For instance, consider two data wires. The probability that pair is correctly connected, P_{pair} , is given in Eq. (1). Assuming independence, a given data wire is uniquely addressable when all its pairings with other wires are correctly connected, i.e., with probability P_{pair}^{N-1} . Thus the expected number of uniquely addressable wires, under this approximation, is $N P_{\text{pair}}^{N-1}$. For large M, N this is approximately $N \exp(-2N(1-\alpha)^M)$. The threshold giving, say, a fraction e^{-x} of uniquely addressable wires then corresponds to $2N(1-\alpha)^M \sim x$ or

$$M \sim \frac{\ln 2}{-\ln(1-\alpha)} \log_2 \frac{2N}{x} \quad (4)$$

When $p = 1/2$ and a fraction $e^{-x} = 0.8$, this becomes $M \sim 2.4 \log_2 \frac{2N}{-\ln 0.8} = 2.4 \log_2(9.0N)$. This approximately matches the actual values from the samples as shown in Fig. 8. This analysis again shows the lowest threshold corresponds to $p = 1/2$.

3.3 Reliable Operations with Variation in Connections

In the previous sections, we showed how additional addressing wires produce uniquely addressable demultiplexer circuits with high probability. Moreover, the required number of additional wires grows only logarithmically with the number of data wires. Beyond this question of fabricating correct molecular circuits, there is the issue of their sensitivity to noise arising during operation or from variation in circuit element characteristics, e.g., the connector resistances. In the context of the demultiplexer, this amounts to the difference in voltages between the addressed data wire and all other wires. Ideally, the addressed wire would be fully on, corresponding to a value 1, while all other wires would be off, i.e., a value 0. In this case, variation due to noise of less than $1/2$ would allow reliable identification and use of the intended data wire.

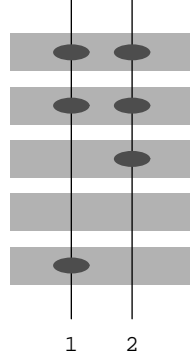


Figure 9: Example connections to two data wires, 1 and 2.

This ideal switching requires that changing even a single connected address wire from 1 to 0 is sufficient to fully switch the data wire’s output. The actual separation in output values for the data wires depends on the type of device forming the connection. If the interconnecting AND logic element (shown in the insert in Fig. 1) consist of high-quality diodes, the switching can be close to the ideal. However, such devices are difficult to build at a molecular scale, and instead they will likely have some leakage current. Even more problematic are connections made of resistive elements, which are less efficient but simpler to fabricate. Here we consider the consequence on the demultiplexer of using such resistors as connections.

Linear circuit elements are not ideal switches. For instance, consider two data wires 1 and 2 shown in Fig. 9. Suppose wire 2 has x resistive connections to address wires that do not also connect to wire 1, wire 1 has y not shared with wire 2 and they have z connections in common (e.g., in Fig. 9, $x = 1$, $y = 1$ and $z = 2$). Activating the address for wire 1 will also turn on z out of $x + z$ address wires connected to wire 2, so its output will be $z/(x + z)$ instead of its ideal value of 0. Similarly, activating the address for wire 2 gives output $z/(y + x)$ for wire 1. For these wires to have distinguishable output with a level of noise s (between 0 and $1/2$), the difference in their outputs in these two situations must be greater than s , i.e.,

$$\begin{aligned} 1 - \frac{z}{x + z} = \frac{x}{x + z} &> s \\ 1 - \frac{z}{y + z} = \frac{y}{y + z} &> s \end{aligned} \tag{5}$$

When $s = 0$, these conditions reduce to x and y both greater than 0, which is the same as the criterion for uniquely addressable data wires discussed above.

For a given value of s and a set of connections, testing this criterion for each pair of data wires determines whether the pattern of connections is robust with respect to this level of noise. By generating a sample of randomly connected circuits with given numbers of data and addressing wires, we can then determine P_{dist} , the probability all data wires have distinguishable addresses, as well as the fraction of wires with such addresses. As the number of addressing wires increases, the addresses for each pair become more likely to have many different connections, and hence a large difference in outputs. Thus we can expect that increasing M also increases the distinguishability of the addresses.

As an example of the effect of noise, Fig. 10 shows the probability all addresses are sufficiently different to tolerate a level of noise ranging from $s = 0$ to 0.4, for the case of $N = 64$ data wires and random connections made with probability $p = 1/2$. The shading indicates the fraction of data wires whose addresses are sufficiently different from all others to have distinguishable output even with the given level of noise. The curve for $s = 0$ is the same threshold examined previously for unique addresses. As the noise level increases, the probability of distinguishable addresses drops, but nevertheless can still be fairly large with a sufficient number of additional addressing wires. However, as the noise level increases beyond 0.1 or so the required number of additional wires grows rapidly.

To quantify the shift in the threshold location due to noise for the resistive circuits, Fig. 11 shows the number of address wires M required to have the probability for correct circuit reach e^{-1} , for noise thresholds

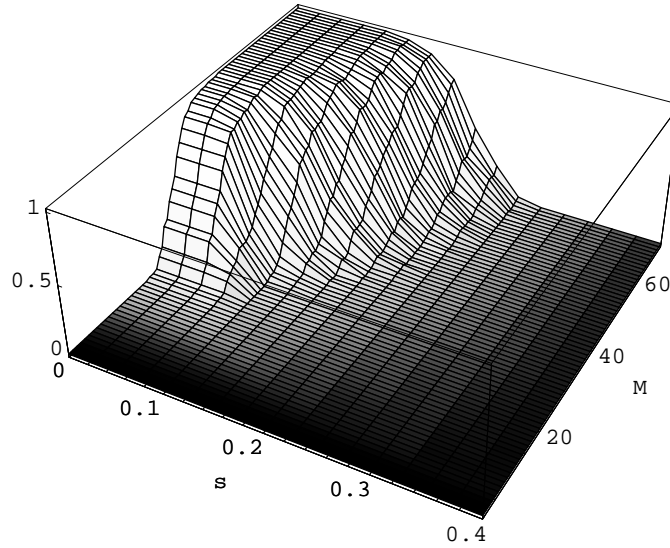


Figure 10: Probability all addresses are sufficiently different to tolerate a level of noise s with M addressing wires for $N = 64$ data wires and random connections with probability $p = 1/2$. The shading indicates the fraction of data wires with distinguishable addresses, ranging from 0 (black) to 1 (white).

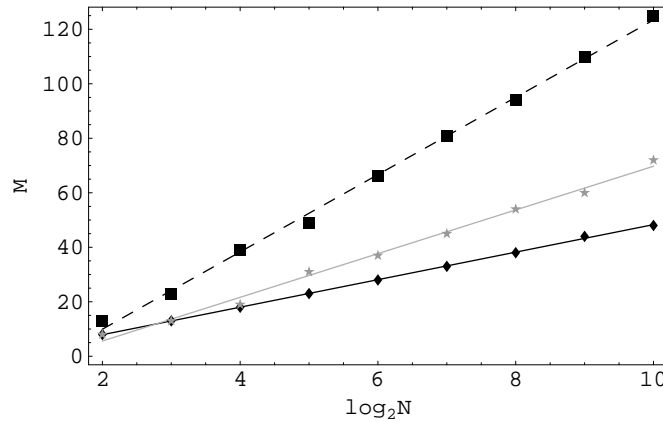


Figure 11: Number of address wires M required to have the probability of a correct circuit exceed e^{-1} as a function of $\log_2 N$ with connection probability $p = 1/2$. The values are for increasing levels of noise: $s = 0$, 0.1 and 0.2 as the black, gray and dashed curves, respectively. The lines are linear fits to the values.

0, 0.1 and 0.2, based on 100 samples for each point. While the required M grows faster with increasing noise, the growth remains linear in $\log N$, i.e., the required number of addressing wires grows only logarithmically with the number of data wires. The lines are linear fits to the points with slopes 5.0, 8.0 and 14.2, respectively. Thus random connections with additional address wires is effective not only in creating unique addresses, but also making each address sufficiently different from the others to ensure considerable difference in the output, even with simple resistive connections.

The logarithmic growth in M can be understood by an argument similar to that given above for unique addresses. That is, we assume each pair of data wires is independent and express the M -dependence of the probability a given pair is distinguishable in the form $P_{\text{pair}} = 1 - CA^M$ where C and A are values independent of M and this expression is valid for large M . The probability, ρ , the two data wires have a pattern of connections described by the values x, y, z used in the discussion of Fig. 9 is a multinomial distribution. For the particular case of uniform random connections, i.e., $p = 1/2$, this distribution is

$$\rho = 4^{-M} \binom{M}{x, y, z, M-x-y-z} \quad (6)$$

P_{pair} is the sum of this multinomial over all choices of x, y, z that satisfy Eq. (5).

For large M , Stirling's approximation for the multinomial gives

$$\rho \propto \exp \left(M \left(H \left(\frac{x}{M}, \frac{y}{M}, \frac{z}{M} \right) - \ln 4 \right) \right) \quad (7)$$

where H is the entropy: $H(u_1, u_2, u_3) = -\sum_{i=1}^4 u_i \ln u_i$ with $u_4 = 1 - u_1 - u_2 - u_3$, with maximum value $\ln 4$ when all arguments equal $1/4$. In the expansion $P_{\text{pair}} = 1 - CA^M$ the value of A is given by the maximum of $\exp(H - \ln 4)$ over the range of x, y, z values giving indistinguishable outputs for the pair, i.e., $y/(y+z) \leq s$ or $x/(x+z) \leq s$. This constrained maximization of H gives

$$A = \frac{1}{2} + \frac{(1/s - 1)^s}{4(1 - s)} \quad (8)$$

which has the limits $3/4$ and 1 for $s = 0$ and $1/2$, respectively.

Assuming each pair's correctness is independent of the others, the overall probability for distinguishable addresses for all data wires is $P_{\text{dist}} = P_{\text{pair}}^{\binom{N}{2}}$. To identify the threshold under this approximation when M and N are large, $P_{\text{dist}} \sim \exp(-\binom{N}{2}CA^M)$. The threshold $P_{\text{dist}} = e^{-1}$ then corresponds to $N^2A^M \sim 1$ or

$$M = \frac{2 \ln 2}{-\ln A} \log_2 N + O(1) \quad (9)$$

which is linear in the logarithm of the number of data wires. The value of the constant C used in the above large- M expression for P_{pair} does not contribute to the leading term in this expression for M .

Using the above expression for A in this threshold for M gives the coefficient of $\log_2 N$ as 4.8, 8.3 and 15.1 for $s = 0, 0.1$ and 0.2 , respectively. These values are within about 10% of the slopes shown in Fig. 11 based on random samples. Furthermore, as the noise level increases toward $1/2$, A approaches 1 so the threshold coefficient grows to infinity, i.e., the required number of additional wires becomes very large.

As a final observation, if the connection devices are intermediate in behavior between resistors and ideal diodes, the switching behavior will be intermediate as well, thereby allowing reliable operation with a threshold between that for unique addresses (which assumed ideal switching or no noise) and that for resistive elements discussed in this section.

4 Identifying Correct Addresses

We discussed fabricating reliable circuits with some random connections. After fabrication, such circuits must be tested to determine which addresses are functional and the actual number of data wires available for use. For the demultiplexer, this amounts to finding which of the 2^M addresses correspond to unique data wires, i.e., the actual connections made between address and data wires. We show that this can be done by

testing the circuit in time proportional to $N \log N$, followed by a typically much more rapid computational step with $O(N^2)$ operations that do not involve physically manipulating the circuit. For definiteness, we describe the procedure for $p = 1/2$, so each address wire is likely to connect to about half of the data wires, and assume $M > \log_2 N$ (for the performance thresholds we identified, in fact M will typically be several times larger than $\log_2 N$).

The procedure makes use of a *testing* wire perpendicular to the data wires, and forming a junction with each of them. This testing wire can be fabricated lithographically, i.e., similar to the address wires. All data wires are initially configured to be connected to this testing wire (i.e., the junctions form logical OR operations). Similarly, a *power* wire connects to all data wires to provide the input current required for operating the demultiplexer as well as for the address finding algorithm described in this section. The testing and power wires are analogous to adding two additional addressing wires to the circuit, such as the configurations illustrated in Fig. 2 and 3, but with connections to all of the data wires.

The testing wire is connected to a sense amplifier able to measure the current on the wire. We assume this amplifier can distinguish among currents through 1) zero data wires, 2) exactly one data wire, and 3) two or more data wires. This assumption is more stringent than that needed just for operating the demultiplexer: during operation the circuit must correctly select one data line due to a change on a single address bit, i.e., distinguish between zero and one active wire. If the additional requirement is too stringent, i.e., the hardware is unable to distinguish between one and more than one active wires, the testing procedure described below will still find addresses for most of the data wires, with high probability. This is because the events leading to two or more active wires is quite rare, as we see in simulations of the testing algorithm, discussed below, that quantify its expected running time.

For simplicity in describing the algorithm, we further assume the sense amplifier has sufficient dynamic range to be able to handle the current from all N data wires being ON. As a practical matter, we can accommodate amplifiers with a limited dynamic range by grouping the N data wires into sets of size L , where L wires being ON is within the dynamic range of the sense amplifier. That is, we test whether wires have been addressed by measuring the total current through L wires. This procedure requires N/L separate tests, but allows this algorithm to work with large numbers of data wires despite limited dynamic range of realistic components.

The address finding procedure consists of repeatedly using the address wires to select a random data wire and determine the connections to that wire. In the simplest hardware implementation, the junctions to the power and testing wires are fixed connections. In this case, the repetitions of the algorithm may find the same data wire multiple times, thereby requiring a large number of repetitions to find addresses for all the data wires. For example, after about half the addresses are found, we will require two repetitions, on average, to find a new data wire. In general, after finding $n < N$ addresses, the next trial finds a new wire with probability $1 - n/N$. Thus, on average, $N/(N - n)$ repetitions will be required to find a new wire. The expected number of repetitions to find all the addresses is then

$$\sum_{n=0}^{N-1} \frac{N}{N-n} \sim N(\ln(N) + O(1)) \quad (10)$$

compared to just N repetitions if each trial always found a new data wire. Most of the contribution to this additional $\ln N$ factor arises from finding address for the last few data wires. If instead we are satisfied with finding a fraction $f < 1$ of the addresses, then the sum ranges up to $fN - 1$ with asymptotic value growing as $-N \ln(1 - f)$, thereby giving an $O(1)$ overhead from extra repetitions instead of $\ln N$ to find all addresses.

We can find addresses more efficiently with a more complex hardware design: fabricating configurable junctions for either the power or testing wire. In particular, we consider junctions that can be altered from acting as a logical OR to acting as an insulator when suitable voltages are applied to individual junctions. Such a change results in disconnecting the selected data wire from the power or testing wire. Configurable junctions of this sort form the basis of memory and logic elements in crossbars of nanoscale wires [8, 9, 10, 11]. However, the junctions required for the address finding procedure are simpler in that they need only be configured once, namely to disconnect a data wire after its address has been found so it will not be found again. With this hardware design each repetition finds a new data wire.

Whether we use configurable junctions or not, the address finding algorithm operates in the same way. We start with all data wires active (i.e., with applied power). We first check for any data wires with no

connections to the address wires by setting all addresses to OFF. This will deselect data wires connected to at least one address wire, i.e., prevent them from delivering current to the testing wire. If current remains on the testing wire, then some data wires have no connections at all to the address wires. Such disconnected data wires are not addressable. If we have configurable junctions, these wires can be removed from consideration in subsequent trials of the algorithm (and from further use in the circuit) by configuring their junctions to disconnect them. Otherwise, we can declare the circuit a failure and discard it. These discards do not have much affect on yield because this situation is extremely rare when the number of address wires M exceeds the thresholds described above. Specifically, the probability a data wire has no connections is 2^{-M} so the probability all N data wires have at least one connection is $(1 - 2^{-M})^N$. In the case of the thresholds, $M = a \log_2 N$ with $a > 1$, so this probability becomes $(1 - N^{-a})^N$. For large N , this is approximately $\exp(-N^{-a+1})$ which rapidly approaches 1 as N increases because $a > 1$. For example, with $N = 64$ and M near the threshold for unique addressing, $M = 5 \log_2 N$, less than one in a million circuits will have any data wires with no connections.

The algorithm then proceeds as a series of trials. Let $N_{\text{available}}$ be the number of *available* data wires, i.e., those connected to the amplifier via the testing wire. Initially, all data wires connected to at least one address wire are available, i.e., typically the trials start with $N_{\text{available}} = N$. With configurable junctions, each time a new address is found, the data wire is disconnected so the number available decreases by one. Otherwise, all data wires continue to remain available and the trials could find the same wire multiple times as described above.

Each trial attempts to find the address of a single data wire as follows. We first turn all address wires to the ON state. Then we randomly choose address wires to turn OFF, one at a time, until the output current goes to zero (indicating all data wires are now deselected). The output will always eventually go to zero, after selecting some number K of addresses, since we have already removed connections to any data wires with no address connections or discarded circuits with such wires. Since each address wire is connected to about half the data wires, setting it to OFF will turn off about half of the available data wires. Thus we have $K \approx \log_2 N_{\text{available}}$ when all data wires are turned off. The last address wire selected before the output current went to zero is then changed back to the ON voltage, turning back on one or more data wires. If current from more than one wire is detected, the trial ends in failure and we continue with a new trial. Otherwise we now have K address wires selecting just one data wire. We now know part (K bits) of the address assigned to this single nanowire, namely 0 for the first $K - 1$ address wires and 1 for the last wire. With these address wires frozen in their states, each of the remaining address wires is then tested to see if it is connected to the data wire by testing each in the ON and OFF mode and observing whether current is being drawn or not. These correspond to bit values 1 and 0 in the address of the data wire. At the conclusion of this trial, if we have configurable junctions included in the circuit, we configure the junction for the single active data wire to disconnect it, preventing it from being found in a subsequent trial and reducing the number of available data wires, $N_{\text{available}}$, in subsequent trials by one.

Trials are repeated until all available data wires have assigned addresses or until some specified upper bound on the number of trials is reached, e.g., at most $3N$ trials to allow some repetition required due to failed trials. An upper bound is needed in case some wires do not have unique addresses, since otherwise trials would continue indefinitely, always failing when attempting to assign an address to wires without a unique address.

A final computational step sorts the list of addresses and checks for addresses that activate multiple data wires. This must not only check for pairs of identical addresses, but also for an address that is a subset of another, as described in Section 3.1. With M above the threshold for unique addressability, any such pairs are unlikely. As a practical matter, this check of the addresses can proceed rapidly as it does not involve any physical manipulation of the circuit (e.g., it can be performed while the circuit is in a warehouse or in transit). For example, with $N = 1024$ and $M = 5 \log_2 N = 50$ it merely involves comparisons among 1024 50-bit numbers.

We estimate the cost of this algorithm primarily by the number of address tests it requires of the circuit. A trial involves testing $K \approx \log_2 N_{\text{available}}$ wires if it fails to find a unique data wire, and all M address wires for successful ones. From the performance thresholds discussed above, we typically have M equal to several times $\log_2 N$. Thus a failed trial requires fewer tests than a successful one, especially with configurable hardware so $N_{\text{available}}$ decreases as addresses are found. To estimate how often a trial fails, suppose after turning off some number of address wires, n data wires remain on, i.e., are not connected to any of those

address wires. The next selected address wire has probability 2^{-n} of being connected to all of them, and hence turning them all off. With probability $1 - 2^{-n}$, the next selected address wire will instead leave at least one data wire still on, thereby requiring additional selection. Thus the last selected address wire will likely turn off just a few data wires, with highest probability for turning off just one. We obtain a rough estimate of the fraction of failed trials by noting a wire turns off one data wire with probability $1/2$ and 2 or more with probability $\sum_{n=2}^N 2^{-n} \approx 1/2$ for large N . Thus we can expect about half the trials will fail. This estimate does not account for the changing likelihood of various values of n as address wires are selected, but nevertheless gives a simple approximation.

The estimate of failure for half the trials gives the total number of tests as roughly $N(K + M) \approx N(\log_2 N + M) \leq 2NM$. It will successfully find the addresses associated with the data wires provided they are unique. As described above, the threshold for unique addresses requires $M = O(\log N)$ and hence a testing time of order $N \log N$. In case some data wires have no connections to address wires or are not uniquely addressable, the algorithm will fail to assign them addresses and they will not be useable with the circuit. Due to the upper bound on the allowed number of trials, the algorithm could also fail to find addresses for a few data wires that do in fact have unique addresses, especially if those addresses share many bits with those of other wires. These will be rare when above the threshold for unique addresses and have the minor effect of somewhat reducing the number of available data wires. Applying the algorithm to randomly generated circuits with $p = 0.5$ and $M = a \log_2 N$ with $a = 5$ and $a = 6$, we find the total number of tests is about $1.06NM$. Hence actual performance is slightly better than implied by the estimate $N(\log_2 N + M) = (1 + 1/a)NM$ given above. One reason for this difference is we observe that only about $1/3$ of the trials fail. Moreover, trials that fail tend to do so with somewhat smaller values of K than those that succeed. At any rate, failed trials add insignificantly to the total number of tests required.

The final computational check on the addresses does not require physical testing of the circuit, but instead only involves computational operations on the address table determined from the testing procedure. In general, a simple pairwise comparison of the addresses will involve $O(N^2)$ computation steps. Specifically, with $p = 1/2$, the addresses will each involve about $M/2$ bits, with variation around that number of order \sqrt{M} . In a simple testing procedure, we can sort the addresses by their size (i.e., number of bits they contain) and within each size by corresponding binary value, an $O(N \log(N))$ procedure. A $O(N)$ scan of the sorted table will identify any duplicates. To check for subsets, we can compare each address A with all those of larger size. That is to compare address A with one of larger size, B , we consider each bit in A . If B does not have that bit (which occurs in about half the cases when $p = 1/2$), then B is not a superset of A and the comparison is over. Otherwise we continue with the remaining bits of A , stopping whenever B is found not to have the corresponding bit. On average, this comparison will complete after about 2 checks (since $\sum_{i=1}^M i2^{-i} < 2$).

A system memory of about NM bits can store the actual addresses of the N data wires. These addresses can be stored in a large conventional memory, since they are only used for the testing of the nanowire crossbar devices in the manufacturing facility and the original creation of a useful circuit. By growing only logarithmically with the number of data wires, for circuits with many wires, such a table will only require a relatively small number of bits compared to the number of components of the nanoscale circuit. Specifically, storing the M address bits for each of the N nanowires requires $2MN$ bits (including both row and column addressing), compared to the N^2 crosspoints in the crossbar.

5 Discussion

Sharp thresholds are commonly found in statistical systems with many components, including percolation and cluster sizes in random graphs. For molecular devices, we can take advantage of thresholds to produce reliable circuits in spite of some randomness and defects in their construction. The demultiplexer is an interesting example since it also solves the problem of connecting nanoscale circuits to larger external devices to move data into or out of the molecular circuits.

Specifically, we showed how additional addressing wires can compensate for limited fabrication abilities at the nanoscale, including inability to connect devices at precise locations and variation in the connection characteristics. Due to the sharp thresholds, the number of additional wires required for good performance grows only logarithmically with the number of data wires. This remains true even when the connections

have some variation in their circuit characteristics or limited switching capability.

Crossed arrays of nanowires can be configured to provide electronic memory and logic circuits using molecular devices at the crosspoints [8, 9, 10, 11]. Thus N wires in each array give N^2 nanodevices. Addressing these wires requires two of the demultiplexer circuits discussed in this paper: one for the rows and another for the columns. Using M addressing wires, these demultiplexers use an area equivalent to $2rMN$ nanodevices, where $r \approx 10$ is the ratio in size of lithographically produced addressing wires to the size of the nanowires. The thresholds for reasonable performance described in this paper require $M = s \log_2 N$, e.g., with $s \approx 5$ for the example of Fig. 7. Thus the ratio of area for the demultiplexers to that of the nanodevices in such an array scales as $2rs(\log_2 N)/N$. With the typical values of r and s , this ratio is below one for $N \approx 1000$ or larger. Thus from an area utilization perspective, the demultiplexer circuits are reasonable only for crossbar arrays with at least 1000 wires or so. Circuits of this size are likely to be feasible [11]. For arrays with substantially more wires, the demultiplexers use only a small fraction of the overall area. This discussion also applies to the area required for the table of addresses, which also has size of order NM as described in Section 4.

Future attempts to fabricate these circuits will identify the level of component variation that must be handled by systems-level design choices, such as the threshold behavior discussed in this paper. Such experimental work is also required to evaluate the proposed fabrication approach of Section 2.1.

The demultiplexer circuit illustrates trade-offs in design requirements between the capabilities of the nanoscale devices and the resources (i.e., additional addressing wires for the multiplexer) required to ensure reliability. Our discussion uses either fully deterministic connections or completely random ones. As we saw in the discussion of thresholds, introducing constraints on the randomness, such as reducing the variation in number of connections, can improve performance. Thus an interesting extension is identifying a variety of additional constraints that give improvement. As fabrication technology improves, it may be able to impose some of these constraints on the process to thereby improve device performance or reliability with fewer additional resources.

One illustration of how improved fabrication affects design arises from the dual benefits of random connections with enough address wires to exceed the performance thresholds we identified. The main motivation for such randomness is in response to the difficulty of fabricating nanoscale connections at precisely specified locations. However, as discussed in Section 3.3, random connections also tolerate some variation in the operation of the devices. This type of defect tolerance arises from the fact that random connections tend to assign very different addresses to the data wires. However, for modest N , e.g., around 100 wires, carefully selected choices for the addresses can achieve this defect tolerance with significantly fewer extra address wires than are required for random connections [26]. Implementing such specific choices requires the ability to fabricate connections at specific locations. Thus if fabrication technology improves to the point of allowing good location specificity, but still giving high defect rates for the connections themselves, this alternate design for the addresses would be beneficial. An interesting intermediate case would arise from a fabrication technology giving some location control at the nanoscale, but with an error range only somewhat exceeding the size of individual nanowires. In this situation, instead of the purely random connection locations discussed in this paper, we could vary the connection likelihood over distances comparable to the nanowire size. This hybrid combination of randomness concentrated near specific locations could allow better performance than uniform randomness, but without requiring fully location-specific fabrication technology. Such an approach is analogous to the hybrid discussed with Fig. 3, which combines precise patterns of an ideal demultiplexer for large groups of adjacent connections with random connections for the remaining address bits.

More broadly, this work provides an application to molecular electronics of exploiting the statistical properties of self-assembly [27] and, in particular, the existence of thresholds in various performance measures. A further possibility as fabrication technology improves is being able to select, from among a large number of defective devices, combinations whose defects tend to cancel [28], thereby reducing the redundancy required to reach high-performance thresholds. Such possibilities illustrate the potential for systems-level considerations and computational processing to compensate for limitations of physical devices for both fabrication and operation [29]. More detailed evaluation of these possibilities will require simulation studies with realistic device models [30] and defect processes, as well as eventual fabrication to enable experiments with actual circuits [9, 13].

Acknowledgment

The authors thank the two anonymous reviewers for their suggestions.

References

- [1] C. P. Collier et al. Electronically configurable molecular-based logic gates. *Science*, 285:391–394, 1999.
- [2] Yu Huang et al. Logic gates and computation from assembled nanowire building blocks. *Science*, 294:1313–1317, 2001.
- [3] Adrian Bachtold et al. Logic circuits with carbon nanotube transistors. *Science*, 294:1317–1320, 2001.
- [4] Daniel B. Klein, editor. *Reputation: Studies in the Voluntary Elicitation of Good Conduct*. Univ. of Michigan Press, Ann Arbor, 1997.
- [5] George M. Whitesides and Bartosz Grzybowski. Self-assembly at all scales. *Science*, 295:2418–2421, 2002.
- [6] Nicholas A. Melosh et al. Ultrahigh-density nanowire lattices and circuits. *Science*, 300:112–115, 2003.
- [7] Yong Chen and R. Stanley Williams. Nanoscale patterning for the formation of extensive wires. US Patent 6,294,450, September 25 2001.
- [8] James R. Heath, Philip J. Kuekes, Gregory S. Snider, and R. Stanley Williams. A defect-tolerant computer architecture: Opportunities for nanotechnology. *Science*, 280:1716–1721, 1998.
- [9] Yong Chen et al. Nanoscale molecular-switch crossbar circuits. *Nanotechnology*, 14:462–468, 2003.
- [10] Greg Snider, Philip J. Kuekes, and R. Stanley Williams. CMOS-like logic in defective, nanoscale crossbars. *Nanotechnology*, 15:881–891, 2004.
- [11] Andre DeHon, Seth Copen Goldstein, Philip J. Kuekes, and Patrick Lincoln. Nonphotolithographic nanoscale memory density prospects. *IEEE Trans. on Nanotechnology*, 4:215–228, 2005.
- [12] Andre DeHon, Patrick Lincoln, and John E. Savage. Stochastic assembly of sublithographic nanoscale interfaces. *IEEE Trans. on Nanotechnology*, 2:165–174, 2003.
- [13] Zhaohui Zhong et al. Nanowire crossbar arrays as address decoders for integrated nanosystems. *Science*, 302:1377–1379, 2003.
- [14] David L. Goodstein. *States of Matter*. Prentice-Hall, Englewood Cliffs, NJ, 1975.
- [15] D. Stauffer and A. Aharony. *Introduction to Percolation Theory*. Taylor and Francis, London, 2nd edition, 1992.
- [16] P. Erdos and A. Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 5:17–61, 1960.
- [17] David J. D. Earn et al. A simple model for complex dynamical transitions in epidemics. *Science*, 287:667–670, 2000.
- [18] Tad Hogg, Bernardo A. Huberman, and Colin P. Williams, editors. *Frontiers in Problem Solving: Phase Transitions and Complexity*, volume 81, Amsterdam, 1996. Elsevier. Special issue of *Artificial Intelligence*.
- [19] Bhaskar Krishnamachari, Stephen B. Wicker, and Ramon Bejar. Phase transition phenomena in wireless ad-hoc networks. In *Symposium on Ad-Hoc Wireless Networks, GlobeCom2001*, 2001.

- [20] W. S. Stornetta, B. A. Huberman, and T. Hogg. Scaling theory for fault stealing algorithms in large systolic arrays. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9(3):290–298, 1990.
- [21] J. von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, volume 34 of *Ann. of Math. Stud.*, pages 43–98. Princeton University Press, 1956.
- [22] Richard P. Feynman. *Feynman Lectures on Computation*. Addison-Wesley, Reading, MA, 1996.
- [23] P. J. Kuekes and R. S. Williams. Demultiplexer for a molecular wire crossbar network. US Patent 6,256,767, July 2001.
- [24] A. Nijenhuis and H. S. Wilf. *Combinatorial Algorithms for Computers and Calculators*. Academic Press, New York, 2nd edition, 1978.
- [25] Jin Zhang et al., editors. *Self-Assembled Nanostructures*, volume 1 of *Nanostructure Science and Technology*. Kluwer Academic, New York, 2002.
- [26] Philip J. Kuekes, Warren Robinett, Gadiel Seroussi, and R. Stanley Williams. Defect-tolerant demultiplexers for nanoelectronics constructed from error-correcting codes. *Applied Physics A*, 80:1161–1164, 2005.
- [27] Tad Hogg. Robust self-assembly using highly designable structures. *Nanotechnology*, 10:300–307, 1999. Available at <http://www.foresight.org/Conferences/MNT6/Papers/Hogg/index.html>.
- [28] Damien Challet and Neil F. Johnson. Optimal combinations of imperfect objects. arxiv.org preprint cond-mat/0203028, Oxford, 2002.
- [29] Tad Hogg and Bernardo A. Huberman. Controlling smart matter. *Smart Materials and Structures*, 7:R1–R14, 1998. arxiv.org preprint cond-mat/9611024.
- [30] M. M. Ziegler et al. Scalability simulations for nanomemory systems integrated on the molecular scale. *Ann. NY Acad. Sci.*, 1006:312–330, 2003.