LETTER

doi:10.1038/nature10262

Neural network computation with DNA strand displacement cascades

Lulu Qian[1], Erik Winfree[1,2,3] & Jehoshua Bruck[3,4]

The impressive capabilities of the mammalian brain—ranging from perception, pattern recognition and memory formation to decision making and motor activity control—have inspired their re-creation in a wide range of artificial intelligence systems for applications such as face recognition, anomaly detection, medical diagnosis and robotic vehicle control[1]. Yet before neuron-based brains evolved, complex biomolecular circuits provided individual cells with the 'intelligent' behaviour required for survival[2]. However, the study of how molecules can 'think' has not produced an equal variety of computational models and applications of artificial chemical systems. Although biomolecular systems have been hypothesized to carry out neural-network-like computations *in vivo*[3,2,4] and the synthesis of artificial chemical analogues has been proposed theoretically[5–9], experimental work[10–13] has so far fallen short of fully implementing even a single neuron. Here, building on the richness of DNA computing[14] and strand displacement circuitry[15], we show how molecular systems can exhibit autonomous brain-like behaviours. Using a simple DNA gate architecture[16] that allows experimental scale-up of multilayer digital circuits[17], we systematically transform arbitrary linear threshold circuits[18] (an artificial neural network model) into DNA strand displacement cascades that function as small neural networks. Our approach even allows us to implement a Hopfield associative memory[19] with four fully connected artificial neurons that, after training *in silico*, remembers four single-stranded DNA patterns and recalls the most similar one when presented with an incomplete pattern. Our results suggest that DNA strand displacement cascades could be used to endow autonomous chemical systems with the capability of recognizing patterns of molecular events, making decisions and responding to the environment.

The human brain is composed of $\sim 10^{11}$ neurons, and each has a few thousand synapses. Each synapse can receive signals from other neurons, raising or lowering the electrical potential inside the neuron. When the potential reaches its threshold, the neuron will fire and a pulse will be sent through the axon to other neurons. Among the simplest mathematical models of neurons is the perceptron, also known as the linear threshold gate[1,18,20]. A linear threshold gate has a number of inputs, $x_1, x_2, \cdots, x_n \in \{0,1\}$, which can be interpreted as arriving at synapses that each have an analogue weight, $w_1, w_2, \cdots, w_n$. The linear threshold gate turns 'on' only when the weighted sum of all inputs exceeds a threshold, th. The output

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i \geq \text{th} \\ 0 & \text{otherwise} \end{cases}$$

can be interpreted as the firing activity on the axon. Linear threshold gates may be used to construct multilayer circuits that are complete for Boolean functions and, more importantly, are exponentially more compact than AND–OR–NOT circuits for a wide class of functions[18,21,22]. Recurrent linear threshold circuits have even provided insights into brain-like computations, such as content-addressable associative memories[19]. A remarkable feature of brains, which is also desirable for molecular circuits, is that complex computations can be carried out by networks with just a few layers and even with unreliable components—a feature that linear threshold circuits share[23].

We first introduce the simple DNA gate architecture, based on what we call the 'seesaw' gate motif, which we use for building arbitrary linear threshold circuits. Because DNA hybridization depends primarily on the logic of Watson–Crick base-pairing, many instances of the same molecular motif can be created by assigning different sequence choices for each logical domain. The abstract diagram for the seesaw gate motif (Fig. 1a) provides a concise representation of a full DNA implementation and can be systematically translated first to the domain level, then to the sequence level, and finally to the molecular level (Supplementary Fig. 1). Each seesaw gate is a node with two sides, connected to one or more wires on each side. Each wire represents a DNA 'signal strand' with two long 'recognition' domains flanking a central short 'toehold' domain (for example, 'input' and 'fuel' in Fig. 1a). Each node represents a DNA gate 'base strand' with one central recognition domain flanked by two toehold domains. The gate base strand is always bound to a signal strand on one side or another, leaving one toehold uncovered (for example, the 'gate:output complex' in Fig. 1a). A DNA 'threshold complex' can also be associated with a node; it has a double-stranded recognition domain with an extended toehold. To read the output signal, a 'reporter' gate is used (Fig. 1b). The reporter is implemented as a threshold-like DNA complex with a fluorophore/quencher pair at the end of the duplex.

There are three basic reactions involved in a seesaw network (Supplementary Fig. 4). They all use the principle of toehold-mediated DNA strand displacement[15], in which a single-stranded DNA binds to a partially double-stranded complex by a single-stranded toehold domain, allowing initiation of branch migration through a recognition domain with identical sequence, and ultimately resulting in replacement and release of the originally bound strand. The first reaction, seesawing, occurs when a free signal on one side of a gate releases a signal bound on the other side. A single step of seesawing results in stoichiometric exchange of equal amounts of activity from a wire on one side of a gate to a wire on the other side (for example, input releases output). Two steps of seesawing completes a catalytic cycle in which a wire on one side of a gate exchanges the activity between two wires on the other side without itself being consumed (for example, input transforms free fuel into free output, see Supplementary Fig. 5). Second, thresholding occurs when a threshold complex absorbs an impinging signal—this happens at a much faster rate than seesawing because of the extended toehold[15]. Third, reporting occurs when a reporter complex absorbs an impinging signal while generating a fluorescence signal.

A single linear threshold gate can be implemented using three types of seesaw gates that implement the three essential subfunctions: multiplying ($w_i x_i$), integrating ($\sum w_i x_i$) and thresholding ($\sum w_i x_i \geq$ th). (1) Multiplying gates (for example, the first layer of seesaw gates in Fig. 1c, e) have a fixed threshold of 0.2 and support multiple outputs with arbitrary weights. Each output strand has a different recognition domain on the right (5′ end) to connect to different downstream gates.

[1]Bioengineering, California Institute of Technology, Pasadena, California 91125, USA. [2]Computer Science, California Institute of Technology, Pasadena, California 91125, USA. [3]Computation and Neural Systems, California Institute of Technology, Pasadena, California 91125, USA. [4]Electrical Engineering, California Institute of Technology, Pasadena, California 91125, USA.

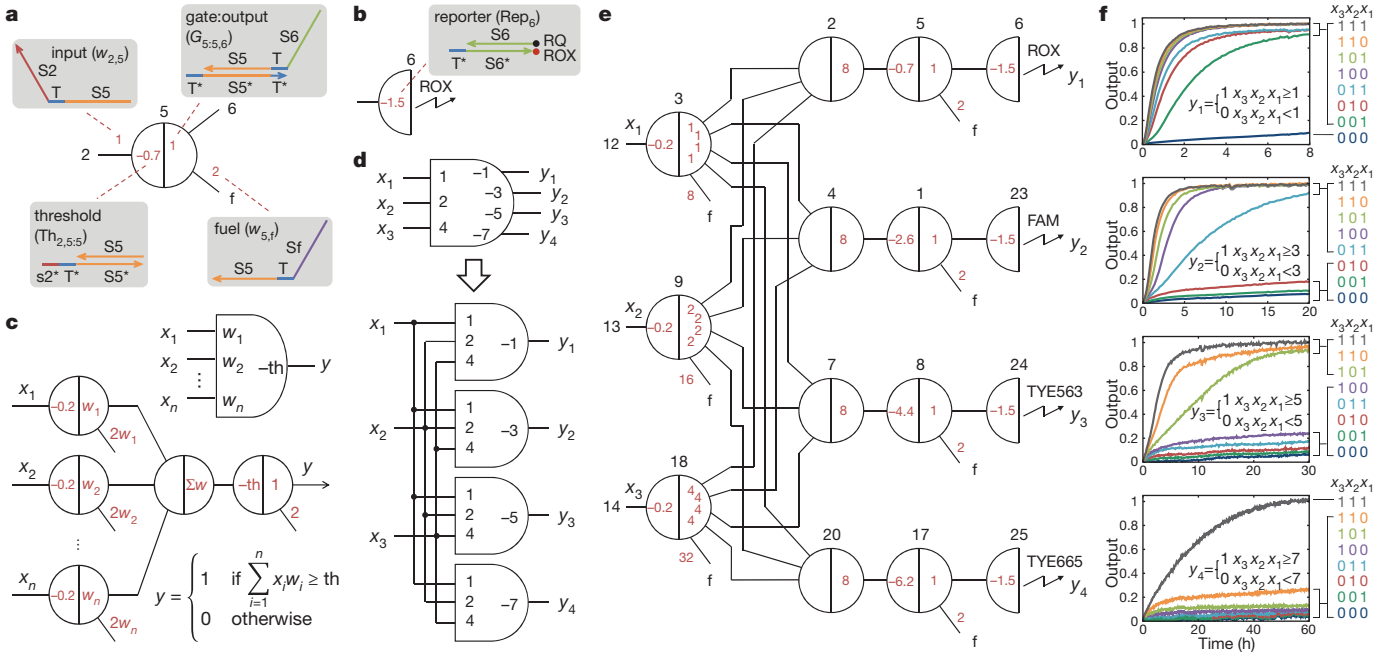

Figure 1 | The seesaw gate motif and the construction of linear threshold gates. a, Abstract diagram of a seesaw gate motif and its DNA implementation. Black numbers indicate the identity of each node (or the interface to that node in a larger network). Positions and signs of red numbers indicate different DNA species, while their absolute values indicate the initial relative concentrations (for details, see Supplementary Information section 1). Each species has a specific role (for example, input) within a gate and has a unique name (for example, $w_{2,5}$) within a network. Coloured lines represent DNA strands at the domain level, with arrowheads marking their 3′ ends and colours indicating distinct DNA subsequences. S2, S5, S6 and Sf are long recognition domains. T is a short toehold domain. T* is the Watson–Crick complement of T, and so on. s2* is the first few nucleotides of S2* from the 3′ end. **b**, Abstract diagram of a reporter and its DNA implementation. Fluorophore ROX is quenched by quencher RQ. **c**, A linear threshold gate and its equivalent seesaw construction. **d**, A general three-input four-output linear threshold gate. **e**, Equivalent seesaw circuit for the general linear threshold gate. Note that thresholds and weights in the final construction are adjusted to obtain improved experimental performance. Fluorophores ROX, FAM, TYE563 and TYE665 are used for four reporters to monitor four outputs y_1 to y_4. **f**, Kinetics experiments of the general linear threshold gate. A total of 60 DNA strands assembled to form 38 initial DNA species (as indicated by the red numbers in **e**) were mixed in solution at their respective concentrations. The standard concentration was $1\times = 16.67$ nM. Input strands x_1 to x_3 were then added with relative concentrations of $0.1\times$ (0, logic 'off') or $0.9\times$ (1, logic 'on'). Output signals y_1 to y_4 were reported by four distinct fluorophores simultaneously (Supplementary Fig. 2). Trajectories for corresponding inputs are shown with matching colours. Domains and strand sequences are listed in Supplementary Tables 1–4, circuit 2. Experiments were performed at 20 °C in Tris-acetate-EDTA buffer containing 12.5 mM Mg^{2+}. Output signals were inferred by fluorescence signals normalized to the maximum completion level (Supplementary Fig. 3).

To clean up variations due to leaky reactions and signal decay, we require all gates to work with a digital abstraction where 'off' signals may be between 0 to 0.2, and 'on' signals may be between 0.8 to 1. If the input is 'off', all outputs will remain 0. If the input is 'on', it will exceed the threshold and catalyse the exchange of fuel and outputs. With an irreversible downstream drain, each output will continue being released until no gate:output complexes remain. Thus, each output level is set by an analogue weight—the initial amount of gate:output complex. (2) Integrating gates (for example, the second layer of seesaw gates in Fig. 1c, e) have no threshold or fuel, but support multiple inputs. All input strands have the same right recognition domain to connect to this gate, but have different left recognition domains corresponding to different upstream gates. Without fuel, the gate exhibits a stoichiometric behaviour with the output level eventually reaching the sum of all inputs. (3) Thresholding gates (for example, the third layer of seesaw gates in Fig. 1c, e) have an arbitrary threshold and an output with a fixed weight of 1. If the input exceeds the threshold, the output will turn 'on'; otherwise it will stay 'off'. To reduce circuit size in cascades, multiplying gates and thresholding gates can be combined and generalized as a fourth type, amplifying gates, that allow both an arbitrary threshold and multiple outputs with arbitrary weights (Supplementary Fig. 6). The full DNA strand displacement cascade implementing a single neuron is shown in Supplementary Fig. 7.

To translate arbitrary linear threshold circuits into seesaw circuits, we develop four transformation rules: complementation, expansion, consolidation and reduction (see Supplementary Fig. 8 for details). Complementation is used to convert a linear threshold circuit with negative weights into an equivalent circuit with positive weights only (for example, Fig. 2a). Expansion is used to transform each n-input linear threshold gate with positive weights into an equivalent network of $n + 2$ seesaw gates (for example, Fig. 1c). Consolidation is used to collect multiple occurrences of the same signal into one when connecting subcircuits together (for example, yielding nodes 3, 9 and 18 in Fig. 1e). Reduction is used to combine an upstream threshold and a directly downstream weight into a single operation after composition (for example, yielding nodes 17 and 8 in Fig. 2b).

For our initial experimental demonstration, we chose a general three-input four-output linear threshold gate (Fig. 1d). It is equivalent to a linear threshold circuit with four parallel gates that each read the same three inputs. The circuit calculates the analogue value of a three-bit binary number, then compares it to 1, 3, 5 and 7. An equivalent seesaw circuit (Fig. 1e) is generated using just two of the above transformation rules: expansion and consolidation. The first layer of seesaw gates fans out each input while multiplying by the corresponding weight; the second layer calculates the sum of all weighted inputs; the third layer implements the corresponding threshold for each output; the final layer of reporters reads the output signals and provides irreversible drains.

In fluorescence kinetics experiments (Fig. 1f), all four outputs (y_1–y_4) achieved the correct 'on' or 'off' states with the complete eight sets of inputs (x_3, x_2, x_1, on right side of graphs, colour coded to match traces), even though the inputs were intentionally 'noisy' ($0.1\times$ standard concentration was used for 'off' inputs and $0.9\times$ for 'on' inputs). In a subcircuit roughly half the size, we tuned weights and thresholds to show that the same set of DNA molecules can implement different linear threshold functions (Supplementary Figs 9 and 10). Although

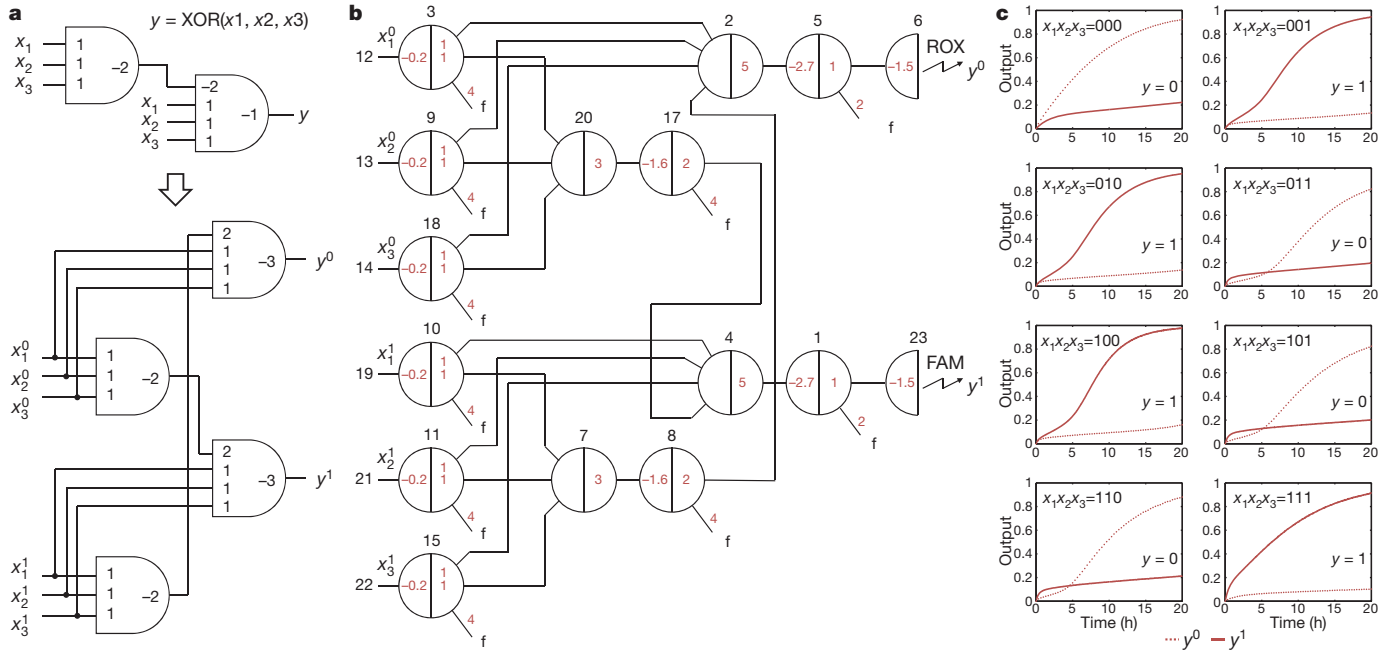

Figure 2 | A linear threshold circuit that computes the three-bit XOR function. This function is given by XOR(x_1, x_2, x_3) = (x_1 + x_2 + x_3) mod 2. **a**, A three-bit XOR circuit and its equivalent dual-rail circuit. **b**, An equivalent seesaw circuit. **c**, Kinetics experiments. A total of 68 DNA strands assembled to form 42 initial DNA species (as indicated by the red numbers in **b**) were mixed in solution at their respective concentrations. The standard concentration was 1× = 50 nM. Six dual-rail input strands were then added with relative concentrations of 0.9× x_i^0 and 0.1× x_i^1 (for $x_i = 0$, logic 'off'), or 0.1× x_i^0 and 0.9× x_i^1 (for $x_i = 1$, logic 'on'). Dotted and solid lines indicate dual-rail output being logic 'off' and logic 'on', respectively. Domains and strand sequences are listed in Supplementary Tables 1–4, circuit 3. Experiments were performed at 25 °C.

thresholding, catalysis and integration have been demonstrated previously in seesaw digital logic circuits[17], this is to our knowledge the first demonstration of variable weights, variable thresholds and linear integration composed together, performing the function of an artificial neuron.

To show computation with negative weights and cascading, a linear threshold circuit that computes the three-bit exclusive-or (XOR) function was demonstrated (Fig. 2a). With an efficient construction, the XOR function with n variables can be realized with $\log_2 n$ linear threshold gates[21], whereas the optimal size of an AND–OR–NOT circuit[22] is at least $2n$. All four transformation rules are used to generate an equivalent seesaw circuit (Fig. 2b). Complementation introduces dual-rail logic[24], where each input x_i is replaced by a pair of inputs x_i^0 and x_i^1, representing logic 'off' and logic 'on' separately; each linear threshold gate is replaced by a pair of gates with only positive weights, producing a pair of dual-rail outputs. Thus, a computed 'off' value can be distinguished from an output that has not yet been computed. This method avoids the difficulty of directly implementing negative weights, at a cost of doubling the size of the circuit. The top half of the seesaw circuit corresponds to the cascade of two linear threshold gates that read inputs x_i^0 and output y^0; the bottom half corresponds to the other two, which read inputs x_i^1 and output y^1. The cross-connection between the two halves appears where there is a negative weight in the original circuit.

In fluorescence kinetics experiments (Fig. 2c), the pair of dual-rail outputs went to their correct 'on'/'off' states, again even with noisy inputs. When the inputs had an even number of 1s, the output y^0 went 'on' and y^1 went 'off', indicating $y = 0$; when the inputs had an odd number of 1s, the output y^0 went 'off' and y^1 went 'on', indicating $y = 1$. With inputs $x_1x_2x_3$ = 000 and 111, the output responded sooner than all the other cases, where the production of output must wait for the upstream linear threshold gate to provide its input. Experimental insights gained from the networks of Figs 1 and 2, comparison to a simpler implementation of a three-bit XOR using deoxyribozymes[25], as well as comparison to other neural network implementations, are discussed in Supplementary Information section 4.

To show a recurrent linear threshold circuit and the power of neural network computation, a four-neuron Hopfield associative memory was demonstrated. A Hopfield network[19] has a number of artificial neurons that are fully connected to each other. If properly trained, which means the weights and threshold of each neuron are properly chosen, the network is able to 'remember' a set of patterns; when initialized with a partial or distorted pattern, the network will recover the most similar remembered pattern. We used the perceptron learning algorithm[1] *in silico* (see Supplementary Information section 5) to train our four-neuron Hopfield network to remember four patterns: 0110, 1111, 0011 and 1000 (Fig. 3a).

To implement negative weights, we again use the dual-rail convention where each signal x_i is replaced by a pair of signals x_i^0 and x_i^1 (Supplementary Fig. 13). To run the network, there are three possible initial states for each neuron: 0 (logic 'off') if $x_i^0 = 1$, 1 (logic 'on') if $x_i^1 = 1$, or unknown if both x_i^0 and $x_i^1 = 0$. For each update of a linear threshold gate, some x_i^0 or x_i^1 can flip from 0 to 1, but not back; the corresponding neuron can change its state from unknown to 0 or to 1, but not back. As the network runs, another possibility arises: a neuron's state is declared invalid if both x_i^0 and $x_i^1 = 1$. Like the original network, this dual-rail Hopfield associative memory can associate an incomplete pattern with a remembered pattern; unlike the original, it is unable to recover a corrupted pattern because the neurons' states cannot flip from 0 to 1 or from 1 to 0; but thanks to the additional states (unknown and invalid), it has the new feature of identifying patterns that are compatible with no single remembered pattern (see Supplementary Information section 5 and the bottom two panels of Fig. 3e).

Following the same transformation rules described above, a seesaw circuit equivalent to the dual-rail Hopfield network is shown in Fig. 3b, containing 24 feedback loops. Initial experiments confirmed that in a circuit with feedback connections between catalytic seesaw gates, leak reactions that occur in DNA strand displacement circuitry are amplified after they exceed the threshold (Supplementary Fig. 14). Therefore, in front of each reporter we add an extra signal restoration step consisting

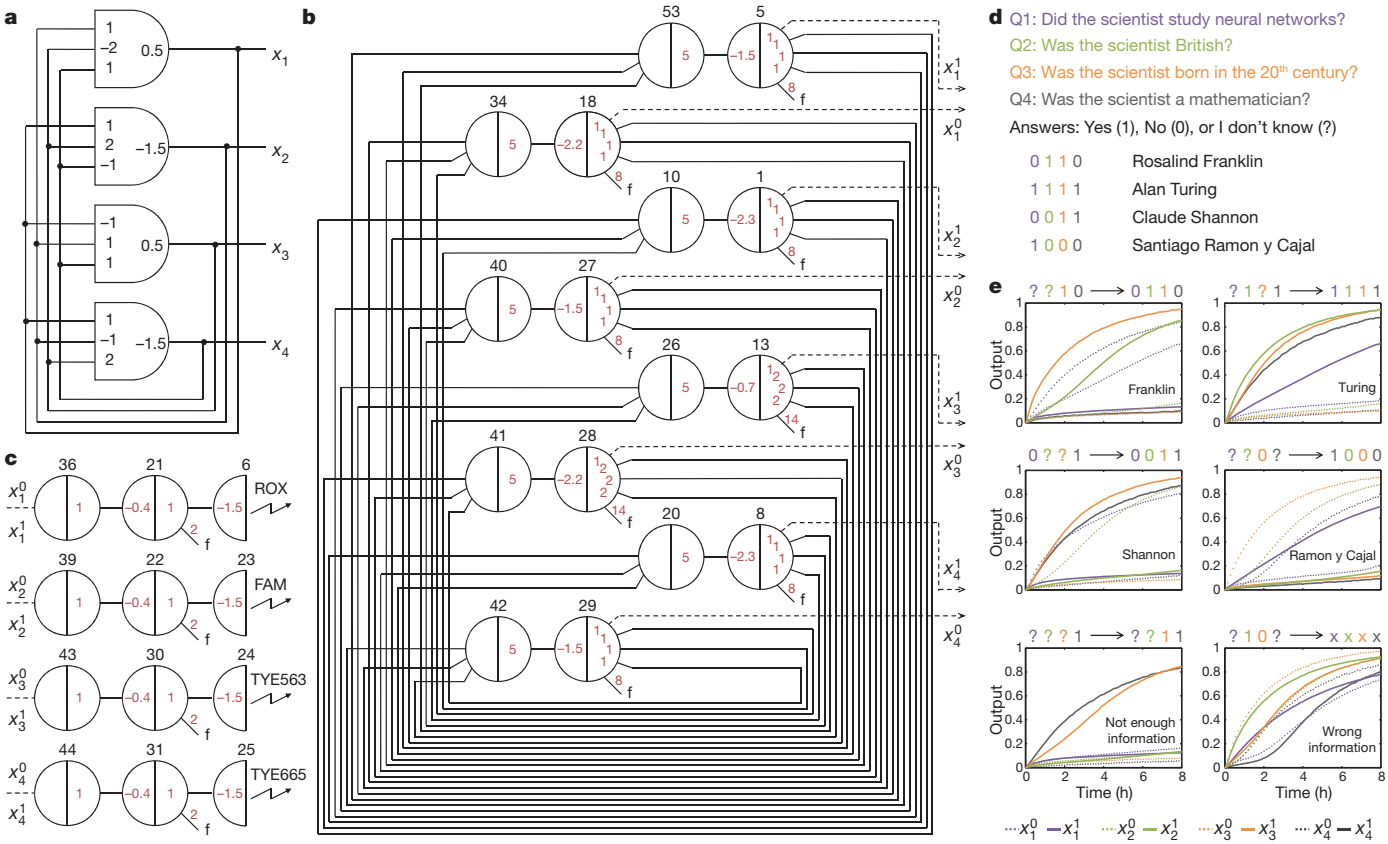

Figure 3 | A four-neuron Hopfield associative memory. **a**, The recurrent linear threshold circuit. **b**, The resulting seesaw circuit using the dual-rail implementation. Dashed lines indicate the connections to reporters. **c**, Four sets of reporters with signal restoration that are connected to either x_i^0 or x_i^1 at any given time. **d**, A 'read your mind' game between a human and the four-neuron DNA associative memory that 'remembers' four scientists according to the answers of four questions. **e**, Kinetics experiments of the 'read your mind' game. A total of 112 DNA strands assembled to form 72 initial DNA species (as indicated by the red numbers in **b**, **c**) were mixed in solution at their respective concentrations. The standard concentration was 1× = 25 nM. Selected inputs corresponding to the human's answers were then added with relative concentrations of 5× (to set the initial states, inputs triggering the update of multiple neurons are used, for example, $w_{53,5}$ for x_1^1 and $w_{34,18}$ for x_1^0). Dotted and solid lines indicate dual-rail outputs x_i^0 and x_i^1, respectively. For each signal, if both dotted and solid lines stay 'off' (less than 0.2), the logic value is unknown, '?'; if the dotted (solid) line goes 'on' (greater than 0.65) and the solid (dotted) line stays 'off', the logic value is '0' ('1'); if both dotted and solid lines go 'on', the logic value is invalid, 'x'. Arrows connect initial states of the four neurons (inputs) to the final states (outputs at 8 h). The eight trajectories in each plot were from two separate experiments (connecting either x_i^0 or x_i^1 to the reporters) because we only have four distinct fluorophores. Sequences of strands are listed in Supplementary Tables 5–7. Experiments were performed at 25 °C.

of an integrating gate and an amplifying gate, in order to suppress leak (Fig. 3c). The values of weights and thresholds determined by *in silico* training were used to determine the concentrations of the 72 DNA species that comprise the memory (Fig. 3b, c). In principle, the same set of DNA molecules could be retrained to remember any of 500 distinct sets of patterns by adjusting weight and threshold concentrations (Supplementary Information section 5).

In the tradition of using game-playing automata as a benchmark for new computing technologies, we demonstrated the Hopfield network in the context of a game called 'read your mind', which is played between a human and the DNA associative memory in a cuvette (Fig. 3d). The game consists of three steps. First, the human thinks of a scientist, choosing from the listed four options (each scientist corresponds to one of the four patterns; for example, Franklin is 0110) or someone else. Second, the human 'tells' the DNA associative memory some of the answers to questions Q1 to Q4 (Fig. 3d) by adding corresponding DNA strands to the cuvette. Finally, after 8 h of 'thinking', the DNA associative memory will guess who is in the human's mind and 'tell' the human the rest of the answers by fluorescence signals. In doing so, the four-neuron DNA associative memory exhibits a brain-like behaviour: associative recall of memories based on incomplete information.

We played the game 27 times with the DNA associative memory, out of 81 possible ways of answering questions Q1 to Q4. Six examples are shown in Fig. 3e; the rest are shown in Supplementary Figs 15–18. The top left data in Fig. 3e can be interpreted as following: when the human 'said' that the scientist was born in the twentieth century (input $x_3 = 1$) but was not a mathematician (input $x_4 = 0$), the DNA associative memory 'guessed' that the scientist did not study neural networks (output x_1 was updated to 0) but was British (output x_2 was updated to 1), which indicated that the scientist was Rosalind Franklin (pattern 0110). Similarly, the DNA associative memory was able to work out the other three scientists correctly—in the best case, only one answer was given by the human (the middle right data). The bottom left data shows that when the information provided by the human matched multiple patterns (that is, input $x_4 = 1$ indicates that the scientist was a mathematician, which is true for both Alan Turing and Claude Shannon), the DNA associative memory was able to identify that they were both born in the twentieth century (output x_3 was updated to 1), while the other outputs remained unknown. The bottom right data show that the DNA associative memory was also able to recognize information that was incompatible with all memorized patterns by producing invalid output.

All experiments reported here were semiquantitatively reproduced by mass action simulations using the exact model developed previously for seesaw digital logic circuits[17] with no changes to any rate constants (see Supplementary Information section 7 and Supplementary Figs 19–24 for comparisons to experiments, and Supplementary Figs 25–27 for simulation predictions for the remaining 54 games).

It is interesting to consider the scale of our reactions. Stochastic simulations suggest that the four-neuron DNA associative memory would function reliably with even just 10 copies of each species at 1× concentrations (Supplementary Fig. 28), which at our concentrations would entail a volume of roughly 1 μm^3, that is, small enough to fit inside a bacterium. In the other direction, scaling up the DNA associative memory to contain more neurons will exacerbate problems with spurious reactions, and may require lower concentrations and thus slower reactions. On the other hand, neural associative memories are intrinsically fault-tolerant[19] and can function well even with only sparse connections[26]. Because of these opposing factors, it is difficult to predict how large a network can be successfully implemented using the approach described here.

To create smart and functional chemical systems, our current constructions will need to be improved and integrated with other chemistries. For sustained autonomous behaviour, it will be important to go beyond use-once architectures to dynamic units that can turn 'on' and 'off' repeatedly as their inputs change. Initial examples of such systems have been demonstrated using enzymes[13] and are theoretically possible in DNA-only systems[27]. Of particular interest would be to implement the dynamics for learning rules within the chemistry itself[28], as hinted at by recent demonstrations of trainable chemical circuits[29]. Nonetheless, even simple linear threshold units could be quite useful in biomedical diagnostics, such as classifying cancers with microRNA signals[9,30]. Furthermore, when DNA strand displacement systems are provided with interfaces for sensing non-nucleic-acid inputs and controlling chemical reactions as output actions[31], an 'intelligent' DNA system could directly perceive and act on its chemical environment.

Received 31 December 2010; accepted 31 May 2011.

1. Rojas, R. *Neural Networks: A Systematic Introduction* (Springer, 1996).
2. Bray, D. Protein molecules as computational elements in living cells. *Nature* **376,** 307–312 (1995).
3. Mjolsness, E., Sharp, D. H. & Reinitz, J. A connectionist model of development. *J. Theor. Biol.* **152,** 429–453 (1991).
4. Buchler, N. E., Gerland, U. & Hwa, T. On schemes of combinatorial transcription logic. *Proc. Natl Acad. Sci. USA* **100,** 5136–5141 (2003).
5. Hjelmfelt, A., Weinberger, E. D. & Ross, J. Chemical implementation of neural networks and Turing machines. *Proc. Natl Acad. Sci. USA* **88,** 10983–10987 (1991).
6. Baum, E. B. Building an associative memory vastly larger than the brain. *Science* **268,** 583–585 (1995).
7. Mills, A. P., Yurke, B. & Platzman, P. M. Article for analog vector algebra computation. *Biosystems* **52,** 175–180 (1999).
8. Kim, J., Hopfield, J. J. & Winfree, E. in *Advances in Neural Information Processing Systems* Vol. 17 (eds Saul, L. K., Weiss, Y. & Bottou, L.) 681–688 (MIT Press, 2004).
9. Zhang, D. Y. & Seelig, G. in *DNA Computing and Molecular Programming* (eds Sakakibara, Y. & Mi, Y.) 176–186 (Lecture Notes in Computer Science, Vol. 6518, Springer, 2011).
10. Laplante, J. P., Pemberton, M., Hjelmfelt, A. & Ross, J. Experiments on pattern recognition by chemical kinetics. *J. Phys. Chem.* **99,** 10063–10065 (1995).
11. Mills, A. P. Jr, Turberfield, M., Turberfield, A. J., Yurke, B. & Platzman, P. M. Experimental aspects of DNA neural network computation. *Soft Comput.* **5,** 10–18 (2001).
12. Lim, H. W. *et al. In vitro* molecular pattern classification via DNA-based weighted-sum operation. *Biosystems* **100,** 1–7 (2010).
13. Kim, J. & Winfree, E. Synthetic *in vitro* transcriptional oscillators. *Mol. Syst. Biol.* **7,** 465 (2011).
14. Adleman, L. M. Molecular computation of solutions to combinatorial problems. *Science* **266,** 1021–1024 (1994).
15. Zhang, D. Y. & Seelig, G. Dynamic DNA nanotechnology using strand-displacement reactions. *Nature Chem.* **3,** 103–113 (2011).
16. Qian, L. & Winfree, E. A simple DNA gate motif for synthesizing large-scale circuits. *J. R. Soc. Interface* doi:10.1098/rsif.2010.0729 (published online 4 February 2011).
17. Qian, L. & Winfree, E. Scaling up digital circuit computation with DNA strand displacement cascades. *Science* **332,** 1196–1201 (2011).
18. Muroga, S. *Threshold Logic and its Applications* Vol. 18 (Wiley-Interscience, 1971).
19. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl Acad. Sci. USA* **79,** 2554–2558 (1982).
20. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biol.* **5,** 115–133 (1943).
21. Kautz, W. H. The realization of symmetric switching functions with linear-input logical elements. *IRE Trans. Electron. Comput.* **EC-10,** 371–378 (1961).
22. Wegener, I. The complexity of the parity function in unbounded fan-in, unbounded depth circuits. *Theor. Comput. Sci.* **85,** 155–170 (1991).
23. Hajnal, A., Maass, W., Pudlák, P., Szegedy, M. & Turan, G. Threshold circuits of bounded depth. *J. Comput. Syst. Sci.* **46,** 129–154 (1993).
24. Müller, D. E. in *Symp. on the Application of Switching Theory* (eds Aiken, H. & Main, W. F.) 289–297 (Stanford Univ. Press, 1963).
25. Lederman, H., Macdonald, J., Stefanovic, D. & Stojanovic, M. N. Deoxyribozyme-based three-input logic gates and construction of a molecular full adder. *Biochemistry* **45,** 1194–1199 (2006).
26. Amari, S. I. Characteristics of sparsely encoded associative memory. *Neural Netw.* **2,** 451–457 (1989).
27. Soloveichik, D., Seelig, G. & Winfree, E. DNA as a universal substrate for chemical kinetics. *Proc. Natl Acad. Sci. USA* **107,** 5393–5398 (2010).
28. Fernando, C. T. *et al.* Molecular circuits for associative learning in single-celled organisms. *J. R. Soc. Interface* **6,** 463–469 (2009).
29. Pei, R., Matamoros, E., Liu, M., Stefanovic, D. & Stojanovic, M. N. Training a molecular automaton to play a game. *Nature Nanotechnol.* **5,** 773–777 (2010).
30. Rosenfeld, N. *et al.* MicroRNAs accurately identify cancer tissue origin. *Nature Biotechnol.* **26,** 462–469 (2008).
31. Simmel, F. C. Towards biomedical applications for nucleic acid nanodevices. *Nanomedicine* **2,** 817–830 (2007).

Supplementary Information is linked to the online version of the paper at www.nature.com/nature.

Acknowledgements We thank P. Rothemund, P. Yin, D. Woods, D. Soloveichik and N. Dabby for comments on the manuscript. We also thank R. Murray for the use of experimental facilities. This work was supported by the NSF (grant nos 0728703 and 0832824 (The Molecular Programming Project)) and by HFSP award no. RGY0074/2006-C.

Author Contributions L.Q. designed the system, performed the experiments and analysed the data; L.Q. and E.W. performed the *in silico* training and wrote the manuscript; E.W. guided the project and discussed the design and the data; and J.B. initiated and guided the project, and discussed the manuscript.

Author Information Reprints and permissions information is available at www.nature.com/reprints. The authors declare no competing financial interests. Readers are welcome to comment on the online version of this article at www.nature.com/nature. Correspondence and requests for materials should be addressed to E.W. (winfree@caltech.edu).