nature nanotechnology

SUPPLEMENTARY INFORMATION

DOI: 10.1038/NNANO.2013.189

Programmable chemical controllers made from DNA

Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig

Contents

S1 Strand displacement

S1.1 Mechanism of strand displacement

Strand displacement provides the mechanism by which all strands and gates in this paper interact. An example reaction involving signal strand *A* and a partial gate complex is shown in Fig. S1. Strand displacement is initiated when complementary toeholds on the signal and gate (*ta* and *ta** in Fig. S1b) bind to each other; the reaction then proceeds through a three-way branch migration step and results in the release of the auxiliary strand <*a tb*> when domains *tb* and *tb** dissociate. Short toehold domains (*ta*, *tb*, ...) are designed to bind to their complement (*ta**, *tb**, ...) reversibly at room temperature. In contrast, long domains (*a*, *a**,...) are designed to bind their complements strongly enough that their dissociation can proceed only through branch migration. If the toehold *tb* for the reverse reaction is completely missing, strand displacement becomes essentially irreversible. Because strand displacement is markedly inhibited by sequence differences, simple sequence design ensures that a long domain can be displaced only by its exact complement.

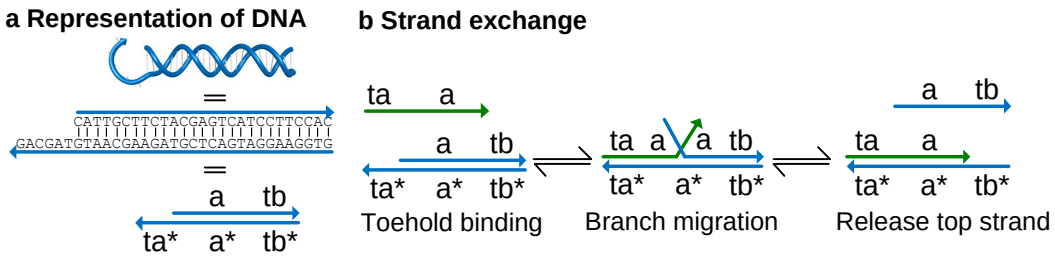


Figure S1: Mechanism of strand displacement. a, Double stranded gate motif at the helix level, sequence level, and domain level. A long domain (labeled as *a*, *b*, etc.) provides signal identity; a short toehold domain (labeled as *ta*, *tb*, etc.) initiates strand displacement. Arrowheads indicate 3' ends and * indicates complementarity. Contiguous nucleotides that act as a unit in strand displacement and hybridization ("domains") are labeled with lower case letters. **b**, Strand displacement mechanism. Details are explained in the text.

S1.2 Reporter strategy

Double stranded reporter complexes are used to follow the time course of output signals. As shown in Fig. S2a, the output signal *C* interacts with the reporter complex to separate the quencher-labeled strand from the fluorophore-labeled strand, which results in the increase of the fluorescence signal (Fig. S2b). The calibration curve of the reporter is made by a linear fit of the final fluorescence values against the concentration of signal *C* (Fig. S2c). This calibration curve can be used to convert arbitrary fluorescence units to the corresponding signal concentration.

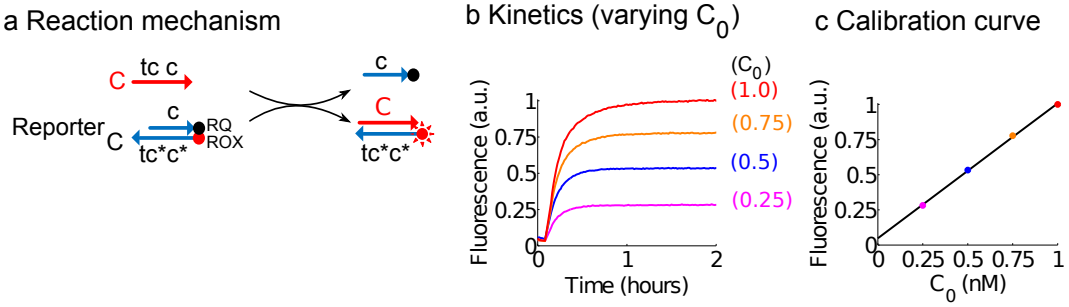


Figure S2: Reporting strategy for reaction kinetics used in this paper. a, A fluorescent reporter for signal C was used to follow the reactions. The reporter consists of two strands, one labeled with fluorescent dye (red dot), the other with a quencher (black dot). Fluorescence is quenched when dye and quencher are co-localized; displacement of the quencher-labeled strand by signal *C* leads to an increase in fluorescence. **b**, Time evolution of the reporter; the concentration of Reporter_C is 3x (1x=1nM), and concentrations of *C* are indicated in the figure. The increase in fluorescence read out by a spectrofluorometer is proportional to the amount of C that gets inactivated by binding to the reporter gate. **c,** Calibration of fluorescence signal based on the measured reaction end points (2 hours in (b)).

S2 Design and processing cycle of plasmid-derived gates

S2.1 Design of the ndsDNA gates

The enzymatic processing of ndsDNA gates in the main paper is detailed in Fig. S3 (domain sequences of Fig.2 and Fig.3 are shown in Table S5; domain sequences of the consensus network are shown in Table S6). For join gates, the restriction enzyme PvuII-HF is used to release the gates from the plasmid, and the nicking enzyme Nb.BsrDI is used to generate nicks in the top strand. For fork gates, the restriction enzyme PvuII-HF cuts the plasmid to release the gates, and then the nicking enzyme Nt.BstNBI is used to nick the top strand of the fork gate.

There are two advantages to this design. First, we can select any high turnover restriction enzyme to cut the gates because there is no constraint on the sequences of toeholds, where the recognition sites of restriction enzymes are. This is desirable because high turnover enzymes can increase the yield of correctly processed gates. Furthermore, the combinations of restriction enzyme PvuII-HF and nicking enzymes Nb.BsrDI and Nt.BstNBI do not lead to secondary structures on signal species or auxiliary strands. This is very important becuase any secondary structure on signal or auxiliary strands can significantly slow down the reaction rate of strand displacement.

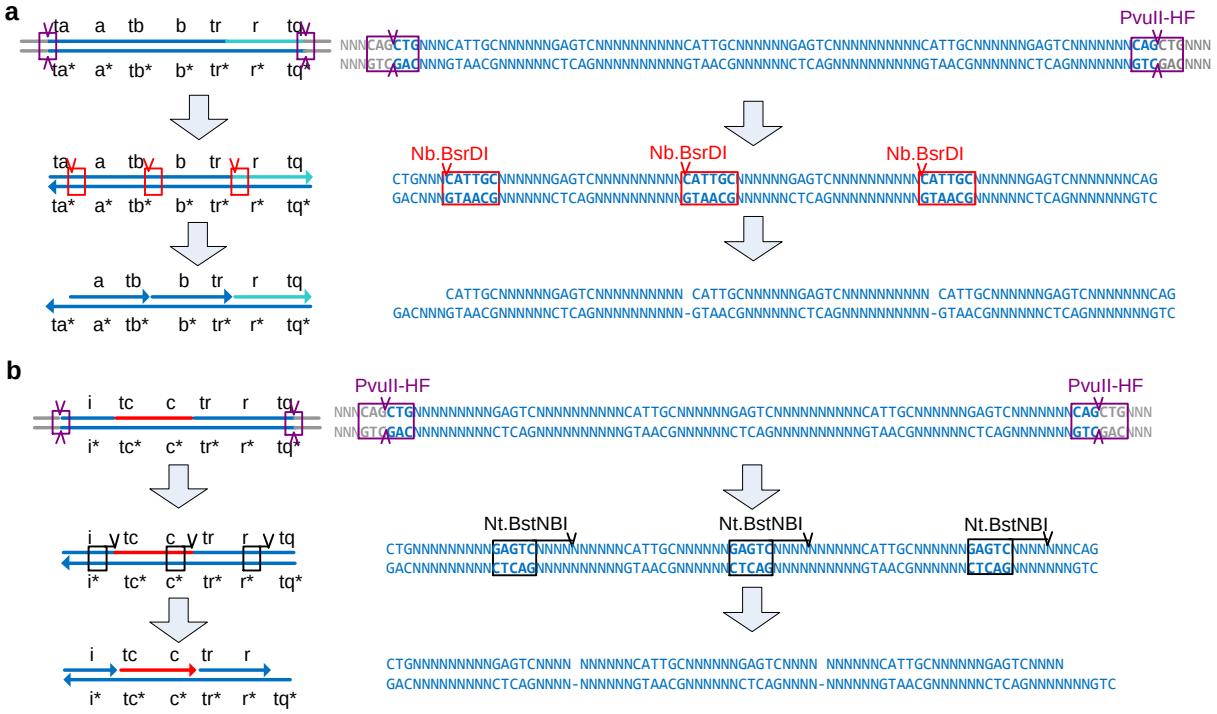


Figure S3: Enzymatic processing of ndsDNA gates in the main paper. a, Enzymatic processing of a join gate. Join gates are released by the digestion with the restriction enzyme PvuII-HF. Then the nicking enzyme Nb.BsrDI is used to generate nicks in the top strand. **b**, Enzymatic processing of a fork gate. Fork gates are released by digestion with the restriction enzyme PvuII-HF. Then, the nicking enzyme Nt.BstNBI is used to generate nicks in the top strands.

S2.2 Alternative design for testing nicking enzyme in S3.1

This section introduces another enzymatic processing of ndsDNA gates (domain sequences in Table S7), and we use this design to test the efficiency of nicking enzyme in S3.1. This design requires four different restriction enzymes and two nicking enzymes as shown in Fig. S4. For the join gates, the nicking enzyme Nb.BsrDI generates nicks in the top strand. The join gates are then released from the plasmid using the two restriction enzymes BglII and BsrBI. For the fork gates, the nicking enzyme Nb.BbvCI is used to generate nicks in the top strand, and the fork gates are then released from the plasmid by the digestion of the restriction enzymes CviQI and NheI-HF.

Because the nicking site of Nb.BbvCI coincides with the toehold, the only repeated sequences in the long domain is the nicking site of Nb.BsrDI. This is advantageous because having less repeated sequences in the long domain can avoid undesired crosstalk with other strands. However, there are two disadvantages in this design. First, the enzyme recognition sites of Nb.BsrDI and Nb.BbvCI impose constraints on sequences, which leads to secondary structures on the signal species and auxiliary strands (Fig. S4c). Even such relatively weak secondary structure can dramatically slow strand displacement kinetics. Second, since the recognition site of Nb.BbvCI will be damaged by the restriction enzyme NheI-HF, the gates have to be nicked before the digestion of NheI-HF. However, the turnover of nicking enzyme is very low when the gates are embeded in either a supercoiled plasmid or a linearized plasmid (as discussed in supplementary material S3.1).

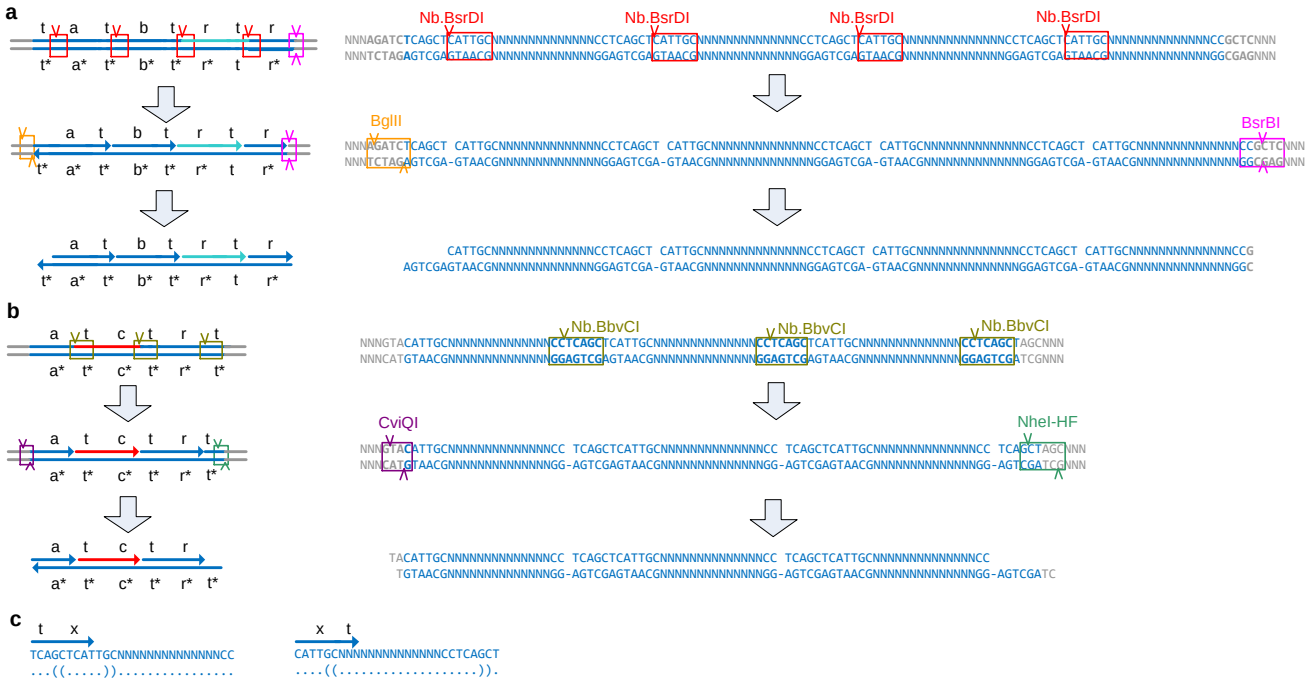


Figure S4: Enzymatic processing of producing ndsDNA gates with identical toeholds. **a**, Enzymatic processing of a join gate. The nicking enzyme Nb.BsrDI is used to generate nicks in the top strand. Restriction enzymes BglII and BsrBI are used to release the gates from the plasmid. **b**, Enzymatic processing of a fork gate. The nicking enzyme Nb.BbvCI is used to nick the top strands. Two restriction enzymes CviQI and NheI-HF are used to release gates from the plasmid. **c**, The nicking sites of Nt.BbvCI and Nb.BsrDI constrain sequences, and both signal strands and auxiliary strands have secondary structures. An unpaired base is represented by a dot, and each base pair by matching parentheses.

S3 Optimization of enzymatic processing

S3.1 Efficiency of nicking enzyme for different substrates

In this section, we compare the efficiency of nicking enzyme Nb.BsrDI for different substrates, namely, (a) gates embedded in a supercoiled plasmid (Fig. S5a), (b) gates embedded in a linearized plasmid (Fig. S5b), and (c) individual gates released from the plasmid backbone (Fig. S5c). The nicking enzyme Nb.BsrDI was tested on the join gates (for detailed enzyme processing, see S2.2), and the gates were analyzed in 10% denaturing PAGE gel. While we increased the amounts of nicking enzyme per unit of plasmid, the incomplete digestion products (71 nt and 43 nt) were still visble (Fig. S5a and Fig. S5b), which indicates that the efficency of the nicking enzyme is low for either linearized or supercoiled plasmid DNA. In contrast, the amounts of incomplete digestion products (71 nt and 43 nt) were minimal in the case where individual gates were released from the plasmid backbone (Fig. S5c), which reflects a higher efficiency of the nicking enzyme.

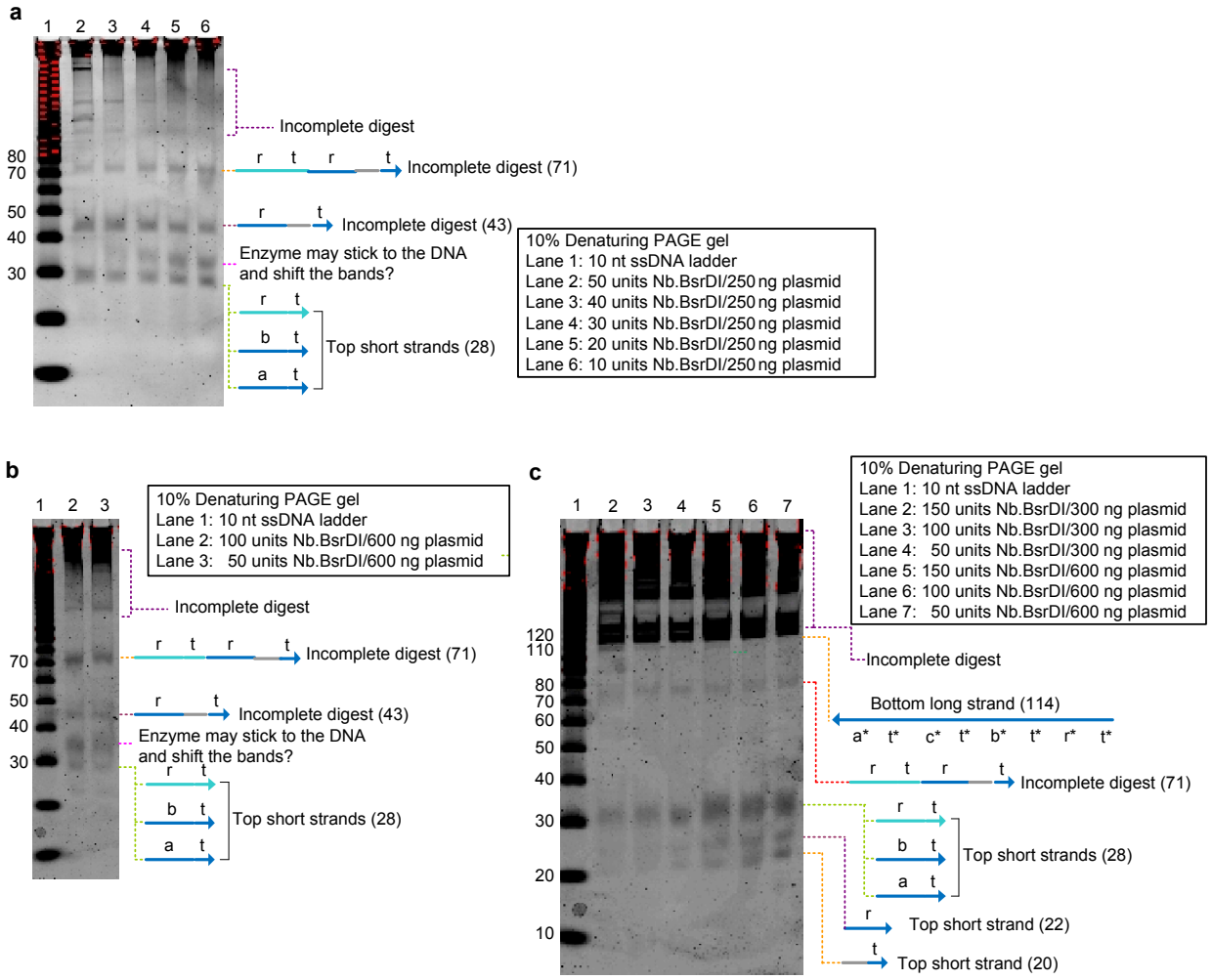


Figure S5: Comparison of nicking enzyme turnover for different substrates. $Join_{AB}$ was nicked for different substrates, and 10% denaturing-PAGE was used for the analysis of the enzyme efficiency. **a**, gates embeded in supercoiled plasmid DNA were digested using different amounts of the nicking enzyme Nb.BsrDI for 8 hours. Bands corresponding to the short top strands (28 nt) can be seen clearly in the gel. Incomplete digestion products (71 nt and 43 nt) are also visible in the gel. **b**, Gates embeded in linearized plasmid DNA were digested using different amounts of the nicking enzyme Nb.BsrDI for 8 hours. Incomplete digestion products (71 nt and 43 nt) are visible in the gel. **c**, $Join_{AB}$ gates were nicked using the nicking enzyme Nb.BsrDI for 8 hours after the gates were released from plasmids through digestion with the restriction enzymes BglII and BsrBI. Bands corresponding to the short top strands (28 nt) and bottom long strand (114 nt) can be seen in the gel. The incomplete digests (43 nt and 71 nt) shown in *a* and *b* are much more than in *c*, which indicates that the nicking enzyme was more efficient after the gates were released from plasmids.

S3.2 Optimization of nicking enzyme digest

We varied the amounts of nicking enzymes to investigate how much nicking enzyme is necessary to process the plasmid-derived gates (the gates were analyzed in 10% denaturing PAGE gel). The nicking enzyme Nb.BsrDI was tested in $Join_{AB}$ gates. There is no visible incomplete digestion products in every test amount of the nicking enzyme Nb.BsrDI (Fig. S6a), but we observe some negative effect on the gate performance when excess amounts of enzymes were used (this is further discussed in S3.4). The nicking enzyme Nt.BstNBI was tested in $Fork_{BC}$ gates. While we increased the amount of nicking enzyme Nt.BstNBI, the incomplete digestion products (37 nt, 48 nt, 70 nt, and 75 nt) were reduced (Fig. S6b). The incomplete digest (33 nt) was visible in the case of 50 units of nicking enzyme per 2.5 μg plasmid (lane 2), which could be caused by star activity.

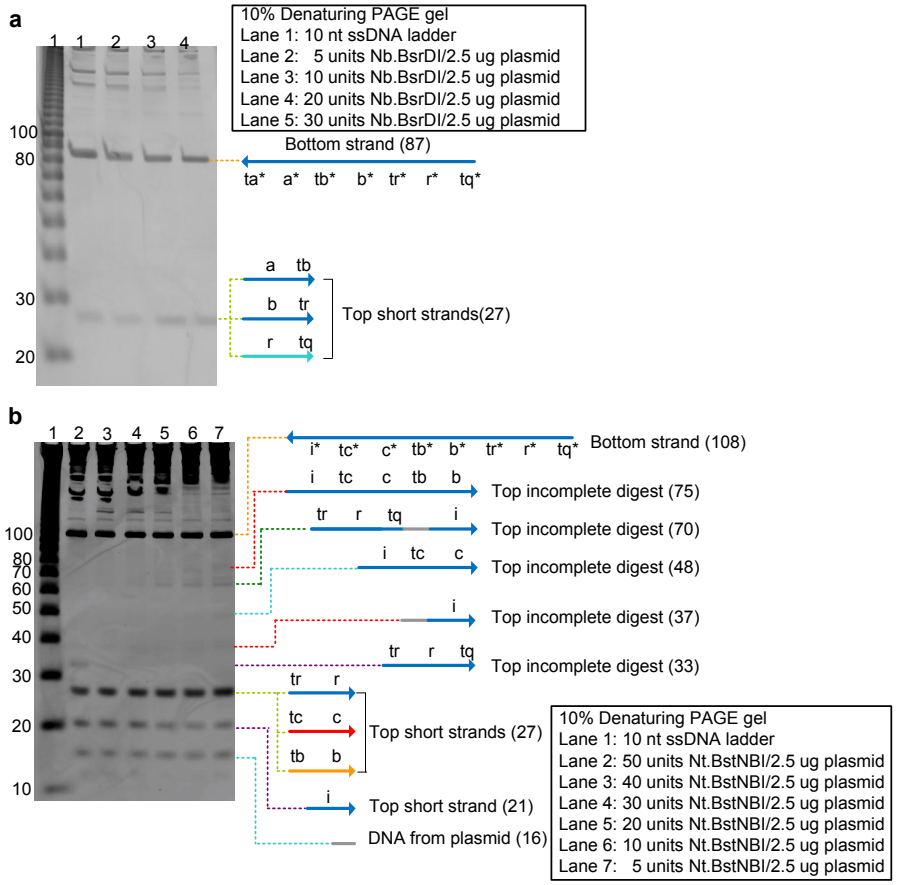


Figure S6: Enzyme amount test for plasmid-derived ndsDNA gates. **a**, 2.5 μg plasmid DNA of $Join_{AB}$ was digested with varying amounts of the nicking enzyme Nb.BsrDI for 8 hours. Bands corresponding to the long bottom strand (87 nt) and to the three short top strands (27 nt) can be seen clearly. Additional high molecular weight bands result from incompletely digested gates and from the plasmid backbone. **b**, 2.5 μg plasmid DNA of $Fork_{BC}$ was digested with varying amounts of the nicking enzyme Nt.BstNBI for 8 hours. The top strand (27 nt) and bottom strand (108 nt) can be clearly seen in the gel. While increasing the enzyme amount, the amounts of bands corresponding to incomplete digest (37 nt, 48 nt, 70 nt, and 75 nt) were reduced. The incomplete digest (33 nt) was present in lane 2, and this may be caused by star activity.

S3.3 Gel analysis of enzymatic processing of ndsDNA gates

Denaturing PAGE gels was used to verify that the ndsDNA gates were formed properly using our enzymatic processing. Fig. S7a and Fig. S7b compare plasmid-derived gates (lane3) to synthesized gates (lane2) for $Fork_C$ and $Fork_{BC}$, respectively. Plasmid-derived gates not only show correct length of the long bottom strand and short top strands, but more importantly, plasmid-derived gates do not show the impurity bands which appear as a smear (under the long bottom strand) in the synthesized gates. This verifies that plasmid-derived gates could generate higher purity gates than synthesized gates, which allows us to get dramatically better results than we might have achieved with synthetic DNA (the circuit performance is discussed in S4.1). The enzymatic processing of $Fork_{CBB}$, $Fork_{BCB}$, $Fork_{BBC}$ were verified using denaturing PAGE gel (Fig. S7c). Correct enzymatic processing of

the gate constructs can be clearly seen in the gel (the band of 135 nt corresponds to the long bottom strand, and the bands of 21 nt and 27 nt correspond to the short top strands). A small portion of incomplete digestion products are also visible in the gel, and S7.3.1 discuss why these complexes do not significantly affect the performance of the gates. The low mobility bands seen in the lanes where plasmid-derived gates were loaded are due to the undigested plasmids backbone.

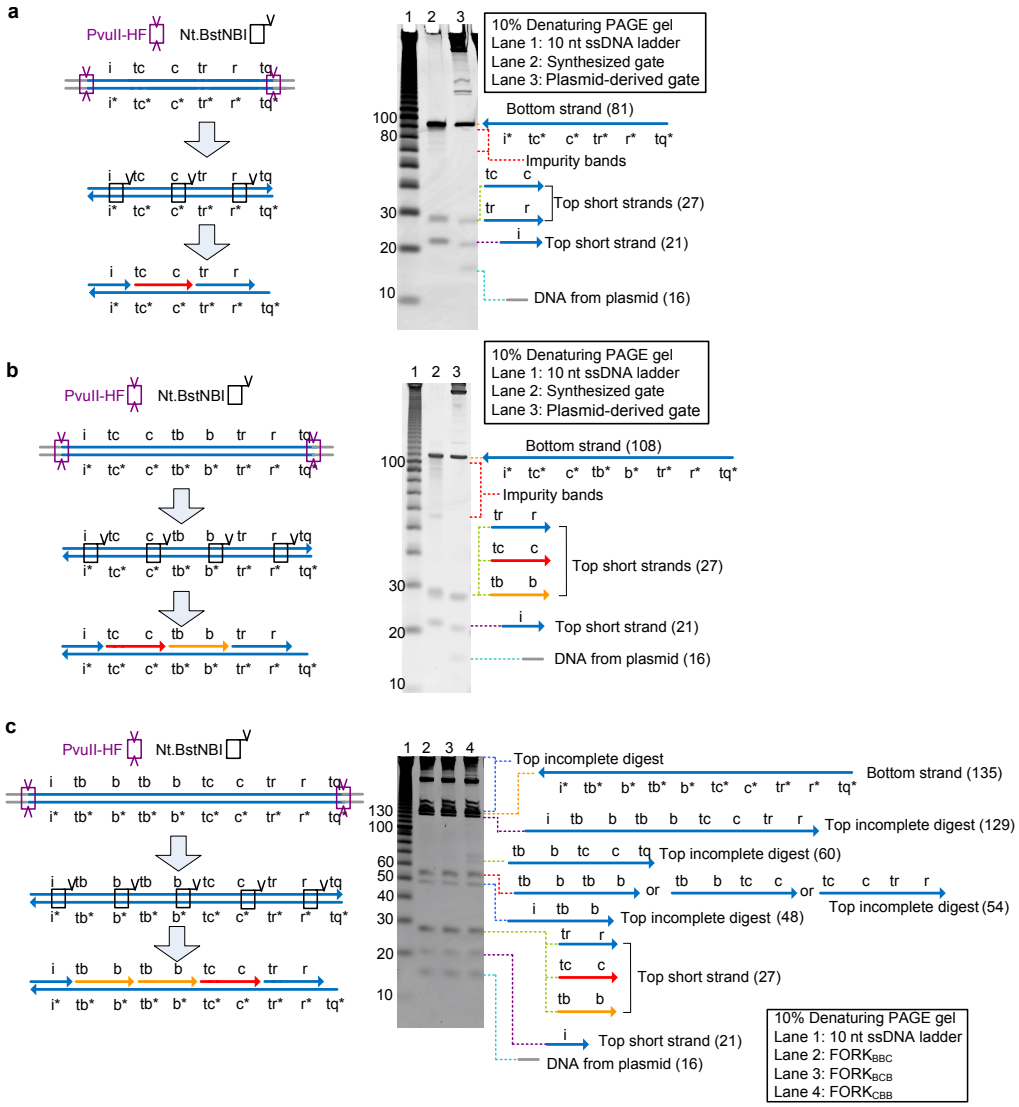


Figure S7: Denaturing gel analysis of enzymatic processing of ndsDNA gate production. a, Analysis of Fork$_C$. Lane 1: 10 nt ssDNA ladder; lane 2: Synthesized control gate; lane 3: Plasmid-derived gate. Bands corresponding to the long bottom strand (81 nt) and to the short top strands (21 nt and 27 nt) can be seen clearly in the gel. **b**, Analysis of Fork$_{BC}$. Lane 1: 10 nt ssDNA ladder; lane 2: Synthesized control gate; lane 3: Plasmid-derived gate. Bands corresponding to the long bottom strand (108 nt) and to the short top strands (21 nt and 27 nt) can be seen clearly in the gel. The synthesized control gates (lane 2) show some impurity strands (the smear under the 108 nt long bottom strand). **c**, Analysis of Fork$_{CBB}$, Fork$_{BCB}$, Fork$_{BBC}$ (plasmid-derived gates). Long bottom strands (135 nt) and short bottom strands (27 nt and 21 nt) can be seen in the gel. An incomplete digest (54 nt) is visible in the gel. Additional high molecular weight bands result from the plasmid backbone can be seen in the lanes of plasmid-derived gates.

S3.4 Impact of over-digestion on the ndsDNA gates

We varied the amounts of the nicking and restriction enzymes to investigate how the degree of digestion affects the gate performance in a functional assay (Fig. S8). The initial leakage of the plasmid-derived gates significantly increased upon addition of excess amounts of either/both of the nicking and restriction enzymes. This is most likely caused by the off-target activity of the enzymes (non-specific digestion of the gates). We found that the

optimal enzyme amount was 4 units each, for both the restriction and nicking enzymes per 1 μg plasmid DNA, and we used this optimal enzyme amount for all experiments in this paper (except where explicitly mentioned).

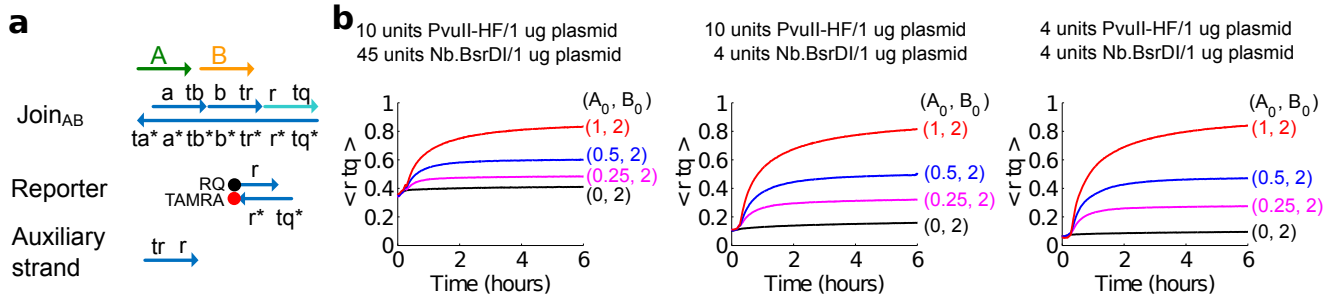


Figure S8: Circuit performance with different enzyme amounts. a, Diagram of plasmid-derived ndsDNA Join$_{AB}$. **b**, Kinetics experiments with plasmid-derived Join$_{AB}$ processed with different enzyme amounts. All auxiliary strands were at 2x(1x = 10 nM). The gate complex was 1.5x, and the experiments were performed at 35°C in 1x TAE/Mg^{2+}.

S3.5 Enzyme dissociation and ndsDNA gate behavior

Plasmid-derived gates were not purified from the enzymes after the digestion step in order to both reduce processing time and increase yield. We found that the presence of the enzymes did not negatively impact the gate performance if the enzymes were heat inactivated or if a small amount of sodium dodecyl sulfate (SDS) was added to the reaction mix for kinetics experiments (Fig. S9). The completion level of plasmid-derived gates with enzyme dissociation (either heat inactivation or addition of SDS) is clearly higher than the completion level of the gates without enzyme dissociation. The data suggest that without proper enzyme dissociation, enzymes can remain attached to their binding sites and interfere with correct gate operation. Therefore, 0.15% SDS was used for all kinetics experiments using plasmid-derived gates in this paper (except where explicitly mentioned).

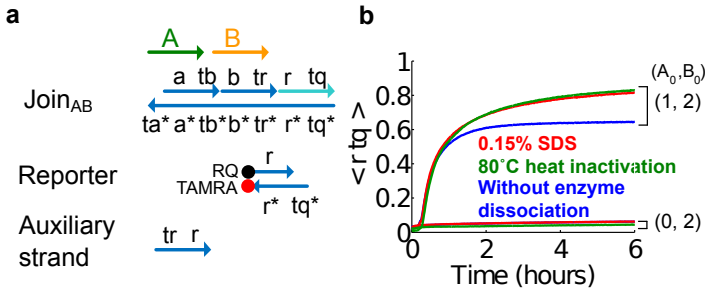


Figure S9: Enzyme dissociation and circuit behavior. a, Diagram of the ndsDNA gate tested here. **b**, Kinetics experiments of the plasmid-derived Join$_{AB}$ using 80°C heat inactivation (green traces), 0.15% sodium dodecyl sulfate (SDS) (red), and a control without heat inactivation or addition of SDS (blue). The standard concentration was 1x=10nM, and all auxiliary strands and input B were at 2x. The gate complex was 1.5x, and the experiments were performed at 35°C in Tris-acetate-EDTA buffer containing 12.5 mM Mg^{2+}.

S4 Comparison between plasmid-derived gates and synthesized gates

S4.1 Circuit performance for fundamental reaction types

In this section, we tested DNA implementation of non-catalytic, catalytic, autocatalytic reactions using plasmid-derived gates, and compare the performance to gates assembled from synthesized DNA. In Fig. S10, panel (i) shows signal strands, gate complexes and auxiliary strands used for the reaction. Panel (ii) and (iii) show the reaction kinetics of plasmid-derived and synthesized gates, respectively. In the test of $A + B \rightarrow C$ (Fig. S10a), the amount of produced product should be exactly the same as the amount of the limiting reactant (in this case, A_0 is the limiting reactant). The completion level versus initial amounts of input signal A is plotted in panel (iv). The synthesized gates show slightly higher leakage and lower completion levels for high concentration of signal A compared to the plasmid-derived gates.

In Fig. S10b, we build the catalytic reaction $A + B \rightarrow C + B$ using both plasmid-derived gates and synthesized gates. Catalytic reactions are expected to be sensitive to gate purity because capture of an input by an impurity or lack of release of an output can interrupt the catalytic cycle. The data in the panel (iv) of Fig. S10b shows that turnover is at least twice as high for the plasmid-derived gate as for the synthetic gate. Here, we define turnover as the number of moles of C that are released for each mole of the catalytic signal B in a particular period of time. For this calculation, the leak reaction (0x input) is subtracted from all traces. Data is only shown for low concentrations since at high concentrations of the catalyst B all available gates will be turned over in both cases. Given large excess of substrate, the turnover should linearly increase over time for an ideal catalyst. Deviation from the linear dependence can indicate sequestration of the catalyst through an undesirable side reaction.

Autocatalytic reactions are extremely sensitive to spontaneous "leak" reactions because any catalytic signal that is released from a faulty gate complex can trigger an exponential amplification cascade. To test for the presence of such leaky reaction pathways in our system, we engineered the autocatalytic reaction cascade $A + B \rightarrow C + 2B$. In this case, the fork gate Fork_{CBB} releases two copies of the catalytic trigger strand B together with one copy of the C signal that is used for readout (Fig. S10 c(i)). The join gate is the same as in the previous examples. We found that plasmid-derived gates were considerably less leaky than synthesized gates and had a much lower rate of untriggered amplification (black traces Fig. S10c (ii) and (iii)). We believe that improved performance with plasmid-derived DNA is the result of improved DNA quality. The quality of synthetic DNA decreases with length due to a (very small) error probability in each synthesis step, while the quality of plasmid DNA is independent of length.

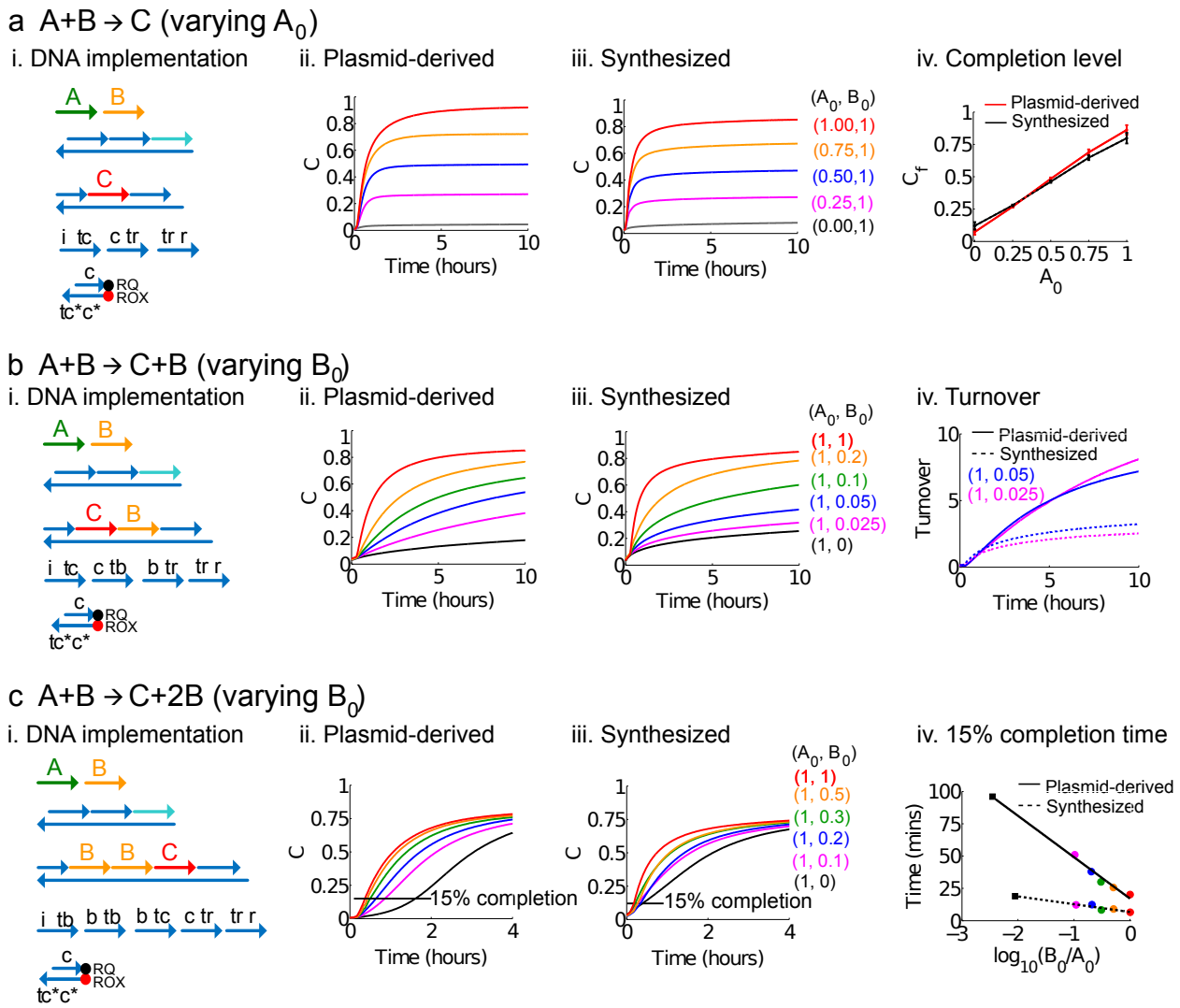


Figure S10: Kinetics data for non-catalytic, catalytic and autocatalytic bimolecular reactions. NdsDNA gates can be used to implement reactions with different kinetics. Panel (i) in each row shows a simplified representation of the ndsDNA gates and signal strands used for the corresponding experiments. Kinetics data for plasmid-derived gates are shown in panel (ii) and data for synthetic gates are shown in panel (iii). A fluorescent reporter for signal C was used to follow the reactions in all experiments. All join and fork gates were at 1.5x, 1x=50nM, and auxiliary strands were at 2x. Experiments were run in Tris-acetate-EDTA buffer containing 12.5mM Mg^{2+} (1x TAE/Mg^{2+}). **a**, Kinetics data for the bimolecular reaction $A + B \rightarrow C$. Signal B was at 2x and different amounts of signal strand A were added as indicated in the figure. Plasmid-derived gates demonstrate similar kinetics to the gates assembled from synthetic DNA. The experiment was run at 25°C. Panel (iv) shows reaction completion levels. Endpoints for all data traces in the second and third panel are shown as a function of the concentration of signal A. **b**, Bimolecular catalytic reaction $A + B \rightarrow C + B$. Signal A was at 1x and different amounts of the catalytic signal B were introduced into the system. The reaction was tested at 35°C. Panel (iii) shows that plasmid-derived gates exhibited higher turnover than synthesized gates. For this analysis, the leak reaction (0x input) was subtracted from the traces with 0.025x and 0.05x catalyst. Then, each trace was divided by the corresponding catalyst concentration to obtain a turnover number, that is a measure for the amount of product produced per unit of catalyst at a given time during the reaction. **c**, Autocatalytic reaction $A + B \rightarrow C + 2B$. Signal A was at 1x and the amount of signal B was varied. Panel (iv) shows a linear fit of the 15% completion time against the logarithm of the relative concentration of signal B. The faster reaction speeds and thus higher slope observed for the synthesized system are likely due to the higher leak in that system.

S4.2 Processing times and cost for making plasmid-derived and synthesized gates

Table S1 and Table S2 compare processing times and cost for making plasmid-derived and synthesized gates. Our primary motivation for the use of plasmid-derived DNA is the improved performance for our application. However, we find that the process for making plasmid-derived gates is slightly cheaper and takes a comparable amount of time to the assembly and purification of gate complexes from synthetic DNA. Our enzymatic processing

method requires plasmid extraction (~2 hours), two steps of enzyme digestion (~2 hours), and buffer purification (~2 hours). In sum, the whole process takes about six hours. In contrast, synthesized gates requires anealing of gates (~2 hours), and PAGE gel purification of gate complexes (~7 hours). In sum, the whole process of making synthesized gates takes about nine hours. The cost of plasmid-derived gates depends on the usage of different enzymes. To yield 300 pmole of the gates (which is enough to run 15 reactions at 30 nM), it costs ~$66 for the Join gates (which uses PvuII-HF and Nb.BsrDI), and it costs ~$99 dollars for the Fork gates (which uses PvuII-HF and Nt.BstNBI). To generate an equal amount of gates assembled from synthesized strands, a gate complex of 100 bp costs ~$260 according to the prices of IDT (including the PAGE purification fee). The enzyme price is based on NEB. We have not included the cloning procedure in this analysis because, like DNA synthesis, it can be outsourced to a commercial gene synthesis company. Furthermore, although the initial cloning procedure may be more costly than direct synthesis of the gates, the resulting plasmids can be propagated cheaply and gates can be prepared from them repeatedly.

Table S1: Processing times comparison between plasmid-derived and synthesized gates

Plasmid-derived gates		**Synthesized gates**	
Processing	Time consumption	Processing	Time consumption
Plasmid extraction	~2 hours	Annealing	~2 hours
Two steps of enzyme digestion	~2 hours	PAGE purification	~7 hours
Buffer purification (ethanol precipitation)	~2 hours		
Total	~6 hours	Total	~9 hours

Table S2: Cost comparison between plasmid-derived and synthesized gates.

Plasmid-derived gates		**Synthesized gates**	
Description	Cost	Description	Cost
Plasmid extraction (QIAGEN)	~$26	PAGE ultramer (100nt bottom strand)	~$75
Restriction enzyme	~$11 (PvuII-HF)	PAGE purified DNA oligo (30nt top strand)	~$185
Nicking enzyme	~$62 (Nt.BstNBI, Fork gate) ~$29 (Nb.BsrDI, Join gate)		
Total	~$99 (Fork gate) ~$66 (Join gate)	Total	~$260

S5 The convergence of the kinetics of the strand displacement-level model to the target CRN

We desire that the DNA realization of a formal CRN implemented according to our design schema behaves quantitatively close to the target CRN. This section substantiates this expectation of kinetic equivalence with a theoretical argument based on the strand displacement-level model. While we cannot experimentally build and test all CRNs, this argument supports the generality of our construction even for untested CRNs.

We show that the strand displacement-level model converges to the desired CRN kinetics in the limit of high concentration of gates and auxiliary species relative to the concentrations of the signal species ("CRN regime"). (The strand displacement model describes each strand displacement event as a elementary reaction. This is the mechanistic model used throughout this paper; see Section S7.) We start with a single reaction $A + B \to C$, and then generalize to arbitrary systems of bimolecular reactions in the following section. A similar argument was developed in [11], but for a different strand displacement implementation of CRNs. The argument presented here is an informal one (for a more rigorous argument, a singular perturbation analysis could be used as in [11].) The convergence argument also allows us to derive the effective rate constants for the formal CRN reactions as functions of strand displacement-level rate constants and concentrations, providing a systematic way to program rates.

S5.1 Theory of convergence for $A + B \to C$

We first outline an argument that our implementation of the formal reaction $A + B \to C$ satisfies the ideal bimolecular reaction rate law

$$\frac{d[C]}{dt} = -\frac{d[A]}{dt} = -\frac{d[B]}{dt} = [A][B]k.$$

The appropriate regime is one with high relative concentration of gates and auxiliary species. The bimolecular rate constant k is given by equation 8 below. This supports the kinetic correctness of our implementation of the formal reaction, and further equation 8 can be used to design the kinetic rate constant of the target reaction. In the next section, we provide numerical simulations illustrating this convergence, and conclude with experimental evidence corroborating our derivation.

We make a number of assumptions, the most important of which is that the gates, auxiliary, and backward auxiliary species (Join_{AB}, Fork_C, $\langle tr\ r\rangle$, $\langle i\ tc\rangle$, $\langle a\ tb\rangle$, $\langle b\ tr\rangle$, $\langle c\ tr\rangle$) are in large excess compared to the concentrations of the signal species. These amounts are assumed to be large enough that they stay effectively constant during the time-course of the experiment, and thus their instantaneous concentrations $[\cdot]$ are the same as their initial concentrations $[\cdot]_0$. Further, we suppose the excess is large enough that a strand displacement reaction involving one of these high concentration species proceeds much faster than any strand displacement reaction between other species. Per our strand displacement-level model, each strand displacement step is a bimolecular elementary reaction $G + \langle S\rangle \to G' + \langle S'\rangle$ where G is the original gate, $\langle S\rangle$ is the displacing strand, $\langle S'\rangle$ is the displaced strand, and G' is the resulting gate. The strand displacement reaction is reversible or irreversible depending on whether or not it leaves a toehold (see Section S1.1). For generality, we allow each strand displacement rate constant to be different.

For this analysis we need to be careful about distinguishing formal species A, B, C from their corresponding signal strands, which we'll write as $\langle A\rangle, \langle B\rangle, \langle C\rangle$. We'll use the notation Join_{AB}-i to mean the Join_{AB} gate after i strand displacements (i.e. with the $i+1$st toehold from the left open). Similarly, for the Fork_C gate, we write Fork_C-i to mean the gate after i strand displacements (i.e. with the $i+1$st toehold from the right open, except for Fork_C-3 which is fully double-stranded). To aid classifying which reactions are relatively fast and which are relatively slow, we box the reactants present in high concentration. We assume that the experiment does not proceed long enough to change these designations. Then the Join_{AB} strand displacements are modeled by the following reactions:

$$\boxed{\text{Join}_{AB}} + \langle A\rangle \underset{k_2}{\overset{k_1}{\rightleftharpoons}} \text{Join}_{AB}\text{-}1 + \boxed{\langle a\ tb\rangle} \tag{1}$$

$$\text{Join}_{AB}\text{-}1 + \langle B\rangle \underset{k_4}{\overset{k_3}{\rightleftharpoons}} \text{Join}_{AB}\text{-}2 + \boxed{\langle b\ tr\rangle} \tag{2}$$

$$\text{Join}_{AB}\text{-}2 + \boxed{\langle tr\ r\rangle} \underset{k_6}{\overset{k_5}{\rightleftharpoons}} \text{Join}_{AB}\text{-}3 + \langle r\ tq\rangle \tag{3}$$

and the Fork$_C$ strand displacements by the following reactions:

$$\boxed{\text{Fork}_C} + \langle r\ tq\rangle \underset{k_8}{\overset{k_7}{\rightleftharpoons}} \text{Fork}_C\text{-}1 + \boxed{\langle tr\ r\rangle} \tag{4}$$

$$\text{Fork}_C\text{-}1 + \boxed{\langle c\ tr\rangle} \underset{k_{10}}{\overset{k_9}{\rightleftharpoons}} \text{Fork}_C\text{-}2 + \langle C\rangle \tag{5}$$

$$\text{Fork}_C\text{-}2 + \boxed{\langle i\ tc\rangle} \xrightarrow{k_{11}} \text{Fork}_C\text{-}3 + \langle i\rangle. \tag{6}$$

The slowest reaction along the forward path is reaction 2 because it is the only forward reaction in which both reactants are present in small amounts (unboxed). The rate of this reaction is [Join$_{AB}$-1][$\langle B\rangle$]k_3. To see how the rate of producing $\langle C\rangle$ is dependent on the rate of this rate-limiting step consider the following Markov chain analysis on a single Join$_{AB}$ gate molecule. After an instance of forward reaction 2 occurs, there is a $[\langle tr\ r\rangle]_0 k_5 / ([\langle b\ tr\rangle]_0 k_4 + [\langle tr\ r\rangle]_0 k_5)$ chance that reaction 3 occurs next releasing $\langle r\ tq\rangle$ before reaction 2 reverses. Thus the rate of producing $\langle r\ tq\rangle$ is roughly

$$[\text{Join}_{AB}\text{-}1][\langle B\rangle]k_3 \frac{[\langle tr\ r\rangle]_0 k_5}{[\langle b\ tr\rangle]_0 k_4 + [\langle tr\ r\rangle]_0 k_5}. \tag{7}$$

After $\langle r\ tq\rangle$ is released, the slow reverse of reaction 3 (neither reactant is in high concentration) constitutes a barrier to undoing the preceeding progress. Thus we can suppose that once $\langle r\ tq\rangle$ is released, the strand displacement cascade 4-6 will proceed to the end. If the rate-limiting step is significantly slower than any step involving a high concentration (boxed) species, then we can ignore the delays due to the fast steps. In this regime expression 7 approximates the rate of producing $\langle C\rangle$.

Finally, we establish the relationship between [Join$_{AB}$-1] and $[A]$. Assume that reaction 1, which is fast in both directions, reaches quasi-equilibrium on a time scale faster than that of the rate limiting step (forward reaction 2). We can think of Join$_{AB}$-1 and free signal strand $\langle A\rangle$ as two forms of the formal species A, with $[A] = [\langle A\rangle] + [\text{Join}_{AB}\text{-}1]$. This is the amount of A that would be recorded when a large amount of reporter for $\langle A\rangle$ is added: Join$_{AB}$-1 reverts to $\langle A\rangle$ and reacts with the reporter on a faster time scale than reaction 2 proceeds forward. Then at quasi-equilibrium, the amount of A in the system will be divided between these two forms with $[\text{Join}_{AB}\text{-}1] = [A][\text{Join}_{AB}]_0 k_1 / ([\text{Join}_{AB}]_0 k_1 + [\langle a\ tb\rangle]_0 k_2)$. No similar quasi-equilibrium exists for B and thus $[\langle B\rangle] = [B]$. Plugging this into expression 7 yields

$$[A][B]k \text{ where } k = k_3 \frac{[\text{Join}_{AB}]_0 k_1}{[\text{Join}_{AB}]_0 k_1 + [\langle a\ tb\rangle]_0 k_2} \cdot \frac{[\langle tr\ r\rangle]_0 k_5}{[\langle b\ tr\rangle]_0 k_4 + [\langle tr\ r\rangle]_0 k_5}. \tag{8}$$

Equation 8 is our final estimate of the effective rate of producing $\langle C\rangle$, and consuming A (in either form) and $\langle B\rangle$. k is the effective bimolecular rate constant of the resulting formal bimolecular reaction $A + B \xrightarrow{k} C$. Note that, in the simplest case, if using equal amounts of all high concentration species, and $k_1 = k_2$, $k_4 = k_5$, then $k = k_3/4$ — that is the overall kinetics is that of the second forward strand displacement step slowed down by a factor of 4.

If our reaction $A + B \to C$ were a part of a larger system of coupled reactions, such that A or B participated as a first reactant somewhere else, then at quasi-equilibrium some amount of A and B would be sequestered in other Join gates. (See the "buffering effect" discussed in [11].) We generalize the above argument to an arbitrary bimolecular CRN in section S5.3.

S5.2 Simulation of convergence for $A + B \to C$, and comparison to experimentally collected kinetics

Fig. S11a visually demonstrates the convergence of the strand displacement-level model (ie reactions 1–6) to the bimolecular model with increasing concentration of the boxed (high concentration) species. The most obvious deviation from ideal bimolecular kinetics at lower concentrations of the boxed species is the initial behavior of the numerical derivative, which obtains a maximum value only after a delay. There are at least two factors contributing to this delay. First, before the pseudo-equilibrium is established between $\langle A\rangle$ and Join$_{AB}$-1, the rate of the rate-limiting step is slower than expected. Second, our Markov chain analysis ignored the delays due to strand displacement steps other than the second strand displacement of the Join$_{AB}$ gate (the rate-limiting step). This further contributes to the observed delay of driving up the production of C. These delays decrease with larger initial concentrations of the boxed species because the pseudo-equilibrium is established faster and the non-rate limiting steps are sped up. The other factor contributing to the deviation from ideal bimolecular kinetics at lower

concentration of the boxed species is that the concentrations of Join_{AB}, $\langle a\ tb \rangle$, $\langle b\ tr \rangle$ all change during the course of the simulation, while our analysis assumes they are constant. Again this approximation becomes more valid with higher concentrations of these species.

Lastly, we turn to the experimental data itself and ask how well does it corresponds to our bimolecular model and expression 8. We used the strand displacement level rate constants k_1–k_5 obtained as discussed in Section S7. Fig. S11b shows that expression 8 agrees well with the bimolecular rate constant fit to experimental data for concentrations of $\langle a\ tb \rangle$, $\langle b\ tr \rangle$ over 1x. At smaller concentrations, the assumptions of our derivation are significantly violated — particularly that these concentrations stay constant throughout the experiment. Further, Fig. S11d shows the behavior of the numerical derivative of the production of C obtained from experimental data. As predicted from the strand displacement level model, the numerical derivative begins to agree with the fitted bimolecular rate law after an initial delay.

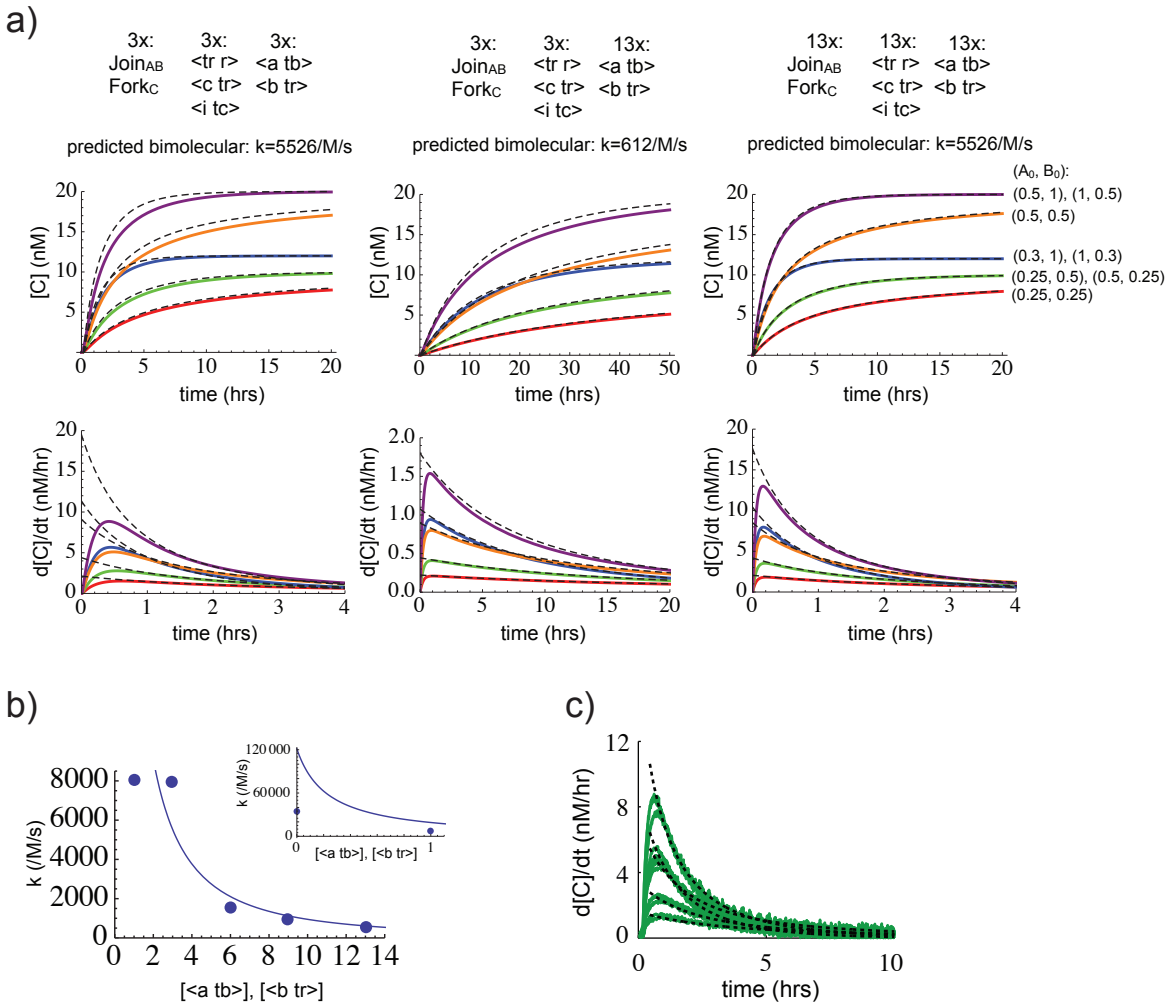


Figure S11: **a.** Convergence of the strand displacement-level simulation (ie reactions 1–6) to the bimolecular idealization. When the initial concentration of the boxed species is increased, the strand displacement-level simulation begins to be well-approximated by a simple bimolecular model $A + B \xrightarrow{k} C$ where k is given by expression 8. Dashed: bimolecular model. Solid: simulation of strand displacement reactions 1–6. **b.** The analytical prediction (expression 8) agrees well with the bimolecular rate constant fit to experimental data (shown in Fig. 4b) when the concentrations of $\langle a\ tb \rangle$, $\langle b\ tr \rangle$ are over 1x. The inset shows that the prediction deviates at 0x and 1x concentrations, because the system no longer satisfies the assumptions of our derivation. Dots: bimolecular rate constants fit to experimental data; solid line: value of expression 8. This figure is plotted on a log-scale in the main text (Fig. 4c). In both (a) and (b): 1x = 40 nM. Strand displacement-level rate constants are from section S7. **c.** Comparison of experimentally measured numerical derivative and bimolecular rate law. The rate of the formation of product C is obtained by taking a time derivative of the data in Fig. 4b (3x case), and the fitted bimolecular model is plotted as black dashed traces. Note that the bimolecular approximation becomes valid after an initial transient during which individual reaction steps equilibrate.

S5.3 Theory of convergence for arbitrary bimolecular CRNs

In this section we generalize the convergence from Section S5.1, which was confined to a single reaction, to arbitrary bimolecular CRNs.

Consider the general form of a bimolecular reaction

$$\alpha + \beta \xrightarrow{k} \gamma + \cdots + \delta \tag{9}$$

where $\alpha, \beta, \gamma, \delta$ index over species (the same species can appear in multiple places.) Analogous to the $A + B \to C$ case discussed above, the Join gate strand displacement reactions can be written in general as follows:

$$\boxed{\text{Join}_{\alpha\beta}} + \langle\alpha\rangle \underset{k_{\alpha\beta2}}{\overset{k_{\alpha\beta1}}{\rightleftharpoons}} \text{Join}_{\alpha\beta}\text{-}1 + \boxed{\langle\text{back}_{\alpha}\rangle} \tag{10}$$

$$\text{Join}_{\alpha\beta}\text{-}1 + \langle\beta\rangle \underset{k_{\alpha\beta4}}{\overset{k_{\alpha\beta3}}{\rightleftharpoons}} \text{Join}_{\alpha\beta}\text{-}2 + \boxed{\langle\text{back}_{\beta}\rangle} \tag{11}$$

$$\text{Join}_{\alpha\beta}\text{-}2 + \boxed{\langle\text{aux}_{\alpha\beta}\rangle} \underset{k_{\alpha\beta6}}{\overset{k_{\alpha\beta5}}{\rightleftharpoons}} \text{Join}_{\alpha\beta}\text{-}3 + \langle\text{trans}_{\alpha\beta}\rangle \tag{12}$$

where the "trans" strand is then input to the Fork gate. Let $j_{\alpha\beta} = [\text{Join}_{\alpha\beta}]_0$, $a_{\alpha\beta} = [\text{aux}_{\alpha\beta}]_0$, $b_{\alpha} = [\text{back}_{\alpha}]_0$, be the initial concentrations of the gates, auxiliary, and backward auxiliary strands, respectively. As before we assume that these quantities are in large excess and do not vary over the time course of the experiment. Then the same Markov chain analysis as in the single reaction case yields that the rate of the production of $\langle\text{trans}_{\alpha\beta}\rangle$ (analog of equation 7) is

$$[\text{Join}_{\alpha\beta}\text{-}1]\,[\langle\beta\rangle]\,k_{\alpha\beta3}\,\frac{a_{\alpha\beta}\,k_{\alpha\beta5}}{b_{\beta}\,k_{\alpha\beta4} + a_{\alpha\beta}\,k_{\alpha\beta5}}. \tag{13}$$

In a regime with large concentrations of the Fork gate and auxiliary species, any trans strand produced is quickly converted to reaction products $\gamma + \cdots + \delta$, and therefore the rate of the production of the trans strand is a valid the measure of the overall reaction rate.

Formal species α is split up in a pseudo-equilibrium between $\langle\alpha\rangle$, $[\text{Join}_{\alpha\beta}\text{-}1]$, and the corresponding Join-1 of other formal reactions with α as the first reactant. Similarly, formal species β is split up in a pseudo-equilibrium between $\langle\beta\rangle$ and Join-1 of other formal reactions with β as the first reactant. So in equation 13, we can approximate $[\text{Join}_{\alpha\beta}\text{-}1]$ as a constant fraction of all of $[\alpha]$, and the free strands $[\langle\beta\rangle]$ as constant fraction of all of $[\beta]$ as follows. Let

$$g_{\alpha\beta} = \frac{j_{\alpha\beta}\,k_{\alpha\beta1}}{b_{\alpha}\,k_{\alpha\beta2}}.$$

Then

$$[\text{Join}_{\alpha\beta}\text{-}1] = [\alpha]\left(\frac{g_{\alpha\beta}}{1 + \sum_{\beta'} g_{\alpha\beta'}}\right)$$

and

$$[\langle\beta\rangle] = [\beta]\left(\frac{1}{1 + \sum_{\beta'} g_{\beta\beta'}}\right).$$

Note that the first sum is over all reactions that share the same first reactant as this reaction, and the second sum is over all reactions whose first reactant is the same as the second reactant of this reaction. Therefore, we can approximate the effective bimolecular rate constant of the implemented formal reaction (9) as:

$$k = k_{\alpha\beta3}\left(\frac{g_{\alpha\beta}}{1 + \sum_{\beta'} g_{\alpha\beta'}}\right)\left(\frac{1}{1 + \sum_{\beta'} g_{\beta\beta'}}\right)\frac{a_{\alpha\beta}\,k_{\alpha\beta5}}{b_{\beta}\,k_{\alpha\beta4} + a_{\alpha\beta}\,k_{\alpha\beta5}}. \tag{14}$$

S5.4 Simulation of the convergence for the consensus network

Fig. S12 shows the convergence of the strand displacement-level model to the formal three reaction CRN of the consensus network (Fig. 5b) with increasing concentration of the boxed (high concentration) species. The strand displacement-level model is described by reactions 10–12, and the analogous reactions of the Fork gate. The rate

constants of the consensus three reaction CRN are obtain by expression 14 above. The scaling factor s is used to uniformly scale up the concentrations of the high concentration species. A larger scaling factor s corresponds to better agreement to the assumptions of our derivation ("CRN regime").

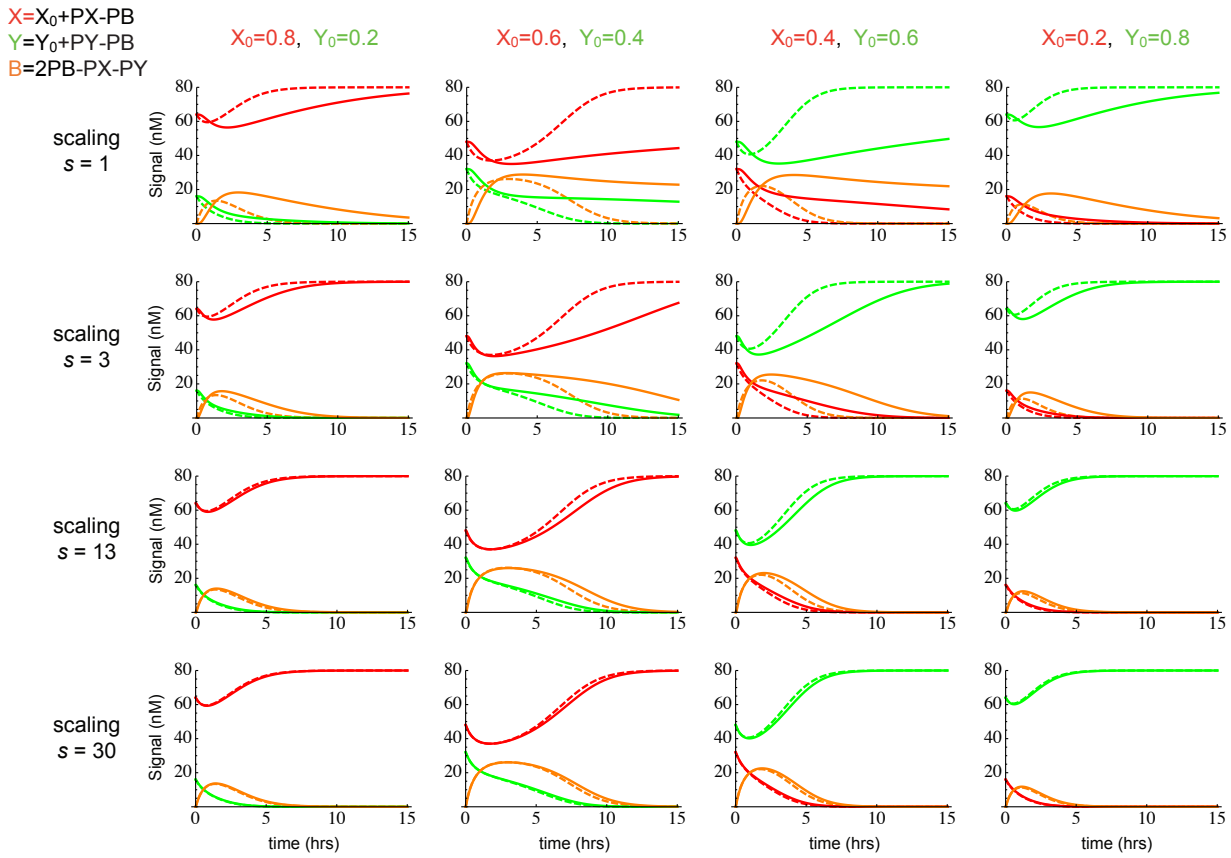


Figure S12: Convergence of the strand displacement-level simulation (reactions 10–12 and the analogous reactions of the Fork gate) to the formal CRN model for the consensus network. When the initial concentration of the boxed species is increased, the strand displacement-level simulation begins to be well-approximated by the three reaction CRN, where the rate constants of the three reactions are derived according to expression 14. Dashed: formal CRN model. Solid: strand displacement-level model. The concentrations used are as in the text (Fig. 5), except that gates and auxiliary strands are scaled by a factor of s. Further, backward auxiliary species are added at $s \cdot 2$x concentration. Strand displacement-level rate constants are from section S7. The concentrations of the signal species are calculated from PX, PY, and PB concentrations as in the main text.

S6 Demonstration of stoichiometry and conservation of mass

In this section, we test our experimental implementation of reaction $A + B \rightarrow C$ to confirm the correct stoichiometry of reactants and products. For this reaction, the amount of output C produced should be equal to the amount of the limiting reactant species by the conservation of mass (Fig. S13a). As shown in Fig. S13b, the output C has the correct stoichiometry in either case of limiting A or B (the kinetics are shown in Fig. 4b). We additionally measured the amount of the non-limiting reactant species that was left over at the end of the reaction and thus directly verified the conservation of mass (Fig. S13b).

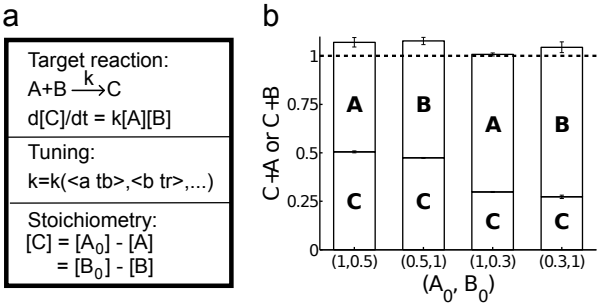


Figure S13: Stoichiometry and conservation of mass of the bimolecular reaction $A + B \rightarrow C$. a, conservation of mass and rate laws. **b**, demonstration of stoichiometry. Signal concentrations varied as indicated in the figure, 1x = 40 nM. Gates Join_{AB}, Fork_{C}, and auxiliary strands <*a tb*> and <*b tr*> were at 3x. (kinetics data are shown in Fig. 4b). The bar graph shows that signal concentrations at the reaction endpoint add up to the initial value of the majority signal as required by conservation of mass (see *a*). The end point values for *C* can be read off directly from the graph in Fig. 4b.

S7 Simulation and modeling

S7.1 Modeling approach

S7.1.1 The DSD language and simulator

Models were constructed using Visual DSD [7], a programming language for the design and analysis of DNA strand displacement devices. Visual DSD is an implementation of the programming language and compiler described in [6], and features automatic compilation of programs to strand displacement reaction networks, together with stochastic and deterministic simulation methods. Programs are written in a textual syntax, described in [6], which supports modules and local parameters to allow for abstraction and code-reuse. A DSD program defines an initial collection of DNA species, which can be single or double-stranded, and the DSD compiler then computes the set of all strand displacement reactions that can be generated from these initial species. The generated reactions can then be simulated using stochastic or deterministic methods. The Visual DSD language is freely available from http://research.microsoft.com/dna.

Here we summarize the textual syntax of the DSD programming language used in this paper. The syntax is defined in terms of elementary *sequences* and *species*. A sequence S comprises one or more *domains*, which can be *long domains* x or *short domains* tx^. A species can be an *upper strand* <S>, a *lower strand* {S} or a *gate* G. An upper strand <S> denotes a sequence S oriented from left to right, while a lower strand {S} denotes a sequence S oriented from right to left. A *double strand* [S] denotes an upper strand <S> bound to the complementary lower strand {S*}. A gate G is composed of double-stranded segments of the form {L'}<L>[S]<R>{R'}, which represents an upper strand <L S R> bound to a lower strand {L' S* R'} along the double-stranded region [S]. The overhanging sequences L, L' and R, R' can potentially be empty, in which case we simply omit them. Gates are built up by concatenating segments G1 and G2 along a common lower strand, written G1:G2. We let D range over *systems* of species. Multiple systems D1, D2 can be present in parallel, written D1 | D2. We also allow module definitions of the form def X(n)=D, where n are the module parameters and X(m) is an instance of the module D with parameters n replaced by m. We assume a fixed set of module definitions, which are declared at the start of the program.

S7.1.2 Programming methodology

DNA strand displacement models were programmed in the DSD language using a modular approach, where separate modules were programmed for each of the main circuit components. Specifically, modules were programmed for the 2-input Join component, for the 1-input, 2-input and 3-input Fork components, and for the various Reporter components. These modules were then used in combination to produce a DSD program for each of the circuits tested experimentally, including the full consensus circuit.

The DSD program for each circuit was automatically compiled to a corresponding set of *strand displacement reactions*, by using DSD in *Infinite* mode. This mode merges associated toehold binding, branch migration and toehold unbinding reactions into a single strand displacement reaction. Further modifications were made to the kinetic rates of the generated strand displacement reactions, to encode distinct hypotheses about the rate constants of interacting strands and gates.

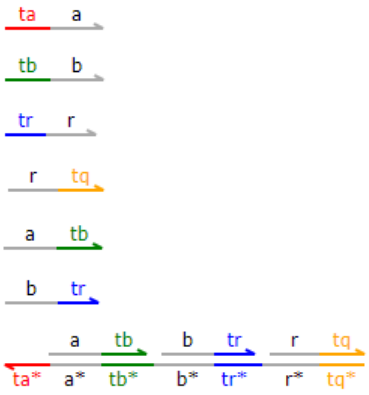
S7.2 DSD modules

All of the models simulated in this paper were derived from the following core set of Join, Fork and Reporter modules, written in the DSD language.

S7.2.1 Join module

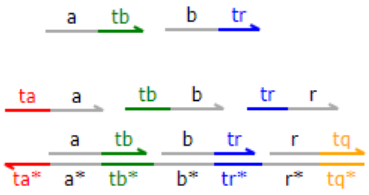
The elementary strands and gates that constitute the Join module are defined as follows, where the graphical representation of each strand or gate is shown adjacent to the corresponding program code. The definitions are parametrized by their respective domains, so that the same strand or gate can be instantiated with different domains depending on the circuit in which it is being used.

```
def Input(ta,a) = <ta^ a>

def Catalyst(tb,b) = <tb^ b>

def Helper(tr,r) = <tr^ r>

def Translator(r,tq) = <r tq^>

def InputR(a,tb) = <a tb^>

def CatalystR(b,tr) = <b tr^>

def Join_AB(ta,a,tb,b,tr,r,tq) =
 {ta^*}[a tb^]:[b tr^]:[r tq^]
```

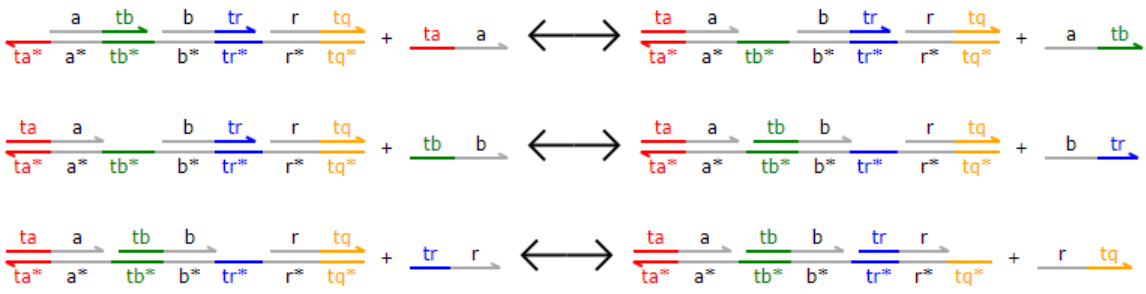


The Join module is defined in terms of these elementary strands and gates as follows:

```
def Join(NA,NB,NH,NG,NV,Leak,ta,a,tb,b,tr,r,tq) =
( NG*(1.0-Leak) * Join_AB(ta,a,tb,b,tr,r,tq)
| NG*Leak * Translator(r,tq)
| NA * Input(ta,a)
| NB * Catalyst(tb,b)
| NH * Helper(tr,r)
| NV * InputR(a,tb)
| NV * CatalystR(b,tr)
)
```



The parameters NA and NB represent the initial concentrations of Input and Catalyst strands, respectively, while the parameters NH and NG represent the initial concentrations of Helper strands and Join_AB gates, respectively. The parameter NV represents the initial concentration of *reverse* strands, InputR and CatalystR, which are used to displace the Input and Catalyst strands, respectively. The parameter Leak represents the initial proportion of leaked Translator strands, as a fraction of the initial concentration of Join_AB gates. Since experimental measurements indicated the presence of a small initial concentration of Translator strands in the absence of Input, Catalyst and Helper strands, we assumed that these Translator strands were produced from a small fraction of Join_AB gates via a *fast leak*. The Leak parameter therefore specifies the fraction of Join_AB gates that undergo this fast leak, giving rise to an initial concentration of Translator strands. The remaining module parameters represent domains that are shared between the various gates and strands, allowing the definition to be instantiated with different domains depending on the circuit in which it is being used. When compiled in Visual DSD, the Join module gives rise to the following strand displacement reactions:

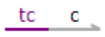


Distinct kinetic rates were assigned to each strand displacement reaction by considering the context in which the reaction takes place. Specifically, since each strand displacement reaction consists of a toehold binding, branch migration and (where applicable) toehold unbinding event, we assumed the reaction rate to be a function of the rates of these three events. Furthermore, since all long domains in a given circuit were of a similar length, we assumed that the rate of branch migration was similar for all reactions in a circuit, based on previous work showing that the rate of branch migration is primarily a function of strand length [12]. Thus, distinct rates were assigned to strand displacement reactions with distinct binding and (where applicable) unbinding toeholds. This results in the following rates for the above reactions, respectively: (k_{ab}, k_{ba}), (k_{br}, k_{rb}), and (k_{rq}, k_{qr}).

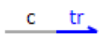
S7.2.2 Fork module

Additional elementary strands and gates that constitute the Fork module are defined as follows:

```
def Output(tc,c) = <tc^ c>
```

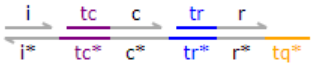



```
def OutputR(c,tr) = <c tr^>
```




```
def End(i,tc) = <i tc^>
```

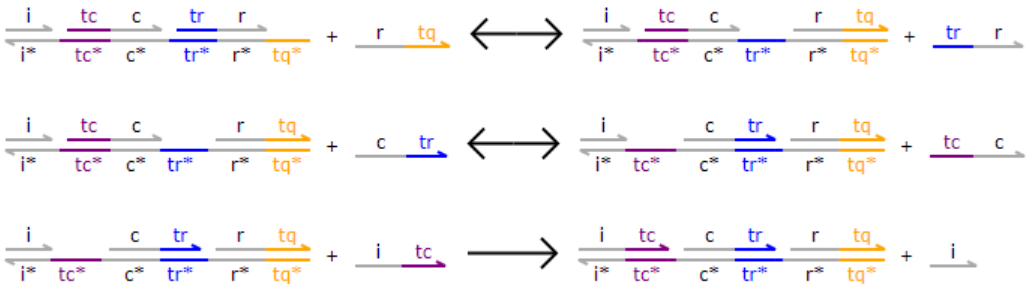

```
def Fork_C(i,tc,c,tr,r,tq) =
 [i]:[tc^ c]:[tr^ r]{tq^*}
```



The Fork module is defined in terms of these elementary strands and gates, together with the Translator strand defined previously:

```
def Fork(N,NH1,NH2,NG,Leak,i,tc,c,tr,r,tq) =
( NG*(1.0-Leak) * Fork_C(i,tc,c,tr,r,tq)
| NG*Leak * Output(tc,c)
| N * Translator(r,tq)
| NH1 * OutputR(c,tr)
| NH2 * End(i,tc)
)
```

The parameter N represents the initial concentration of Translator strands, while the parameters NH1 and NH2 represent the initial concentrations of OutputR and End strands, which are used to expel the output and lock down the Fork_C gate, respectively. The parameter NG represents the initial concentration of Fork_C gates, while the parameter Leak represents the fraction of Fork_C gates that undergo a fast leak, giving rise to an initial concentration of Output strands. When compiled in Visual DSD, the Fork module gives rise to the following strand displacement reactions:

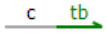


As with the Join circuit, distinct kinetic rates were assigned to each strand displacement reaction with distinct binding and (where applicable) unbinding toeholds, giving rise to the following rates for the above reactions, respectively: (k_{qr1}, k_{rq}), (k_{rc}, k_{cr}),(k_c).

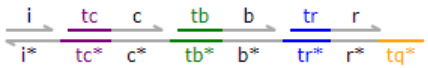
S7.2.3 Fork2 module

Additional elementary strands and gates that constitute the Fork2 module are defined as follows:

```
def Output2R(c,tb) = <c tb^>
```




```
def Fork_BC(i,tc,c,tb,b,tr,r,tq) =
 [i]:[tc^ c]:[tb^ b]:[tr^ r]{tq^*}
```

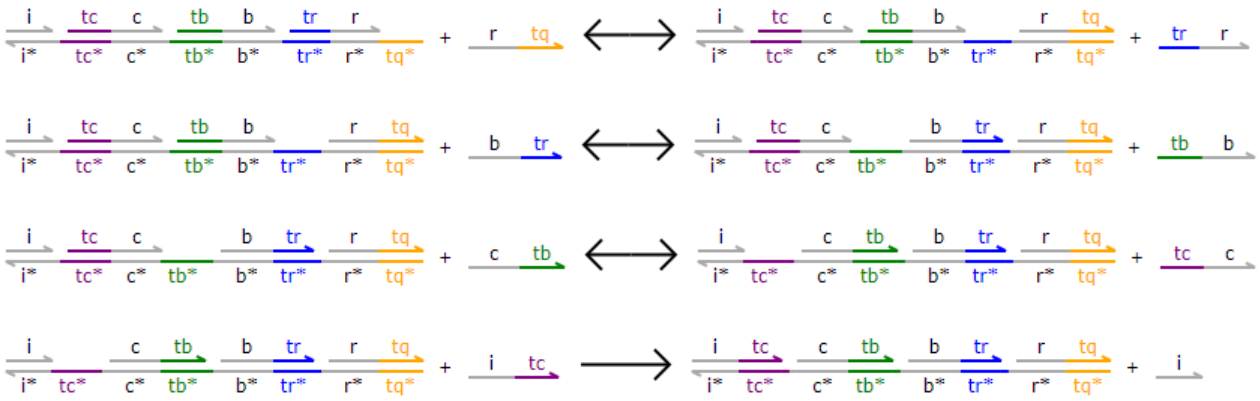


The Fork2 module is defined in terms of these elementary strands and gates, together with the previously defined Output, Translator, CatalystR and End strands:

```
def Fork2(N,NH1,NH2,NH3,NG,Leak,i,tc,c,tb,b,tr,r,tq) =
( NG*(1.0-Leak) * Fork_BC(i,tc,c,tb,b,tr,r,tq)
| NG*Leak * Output(tc,c)
| N * Translator(r,tq)
| NH1 * CatalystR(b,tr)
| NH2 * Output2R(c,tb)
```

```
| NH3 * End(i,tc)
)
```

The parameter N represents the initial concentration of Translator strands, while the parameters NH1, NH2 and NH3 represent the initial concentrations of CatalystR, OutputR and End strands, which are used to expel the catalyst and the output, and to lock down the Fork_BC gate, respectively. When compiled in Visual DSD, the Fork2 module gives rise to the following strand displacement reactions:

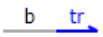


Distinct kinetic rates were assigned to each strand displacement reaction, respectively, as follows: (k_{qr2}, k_{rq}), (k_{rb}, k_{br}),(k_{bc}, k_{cb}),(k_c).

S7.2.4 Fork3 module

Additional elementary strands and gates that constitute the Fork3 module are defined as follows:

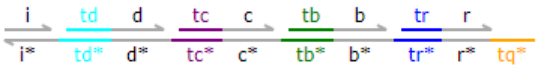
```
def Output1R(b,tr) = <b tr^>
```




```
def Output3R(d,tc) = <d tc^>
```


```
def End3(i,td) = <i td^>
```

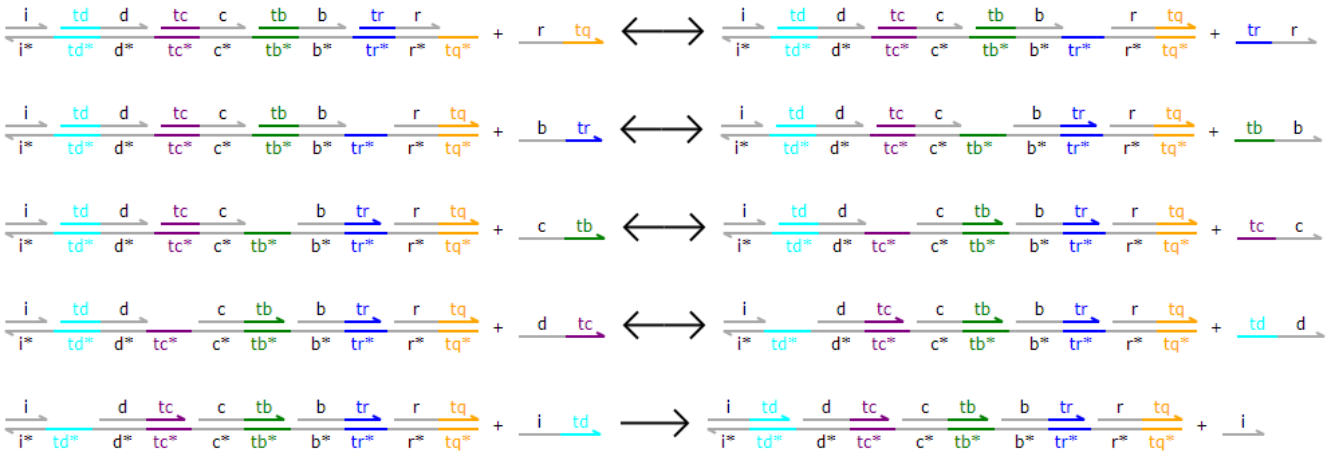

```
def Fork_BCD(i,td,d,tc,c,tb,b,tr,r,tq) =
 [i]:[td^ d]:[tc^ c]:[tb^ b]:[tr^ r]{tq^*}
```



The Fork3 module is defined in terms of these elementary strands and gates, together with the previously defined Translator and Output2R strands:

```
def Fork3(N,NH1,NH2,NH3,NH4,NG,Leak,i,td,d,tc,c,tb,b,tr,r,tq) =
( NG*(1.0-Leak) * Fork_BCD(i,td,d,tc,c,tb,b,tr,r,tq)
| N * Translator(r,tq)
| NH1 * Output1R(b,tr)
| NH2 * Output2R(c,tb)
| NH3 * Output3R(d,tc)
| NH4 * End3(i,td)
)
```

One notable difference from the definitions of the Fork and Fork2 modules is that here the Leak parameter represents the fraction of Fork_BCD gates that undergo a fast leak, without specifying the initial concentration of leaked output strands. This is because, depending on the instantiation of the Fork3 module, the output could be in any one of three possible positions. As a result, the initial concentration of leaked output strands is specified as a parameter of the Reporter module. When compiled in Visual DSD, the Fork3 module gives rise to the following strand displacement reactions:

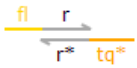


Another difference from the Fork and Fork2 modules is that the Fork3 module is never instantiated to produce the outputs (B,C,D). Instead, it is instantiated to produce the outputs (B,C,B), (B,B,C) and (C,B,B), with each combination resulting in different sets of kinetic rates.

S7.2.5 Reporter modules

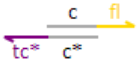
Additional elementary strands and gates that constitute the Join and Fork Reporter modules are defined as follows:

```
def Rep_Join(r,tq) = <fl^>[r]{tq^*}
```



```
def Signal_Join(fl,r) = <fl^ r>
```

```
def Rep_Fork(tc,c) = {tc^*}[c]<fl^>
```

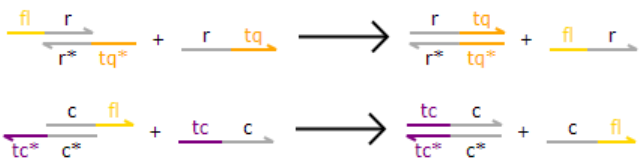


```
def Signal_Fork(c,fl) = <c fl^>
```

The RepJoin and RepFork modules are defined in terms of these elementary strands and gates, together with the previously defined Translator and Output strands:

```
def RepJoin(N,NR,r,tq) =
( N * Translator(r,tq)
| NR * Rep_Join(r,tq)
| 0 * Signal_Join(fl,r)
)
def RepFork(N,NR,tc,c) =
( N * Output(tc,c)
| NR * Rep_Fork(tc,c)
| 0 * Signal_Fork(c,fl)
)
```

The parameters N and NR denote the initial concentration of Output or Translator strands and Reporter gates, respectively. When compiled in Visual DSD, the Reporter modules give rise to the following strand displacement reactions:



S7.3 Sources of interference

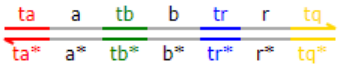
In this section we explore possible sources of interference between DNA species, and how these could potentially be incorporated in the DSD models.

S7.3.1 Incomplete digests

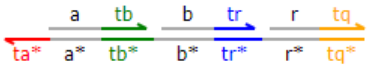
During enzymatic processing, nicking enzymes were used to cut double stranded complexes at multiple sites, in order to produce the desired nicked double stranded DNA gates. Restriction enzymes were also used to release double-stranded gates from their plasmid backbone. However, in some cases these enzymes did not cut on every recognition site, resulting in partially digested duplexes. In this section, we use the Join_AB gate to illustrate the most common reaction pathways involving partially digested duplexes, and demonstrate why these partial digestions do not significantly affect performance.

As described in S9.2, four identical gates were encoded in each plasmid, separated by spacer sequences. In some cases the restriction enzyme PvuII-HF did not cut on every position to release the gates from the plasmid, resulting in incomplete digestion products that were longer than the gate. A single uncut position resulted in an incomplete digest of ~100bp, consisting of a gate and a spacer sequence, while multiple uncut positions resulted in bands of ~200nt, ~300nt, etc., up to the length of the full plasmid. However, since the spacer sequences are double-stranded, we hypothesized that the presence of these sequences alone would not significantly affect the gate performance.

In the case of the nicking enzymes, consider the double stranded complex used to form the Join_AB gate:

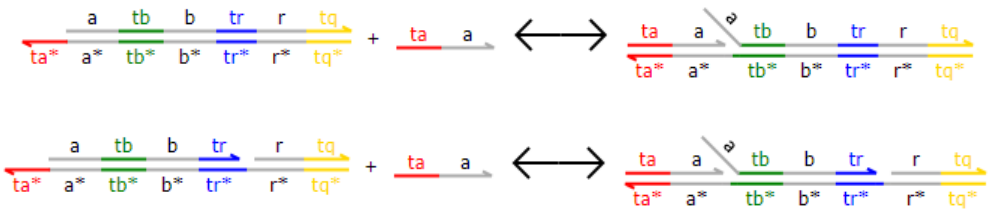


This duplex should be nicked in 3 distinct locations in order to give the desired Join_AB gate, assuming that nicked toeholds spontaneously unbind:



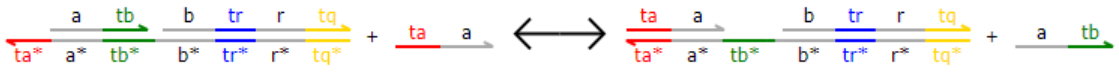
If one or more nicks fail, the set of possible digests is as follows:

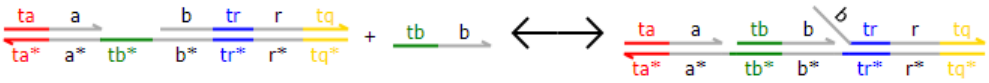
Three of these digests do not expose any toeholds, and thus cannot interact with any strands, resulting in almost no observed effect on circuit behavior. For two of the partial digests, the strand <ta a> can transiently bind to the exposed ta toehold:



However, the bound <ta a> strand is unable to displace any of the incumbent strands due to their length, and will rapidly unbind.

For one of the partial digests a strand displacement reaction is possible, and the resulting duplex could temporarily bind a <tb b> strand:





However, as before the bound <tb b> strand is unable to displace the longer incumbent strand, and will rapidly unbind.

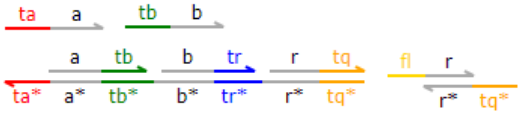
A similar approach was used to model the effects of incomplete digests on the Fork modules. In all cases, incomplete digests could only temporarily sequester a strand. This suggests that the effects of incomplete digests are relatively minor, resulting in only a transient slow-down of circuit dynamics.

S7.3.2 Leaks

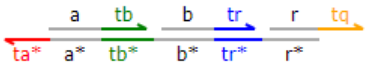
Leak reactions can be broadly defined as unintended interactions between DNA strands, and are a common occurrence in almost all strand displacement circuits (see e.g. [9, 10]). A recent study [4] highlighted such leakage as a major hindrance in the experimental implementation of amplification cascades, and identified four main types of leakage that occur in toehold-mediated strand displacement systems. Here we summarize the three types of leakage that were observed in the strand displacement circuits reported in this paper, using the Join_AB gate as an example (S7.9.4).

First, we recall that the Reporter for the Join_AB circuit consists of a strand <r> with a quencher on the 5′ end, bound to a complementary strand <r* tq*> with a fluorophore on the 3′ end. Even in the absence of the output strand <r tq>, there is still some background fluorescence observed, most likely due to breathing of the base pairs adjacent to the fluorophore, resulting in imperfect quenching. This background fluorescence was subtracted from all of the data obtained for the Join_AB circuit.

Type I leakage When the Join_AB and Reporter gates were combined in the presence of inputs A and B, a small amount of leakage was observed as an increase in fluorescence, which rapidly leveled off.

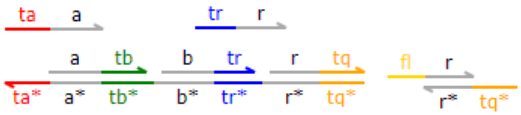


Since this leakage was also observed in the absence of inputs A and B, it corresponds directly to a type I leakage [4], characterized by a small fraction malformed Join_AB gates, interacting directly with the Reporter gate. We hypothesized that this type I leakage was a result of over-digestion by the nicking or restriction enzymes, since Fig. S8 clearly showed that increased concentrations of these enzymes resulted directly in increased leakage. A possible mechanistic explanation is that over-digestion caused additional bases to be removed adjacent to the restriction site, resulting in either the partial or full exposure of a toehold. This could give rise to the following gate, which would then interact directly with the reporter on toehold tq to displace the signal.

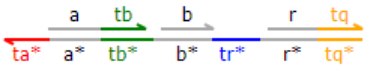


Since the amount of leakage rapidly stabilized, we hypothesized that it corresponded directly to the proportion of malformed gates.

Type II leakage When the Join_AB and Reporter gates were combined in the presence of input A and helper <tr r>, a small amount of leakage was also observed, which again rapidly leveled off.

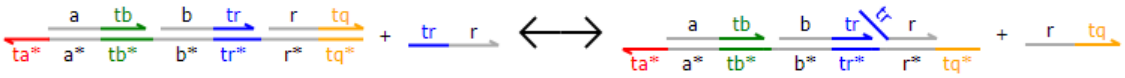


The observed leakage was higher than the type I leakage observed solely in the presence of the Join_AB and Reporter gates, and corresponds directly to a type II leakage [4], characterized by a small fraction of malformed Join_AB gates interacting with helper strands. As above, we hypothesized that this type II leakage was a result of over-digestion by the nicking or restriction enzymes. A possible mechanistic explanation is that over-digestion caused additional bases to be removed adjacent to the nicking site, resulting in either the partial or full exposure of a toehold. This could give rise to the following gate, which would then interact directly with the helper strand on toehold tr to displace the output, which in turn would displace the signal.



A similar type of leakage was also observed when the Join_AB and Reporter gates were combined in the presence of input B and helper <tr r>.

Type III leakage This type of leakage is characterized by a slow, steady increase in fluorescence that asymptotically approaches full conversion to maximum fluorescence output [4]. At least part of this is likely due to blunt-end displacement, where a strand invades due to spontaneous breathing of bases adjacent to a nicking or restriction site. As an example, consider the following leak involving the Join_AB gate and the Helper strand:



Similar leakage could also occur between the Join_AB gate and the A or B strands. Interestingly, type III leakage was virtually non-existent for plasmid-derived gates, but it was observed to be consistently higher for synthetic gates (e.g. Fig. S10a). This reduction of type III leakage is an important advantage of plasmid-derived gates.

Choice of leak model One approach to modeling the observed type I and type II leakage was to include the set of possible malformed Join_AB gates described above, and to explicitly model all of the resulting leak reactions. In the case of the Join_AB circuit, this gave rise to a total of 18 reactions and 22 species, in contrast to the 4 reactions and 14 species modeled initially, requiring in the worst case 10 additional parameters. Not only were these parameters under-constrained, but more importantly the observed behavior could be captured with a much simpler model requiring many fewer parameters. This stems from the fact that type I and type II leaks all complete very rapidly, resulting in a small amount of initial fluorescence that leveled off shortly after the beginning of the experiment. Thus, they can be approximated by a simplified model which assumes that a fraction of the gates are initially converted to an output signal. Given that both models gave similar fits to the data, we opted for the simpler model.

S7.4 DSD components

The core DSD modules were used to define a collection of DSD components, by instantiating the various domain parameters. In particular, the ForkCBB, ForkBCB and ForkBBC components are all instances of the Fork3 module, with different domain parameters.

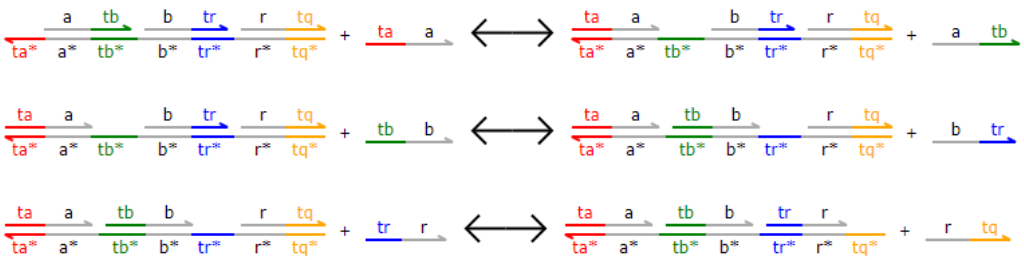
```
def RepT(N,NR) =                          RepJoin(N,NR,r,tq)
def RepC(N,NR) =                          RepFork(N,NR,tc,c)
def JoinAB(NA,NB,NH,NG,NV,Leak) =         Join(NA,NB,NH,NG,NV,Leak,ta,a,tb,b,tr,r,tq)
def ForkC(NT,NH,NI,NG,Leak,NR) =          ( Fork(NT,NH,NI,NG,Leak,i,tc,c,tr,r,tq)
                                          | RepC(0.0,NR) )
def ForkBC(NT,NHB,NHC,NHI,NG,Leak,NR) =   ( Fork2(NT,NHB,NHC,NHI,NG,Leak,i,tc,c,tb,b,tr,r,tq)
                                          | RepC(0.0,NR) )
def ForkCBB(NT,NH1,NH2,NH3,NHI,NG,Leak,NR) = ( Fork3(NT,NH1,NH2,NH3,NHI,NG,Leak,i,tb,b,tb,b,tc,c,tr,r,tq)
                                          | RepC(NG*Leak,NR) )
def ForkBBC(NT,NH1,NH2,NH3,NHI,NG,Leak,NR) = ( Fork3(NT,NH1,NH2,NH3,NHI,NG,Leak,i,tc,c,tb,b,tb,b,tr,r,tq)
                                          | RepC(NG*Leak,NR) )
def ForkBCB(NT,NH1,NH2,NH3,NHI,NG,Leak,NR) = ( Fork3(NT,NH1,NH2,NH3,NHI,NG,Leak,i,tb,b,tc,c,tb,b,tr,r,tq)
                                          | RepC(NG*Leak,NR) )
```

A summary of the strand displacement reactions for these DSD components is provided in Fig. S14 and Fig. S15.

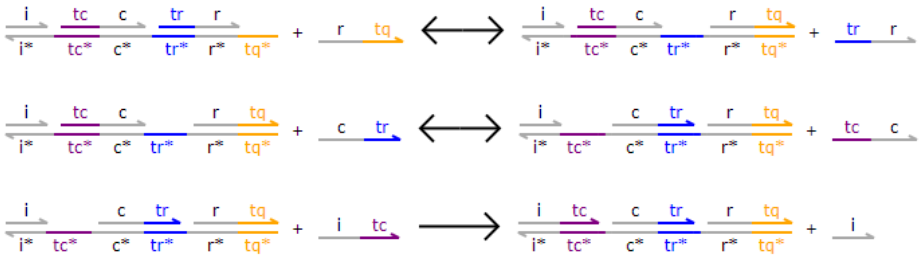
Rep_T

Rep_C

Join_{AB}



Fork_C



Fork_{BC}

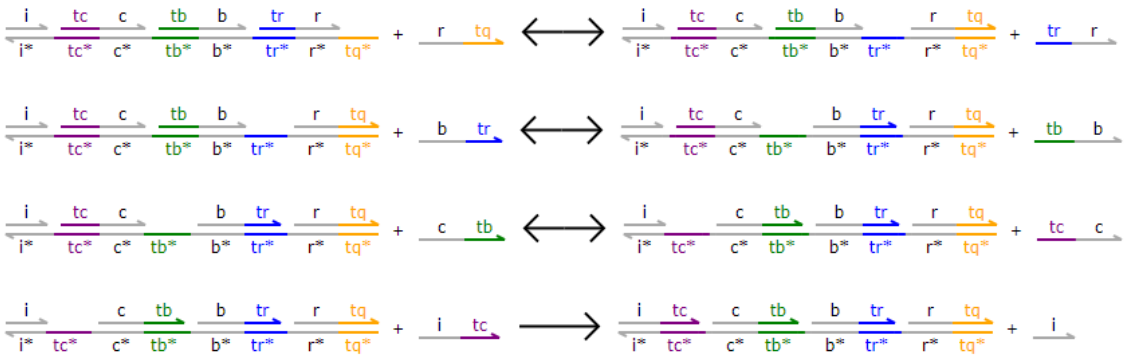
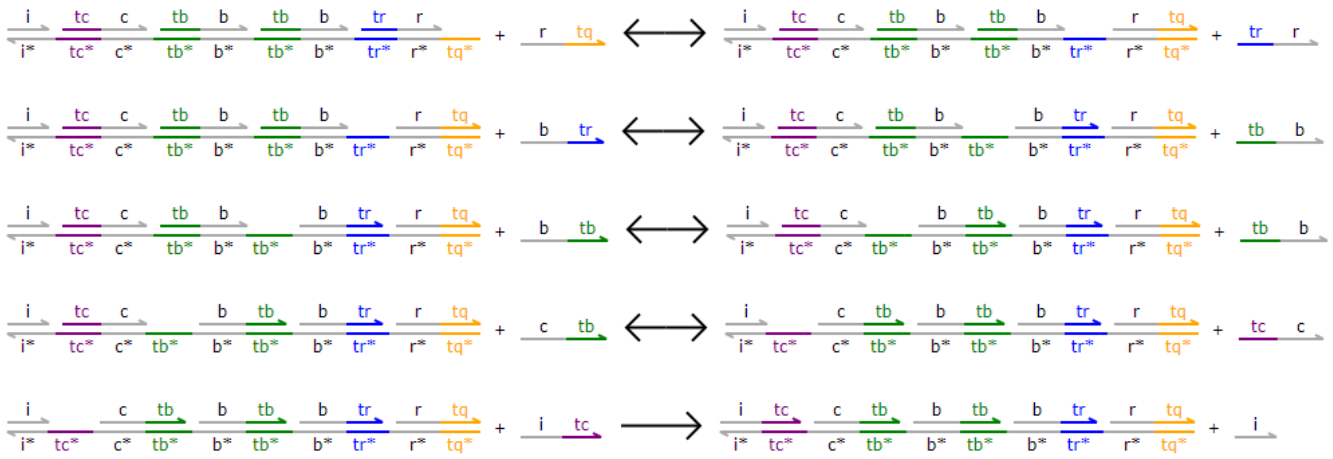
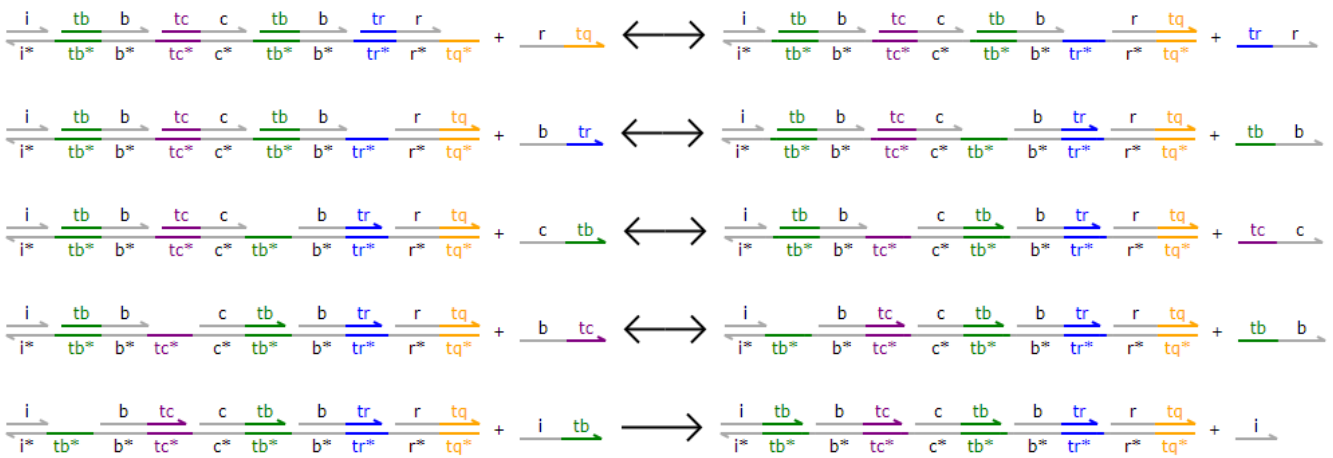


Figure S14: DNA strand displacement reactions for the DSD components Rep_T, Rep_C, Join_{AB}, Fork_C and Fork_{BC}.

Fork$_{BBC}$



Fork$_{BCB}$



Fork$_{CBB}$

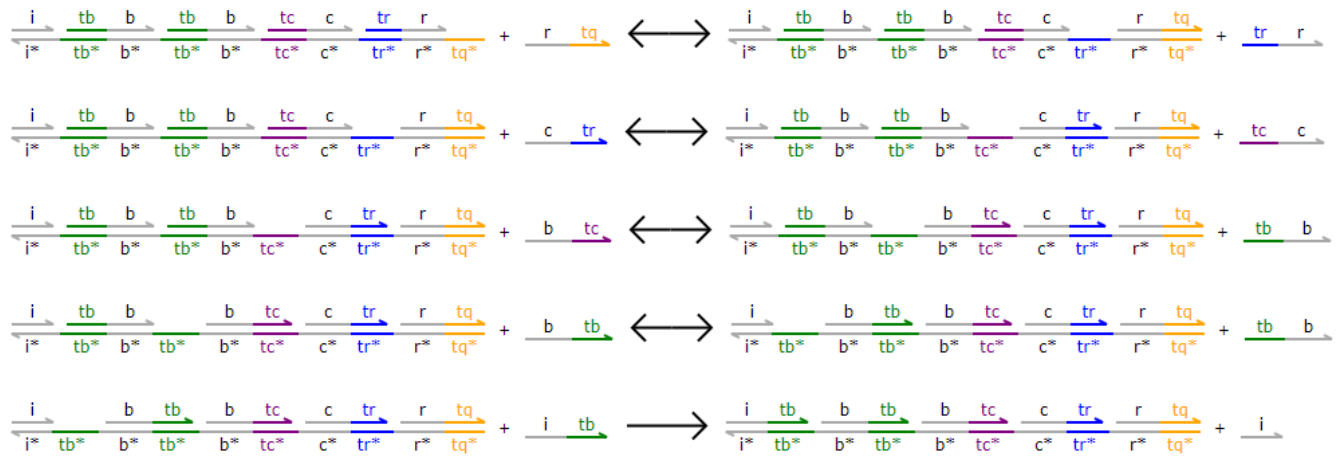


Figure S15: DNA strand displacement reactions for the DSD components Fork$_{BBC}$, Fork$_{BCB}$ and Fork$_{CBB}$.

S7.5 Join, Fork and elementary reaction circuits

In order to simulate the various Join, Fork and elementary reaction circuits that were tested experimentally, a separate circuit was defined for each experiment by combining the components defined in S7.4. Each experiment was assumed to correspond to a given circuit, with a chosen set of strands supplied at varying initial concentrations.

The parameters N, NA and NB represent the variable initial concentrations of the strands that were used to trigger the circuit, while the parameter NV represents the variable initial concentration of reverse strands that were used to slow down the circuit progression. Since experimental measurements indicated a net loss of input strands, this loss was modeled via a fraction of 'bad' input strands, assumed to be non-functional. Leak and Bad parameters were assigned separately for each experiment. Circuits for the experiments presented in Figures S17, S18 and S19 are defined as follows, together with their corresponding parameters:

```
directive parameters [
X = 50.0; NG = 75.0; NH = 100.0; NR = 150.0;                (* Default *)
Xv = 40.0; NGv = 120.0; NHv = 120.0; NRv = 120.0;           (* Reversibles *)
Xh = 10.0; NGh = 15.0; NHh = 20.0; NRh = 30.0;              (* Halves *)
Xi = 10.0; NGi = 30.0; NHi = 30.0; NRi = 30.0; NMi = 100.0; (* Intermediates *)
Xr = 1.0; NRr = 3.0;                                        (* Reporters *)
Xm = 40.0; NGm = 80.0; NHm = 80.0; NRm = 120.0]             (* Consensus *)

def RepT_AddT(N) =     RepT(N*(1.0-badRepT),NRr)
def RepC_AddC(N) =     RepC(N*(1.0-badRepC),NRr)
def JoinAB_AddA(N) =   ( JoinAB(Xi*N*(1.0-badJoinAB_A),NHi,NHi,NGi,0.0,leakJoinAB_A) | RepT(0.0,NRi) )
def JoinAB_AddB(N) =   ( JoinAB(NMi,Xi*N*(1.0-badJoinAB_B),NHi,NGi,0.0,leakJoinAB_B) | RepT(0.0,NRi) )
def JoinAB_AddH(N) =   ( JoinAB(NMi,NMi,Xi*N*(1.0-badJoinAB_H),NGi,0.0,leakJoinAB_H) | RepT(0.0,NRi) )
def ForkC_AddT(N) =    ForkC(Xi*N*(1.0-badForkC_T),NHi,NHi,NGi,leakForkC_T,NRi)
def ForkC_AddH(N) =    ForkC(NMi,Xi*N*(1.0-badForkC_H),NHi,NGi,leakForkC_H,NRi)
def ForkBC_AddT(N) =   ForkBC(Xi*N*(1.0-badForkBC_T ),NHi,NHi,NHi,NGi,leakForkBC_T,NRi)
def ForkBC_AddHB(N) =  ForkBC(NMi,Xi*N*(1.0-badForkBC_HB),NHi,NHi,NGi,leakForkBC_HB,NRi)
def ForkBC_AddHC(N) =  ForkBC(NMi,NMi,Xi*N*(1.0-badForkBC_HC),NHi,NGi,leakForkBC_HC,NRi)
def ForkCBB_AddT(N) =  ForkCBB(Xi*N*(1.0-badForkCBB_T),NHi,NHi,NHi,NHi,NGi,leakForkCBB_T,NRi)
def ForkCBB_AddHC(N) = ForkCBB(NMi,Xi*N*(1.0-badForkCBB_HC),NHi,NHi,NHi,NGi,leakForkCBB_HC,NRi)
def Non(N) =           ( JoinAB(X*N*(1.0-badNon),NH,NH,NG,0.0,leakNon_J)
                       | ForkC(0.0,NH,NH,NG,leakNon_F,NR) )
def Non_Mix(NA,NB) =   ( JoinAB(Xv*NA*(1.0-badMix_A),Xv*NB*(1.0-badMix_B),NHv,NGv,NHv,leakMix_J)
                       | ForkC(0.0,NHv,NHv,NGv,leakMix_F,NRv) )
def Non_Rev(N,NV) =    ( JoinAB(Xv*N*(1.0-badRev_A),Xv*N*(1.0-badRev_B),NHv,NGv,NV,leakRev_J)
                       | ForkC(0.0,NHv,NHv,NGv,leakRev_F,NRv) )
def Cat(N) =           ( JoinAB(X*0.5*(1.0-badCat_A),X*N*(1.0-badCat_B),NH,NG,0.0,leakCat_J)
                       | ForkBC(0.0,NH,NH,NH,NG,leakCat_F,NR) )
def AutoBBC(N) =       ( JoinAB(X*(1.0-badBBC_A),X*N*(1.0-badBBC_B),NH,NG,0.0,leakBBC_J)
                       | ForkBBC(0.0,NH,NH,NH,NH,NG,leakBBC_F,NR) )
def AutoBCB(N) =       ( JoinAB(X*(1.0-badBCB_A),X*N*(1.0-badBCB_B),NH,NG,0.0,leakBCB_J)
                       | ForkBCB(0.0,NH,NH,NH,NH,NG,leakBCB_F,NR) )
def AutoCBB(N) =       ( JoinAB(X*(1.0-badCBB_A),X*N*(1.0-badCBB_B),NH,NG,0.0,leakCBB_J)
                       | ForkCBB(0.0,NH,NH,NH,NH,NG,leakCBB_F,NR) )
```

As an example, consider the DSD program for the experimentally tested JoinAB_AddA circuit, in which the initial concentration of the Input strand A is varied:

```
def JoinAB_AddA(N) = ( JoinAB(Xi*N*(1.0-badJoinAB_A),NHi,NHi,NGi,0.0,leakJoinAB_A) | RepT(0.0,NRi))
```

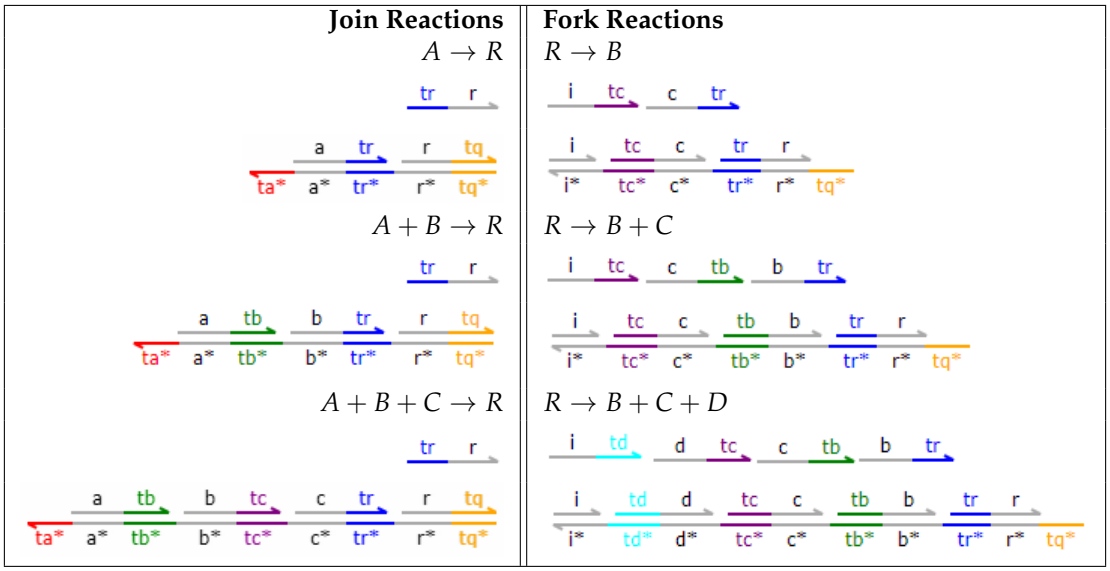
The circuit consists of the JoinAB and RepT components and is parametrized by N, which denotes the proportion of Input strands that are initially present in the system. In the experimental setup N was given the values (0, 0.2, 0.4, 0.6), with the unit concentration Xi = 10 nM, the concentration of helper strands NHi = 3X, the concentration of functional gates NGi = 3Xi, the concentration of reverse strands set to 0.0, and the concentration of reporter gates NRi = 3Xi. The proportions of bad strands (badJoinAB_A) and leaked strands (leakJoinAB_A) were fitted to the experiment.

S7.6 A methodology for implementing arbitrary CRNs

We summarize our methodology for implementing arbitrary chemical reaction networks in DNA. Each formal species is implemented as a single DNA strand consisting of two domains: a short toehold domain at the 5′ end, followed by a long recognition domain at the 3′ end (5′-toehold-recognition-3′). Each formal reaction $X_1 + ... +$

$X_N \rightarrow Y_1 + ... + Y_M$ is implemented as two separate reactions, a join reaction $X_1 + ... + X_N \rightarrow R$ and a fork reaction $R \rightarrow Y_1 + ... + Y_M$. The reason for this separation into join and fork reactions stems from the need to ensure that both reactant and product species have the same orientation. Specifically, the intermediate output of the join reaction R has an inverted orientation with the toehold at the 3′ end, and needs to be converted to a species with the toehold at the 5′ end. In spite of the additional cost, one advantage of this separation is that the implementations of join and fork reactions can potentially be re-used in multiple circuits. For example, consider the non-catalytic, catalytic and autocatalytic reactions $A + B \rightarrow C$, $A + B \rightarrow B + C$ and $A + B \rightarrow 2B + C$, respectively. All three reactions were implemented using the same join reaction $A + B \rightarrow R$, with three different fork reactions $R \rightarrow C$, $R \rightarrow B + C$ and $R \rightarrow 2B + C$, respectively.

In general, each join reaction $X_1 + ... + X_N \rightarrow R$ is implemented using two species: a nicked double-stranded gate that joins all the reactants $X_1, ..., X_N$, together with a single helper strand that produces the intermediate output R. Each fork reaction $R \rightarrow Y_1 + ... + Y_M$ is implemented using $M + 2$ species: a nicked double stranded gate containing the products $Y_1, ..., Y_M$, M helper strands for displacing each of the products, and an additional helper strand at the end to make the reaction irreversible. Thus, each reaction $X_1 + ... + X_N \rightarrow Y_1 + ... + Y_M$ with N reactants and M products is implemented using $M + 4$ species consisting of 2 gates and $M + 2$ strands.



We summarize the number of DNA species needed to implement the reactions considered in this paper:

Reaction	Gates	Strands	Total
$A + B \rightarrow C$	2	3	5
$A + B \rightarrow B + C$	2	4	6
$A + B \rightarrow 2B + C$	2	5	7
$X + Y \rightarrow 2B + PB$	2	5	7
$B + X \rightarrow 2X + PX$	2	5	7
$B + Y \rightarrow 2Y + PY$	2	5	7

A circuit consisting of a set of reactions together with a set of initial formal species is implemented by including the single DNA strands corresponding to each of the formal species, together with a reporter gate for each species that needs to be observed. For example, the consensus network is implemented using 21 species, 7 for each of the 3 reactions of the network, together with 2 initial species and 3 reporters. For comparison, an oscillator can be realized as a relatively simple reaction network such as the predator prey model $X + Y \rightarrow 2Y$, $X \rightarrow 2X$, $Y \rightarrow 0$, where the last reaction is a unimolecular degradation reaction consisting of only a single gate with no helpers (see [11]). Implementing this particular reaction network would therefore require 14 species, roughly two thirds the number required for the three-reaction consensus network. If fractional stoichiometry in a product is required, this can be achieved by mixing fork gates that release different numbers of products. For example, if 75% of the fork gates can release exactly one copy of the output signal while the other 25% cannot release a functional output, the reaction will, on average, turn 1x of a reactant into ¾ x of product.

One important point to note is that, if we only wish to implement non-composable reactions individually, it is possible to use fewer species. For example, the reaction $A + B \rightarrow C$ could be implemented using a single join reaction $A + B \rightarrow R$, without the fork reaction $R \rightarrow C$. But the R strand will have an inverted orientation to the A and B strands, and therefore cannot be used as an input to any subsequent reactions. In general, we believe that the additional fork reactions and accompanying auxiliary strands are a price worth paying for standardized components that can be robustly and modularly composed into functional systems.

To summarize, in our design we enforce that all formal species have the exact same domain structure and, except for the domains constrained by the nicking enzyme binding site, have independent sequence. Without these constraints, we can implement an individual bimolecular reaction with fewer strands, but we would lose the ability to compose reactions into arbitrary CRNs.

For comparison, the reaction mechanism in [11] is equally general but requires fewer strands than the mechanism used for this manuscript. Similar to our scheme, a reaction $X_1 + .. + X_N \rightarrow Y_1 + ... + Y_M$ is implemented as two separate join and fork reactions. Although each join and fork requires only a single gate, these gates have a much more complex structure including overhanging strands, where the combined lengths of the overhangs is roughly equivalent to the combined lengths of the $M + 2$ helper strands used in our scheme. Each formal species also has a more complex composition, consisting of 4 functionally distinct domains rather than 2. Furthermore, the design in [11] is not compatible with plasmid encoding. We also note that the DNA AND logic gate in [9] uses more strands and a more complex reaction mechanism than the DNA AND gate by [10] but is more robust and more easily composed into large circuits. Again, standardization may be somewhat costly at the level of components, but enables reliable composition. In the longer term, we hope that approaches such as the one developed here will make it possible to shift the focus of engineering from the DNA level to the module or systems level, effectively abstracting away the strand-level complexity.

S7.7 Consensus Network

S7.7.1 Reporter strategy

In the consensus network, we cannot directly and continuously measure the values of the DNA strands corresponding to formal species X, Y, and B. Our fluorescent reporters act as irreversible sinks for the strands they detect and measurement of X, Y and B would thus interfere with the progress of the reaction. Instead, we are measuring strands PX, PY, and PB that are released from the same fork gates as X, Y and B and thus report how rapidly these gates are used up. Then, we calculate the values of X, Y and B using the conservation equations shown in Fig. 5b of the main text. Note that these calculations can result in an apparent >100% production yield in some cases, as a result of small errors in determining the amounts of PX, PY, and PB. Errors are due to multiple sources: (1) gate to gate variations in the release of the reporter strands and (2) errors in the calibration using fluorescent reporters and (3) errors in determining the exact concentrations of DNA species.

S7.7.2 DSD components

We used a modular approach to model the components of the consensus network, by instantiating the modules defined in S7.1. The network consists of a combination of Reporter, Join and Fork components, defined as follows. The join reporter circuits were parametrized by the initial concentration NR of reporter gates.

```
RepJoinBX(NR) = RepJoin(0.0,NR,rbx,u1)
RepJoinBY(NR) = RepJoin(0.0,NR,rby,u1)
RepJoinXY(NR) = RepJoin(0.0,NR,rxy,u1)
```

The join circuits were parametrized by the initial concentrations of inputs and helpers, together with the concentration of leaked output strands.

```
JoinBX(NB,NX,NH,NG,Leak) = Join(NB,NX,NH,NG,0.0,Leak,t,b,t,x,t,rbx,u1)
JoinBY(NB,NY,NH,NG,Leak) = Join(NB,NY,NH,NG,0.0,Leak,t,b,t,y,t,rby,u1)
JoinXY(NX,NY,NH,NG,Leak) = Join(NX,NY,NH,NG,0.0,Leak,t,x,t,y,t,rxy,u1)
```

The fork circuits were parametrized by the initial concentration of input and helper strands, together with the concentration of leaked output strands and reporter gates.

```
Fork2B(NR,NH1,NH2,NH3,NH4,NG,Leak,NR) =   ( Fork3(NR,NH1,NH2,NH3,NH4,NG,Leak,ig,t,b,u3,pb,t,b,u2,rxy,u1)
```

```
                                        | RepFork(NG*Leak,NR,u3,pb) )
ForkXY2B(NR,NH1,NH2,NH3,NH4,NG,Leak,NR) = ( Fork3(NR,NH1,NH2,NH3,NH4,NG,Leak,ig,t,b,u3,pb,t,b,u2,rxy,u1)
                                        | RepFork(NG*Leak,NR,t,b) )
Fork2X(NR,NH1,NH2,NH3,NH4,NG,Leak,NR) =   ( Fork3(NR,NH1,NH2,NH3,NH4,NG,Leak,ig,t,x,u3,px,t,x,u2,rbx,u1)
                                        | RepFork(NG*Leak,NR,u3,px) )
Fork2Y(NR,NH1,NH2,NH3,NH4,NG,Leak,NR) =   ( Fork3(NR,NH1,NH2,NH3,NH4,NG,Leak,ig,t,y,u3,py,t,y,u2,rby,u1)
                                        | RepFork(NG*Leak,NR,u3,py) )
```

S7.7.3 DSD circuits

In order to simulate the various circuits that were tested experimentally, a separate circuit was defined for each experiment by combining the above components as follows.

```
def JoinBX_AddB(N) = ( JoinBX(N*Xi*(1.0-badJoinBX_B),NMi,NMi,NGi,leakJoinBX_B) | RepJoinBX(NRi) )
def JoinBX_AddX(N) = ( JoinBX(NMi,N*Xi*(1.0-badJoinBX_X),NMi,NGi,leakJoinBX_X) | RepJoinBX(NRi) )
def JoinBX_AddH(N) = ( JoinBX(NMi,NMi,N*Xi*(1.0-badJoinBX_H),NGi,leakJoinBX_H) | RepJoinBX(NRi) )
def JoinBY_AddB(N) = ( JoinBY(N*Xi*(1.0-badJoinBY_B),NMi,NMi,NGi,leakJoinBY_B) | RepJoinBY(NRi) )
def JoinBY_AddY(N) = ( JoinBY(NMi,N*Xi*(1.0-badJoinBY_Y),NMi,NGi,leakJoinBY_Y) | RepJoinBY(NRi) )
def JoinBY_AddH(N) = ( JoinBY(NMi,NMi,N*Xi*(1.0-badJoinBY_H),NGi,leakJoinBY_H) | RepJoinBY(NRi) )
def JoinXY_AddX(N) = ( JoinXY(N*Xi*(1.0-badJoinXY_X),NMi,NMi,NGi,leakJoinXY_X) | RepJoinXY(NRi) )
def JoinXY_AddY(N) = ( JoinXY(NMi,N*Xi*(1.0-badJoinXY_Y),NMi,NGi,leakJoinXY_Y) | RepJoinXY(NRi) )
def JoinXY_AddH(N) = ( JoinXY(NMi,NMi,N*Xi*(1.0-badJoinXY_H),NGi,leakJoinXY_H) | RepJoinXY(NRi) )
def Fork2B_AddR(N) =          Fork2B(N*Xi*(1.0-badFork2B_R ),NMi,NMi,NMi,NMi,NGi,leakFork2B_R,NRi)
def Fork2B_AddH1(N) =         Fork2B(NMi,N*Xi*(1.0-badFork2B_H1),NMi,NMi,NMi,NGi,leakFork2B_H1,NRi)
def Fork2B_AddH2(N) =         Fork2B(NMi,NMi,N*Xi*(1.0-badFork2B_H2),NMi,NMi,NGi,leakFork2B_H2,NRi)
def Fork2X_AddR(N) =          Fork2X(N*Xi*(1.0-badFork2X_R ),NMi,NMi,NMi,NMi,NGi,leakFork2X_R,NRi)
def Fork2X_AddH1(N) =         Fork2X(NMi,N*Xi*(1.0-badFork2X_H1),NMi,NMi,NMi,NGi,leakFork2X_H1,NRi)
def Fork2X_AddH2(N) =         Fork2X(NMi,NMi,N*Xi*(1.0-badFork2X_H2),NMi,NMi,NGi,leakFork2X_H2,NRi)
def Fork2Y_AddR(N) =          Fork2Y(N*Xi*(1.0-badFork2Y_R ),NMi,NMi,NMi,NMi,NGi,leakFork2Y_R,NRi)
def Fork2Y_AddH1(N) =         Fork2Y(NMi,N*Xi*(1.0-badFork2Y_H1),NMi,NMi,NMi,NGi,leakFork2Y_H1,NRi)
def Fork2Y_AddH2(N) =         Fork2Y(NMi,NMi,N*Xi*(1.0-badFork2Y_H2),NMi,NMi,NGi,leakFork2Y_H2,NRi)
def XY2B(Nx,Ny,X,NH,NG,NR) = ( JoinXY(X*Nx*(1.0-badXY2B_X),X*Ny*(1.0-badXY2B_Y),NH,NG,leakXY2B_J)
                             | ForkXY2B(0.0,NH,NH,NH,NH,NG,leakXY2B_F,NR) )
def BX2X(N,X,NH,NG,NR) =      ( JoinBX(X*(1.0-badBX2X_B),X*N*(1.0-badBX2X_B),NH,NG,leakBX2X_J)
                             | Fork2X(0.0,NH,NH,NH,NH,NG,leakBX2X_F,NR) )
def BY2Y(N,X,NH,NG,NR) =      ( JoinBY(X*(1.0-badBY2Y_B),X*N*(1.0-badBY2Y_B),NH,NG,leakBY2Y_J)
                             | Fork2Y(0.0,NH,NH,NH,NH,NG,leakBY2Y_F,NR) )
def Consensus(Nx,Ny) =        ( XY2B(Nx,Ny,Xm,NHm,NGm,NRm)
                             | BX2X(0.0,0.0,NHm,NGm,NRm)
                             | BY2Y(0.0,0.0,NHm*1.2,NGm*1.2,NRm)
                             | NHm * 0.8 * <ig t^> )
```

S7.8 Numerical simulation of circuit induction

To compare the dynamical behavior of the model with the experimental data, we sought to perform simulations that recapitulated the experimental conditions as closely as possible. While the experiments differed in their quantitative details (concentrations of initial gates and strands, concentrations of inducing strands, etc.), they all followed a standardized experimental protocol. The protocol begins by pre-mixing the initial DNA gates and strands at time T_0. Following pre-mixing, the initial gates and strands are loaded into laboratory equipment and their fluorescence output is measured. We refer to the time at which the first fluorescence measurement is recorded as time 0, meaning that T_0 is a negative value. For example, if the initial gates and strands are mixed 15 min before the first fluorescence measurement, we have $T_0 = -0.25$ h. At a later time T_1, inducing input or catalyst strands are added to the solution. Fluorescence measurements are collected from time 0 until completion of the experiment, providing observations $(t, y) = (t_1, y_1), (t_2, y_2), \ldots, (t_N, y_N)$, where $t_1 = 0$ and y_k is the fluorescence measurement at time t_k. Experiment-specific details are explicitly stated in the corresponding figure legends.

We assume that spatially homogeneous deterministic dynamics are adequate to describe the underlying dynamics of the DNA circuits. Potential spatial heterogeneity is removed by actively pre-mixing the initial gates and strands prior to the first fluorescence measurement. Stochasticity is unlikely to be of major importance, since the copy number of specific strands is large (1 nM in a 600 μL volume corresponds to 3.6×10^{11} molecules). Model

simulation is efficiently achieved through numerical integration of ordinary differential equations, which are derived from the system reactions assuming mass action kinetics. We made use of automatic code generation in DSD to create the differential equations in a form suitable for the Microsoft Research Solvers library [1], which is used to perform the numerical integration.

The experiments were simulated in two halves: i) a pre-mixing simulation from time T_0 to time T_1, starting from the vector of initial conditions corresponding to the concentrations of initial gates and strands, then ii) induction by input and/or catalyst strands from time T_1 to time t_N, where the initial condition is the vector sum of the final vector of concentrations from i) and the relevant quantity of input/catalyst strands added at time T_1.

S7.9 Model selection, behavior and parameterization

S7.9.1 Bayesian parameter inference

To assess the plausibility of a model of a specific circuit, we first determine optimal parameter values using probabilistic inference techniques. In particular, we seek to approximate the probability distribution of the parameters, given a model hypothesis and some observation data. i.e. we attempt to approximate the posterior density $Pr(\theta|H,D)$, where θ is the vector of parameters to be inferred, H is the model hypothesis, and D is the set of experimental data used for inference. The posterior distribution is related to an evidence or likelihood function $Pr(D|\theta,H)$ according to Bayes' rule:

$$Pr(\theta|H,D) = \frac{Pr(D|\theta,H)Pr(\theta|H)}{Pr(D)}$$

We obtained approximations of the posterior distributions using the Filzbach software, available from the author's website [2], which uses a Metropolis-Hastings (MH) Markov Chain Monte Carlo (MCMC) sampling routine. MCMC is a stochastic search strategy that forms a Markov chain of proposal parameter vectors, moving to new proposal vectors based on the ratio of the likelihoods of the proposal and previous points. By biasing the stochastic search in parameter regions of high probability mass, we converge on the true joint posterior distribution of the parameters more efficiently than would be possible with a purely random search.

To define the likelihood of a parameter vector, we assumed that the experimental measurements were noisy samples drawn independently from Gaussian distributions centered on the model predictions of the fluorescence. Therefore, in the optimal case that the model precisely describes the underlying biological behavior, deviations of the measurements result purely from experimental error. Consequently, a data point y_i is distributed as $y_k \sim N(x_k, \sigma^2)$, where x_k is the model prediction at $t = t_k$ and σ^2 is the variance of experimental error. We assume that the variance of experimental error is proportional to the measured fluorescence signal (i.e. $\sigma = \alpha\sqrt{y_k}$ for some α), reflecting the Poisson distribution of fluorescence readings [12]. Therefore, the likelihood function is formed as the product of the probabilities of each data-point

$$L(\theta) = Pr(D|\theta) = \prod_{k=1}^{N} \frac{1}{\alpha\sqrt{2\pi y_k}} \cdot e^{-\frac{(y_k - x_k)^2}{\alpha^2 y_k}}$$

S7.9.2 Parameter types

Parameters were classified using one of the following types:

- *Kinetic* parameters or reaction rates. The kinetic models of each circuit contain a number of reactions, the rates of which are all unknown. To parameterize these reactions, we initially supposed that the overall strand displacement rates could be assigned purely based on the toehold sequence (and not the recognition domain). However, we quickly found that this model was not sufficient to describe the large quantity of kinetic data, and found that the sequence of the recognition domain could also influence the overall strand displacement rate.

- *Leak* parameters. The leak parameters appear in the simulation as a fraction of gates that have spontaneously leaked. Therefore, the initial condition for the corresponding gate is set to $N_G \times (1 - leak)$ and the initial condition for the output of the gate is set to $N_G \times leak$, where N_G is the intended concentration of gates, and *leak* is the leak parameter. Circuits composed of both *Join* and *Fork* gates therefore have two leak parameters.

- *Bad* parameters. The bad parameters are applied only to input strands that are not in excess, which we define as being $> 1x$.

- *Noise* parameters. A distinct noise parameter was assigned to each dataset, and inferred using our parameter estimation procedure.
- *Timing* parameters. During each experiment, the time at which inducing strands were added to the cuvette varied, which could lead to slightly variable kinetic behaviors. Therefore, the time at which the gates and helpers were mixed (T_0) and the time at which the inducing strands were added (T_1) were considered as parameters to be inferred.

S7.9.3 Determining the relationship between strand sequence and rate of strand displacement reaction

Models of the non-catalytic, catalytic and autocatalytic circuits, and their constituent parts were analyzed for their ability to reproduce the experimental data, using our parameter inference procedure. Data from 23 separate experiments each containing 3–10 different traces were used simultaneously, to ensure that the inferred parameters were generally applicable. We found that the kinetic behaviors could not be described by a parameterization in which strand displacement depends only on the sequence of the toehold domain. Instead, we assigned distinct rates to each strand, initially giving rise to 13 kinetic rate parameters. We then extended this list by allowing a distinct rate of binding to reporter gates (k^R_{rtq} and k^R_{tcc}). From here on, we refer to this as the "Fix all" parameterization.

When fitting our default parameterization, we found that it wasn't possible to capture all measured kinetic behaviours. In particular, experiments where auxiliary strands were added to single gates that had already received their inputs were problematic (Figures S17, S18). We supposed that there were conflicts in the usage of particular strands between different circuits/experiments. For example, the <b tr> strand is displaced by <tb b> in the Join_{AB} gate, while it is an auxiliary strand in the Fork_{BC} gate. A similar conflict is apparent for <tr r> and <r tq>. To address this, we tried assigning distinct rates for different gates, giving rise to three new hypotheses (*Vary* <r tq>, *Vary* <tr r> and *Vary* <b tr>). Another possibility for rate conflict could derive from the displaced strand having a different sequence. For example, <tc c> strands displace <c tr> strands in Fork_{C} and Fork_{CBB} gates, though displace <c tb> strands in Fork_{BC} gates. A similar observation can be made about <tb b>, giving rise to a further two hypotheses about the source of the conflict (*Vary* <tc c> and *Vary* <tb b>).

We determined which conflict hypothesis was most likely by running multiple MCMC analyses for each, and calculating the Bayesian Information Criterion (BIC) in each case. The BIC is an approximation for the conditional probability of a particular hypothesis, given observations. It is comprised of two penalty terms, one which penalizes deviations from the observations, and another which penalizes additional parameters to be inferred. As such, lower BIC scores are desirable, which find an optimal compromise between the complexity of the model being used and the dynamical behavior it is capable of describing. We found that the *Vary* <r tq> hypothesis provided the best BIC score, with other *Vary* <x x> hypotheses performing slightly worse than the default *Fix all* hypothesis (Fig. S16).

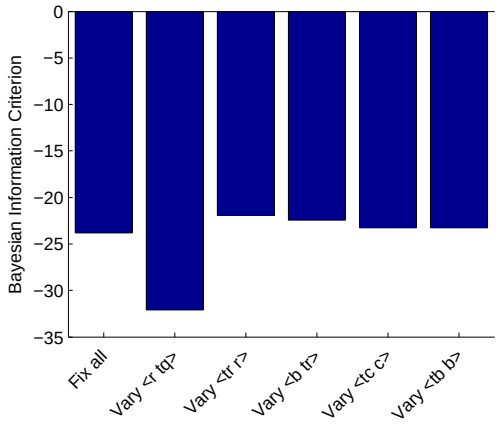


Figure S16: Comparison of model refinements. To determine the best modification to the unique context hypothesis, we implemented and analyzed 5 hypotheses, in which specific strands were assumed to have binding rates that were specific to their binding context. For each hypothesis, 10 independent MCMC chains were computed, and analyzed for the minimum observed BIC score, using the Filzbach software.

S7.9.4 Kinetics of individual Join and Fork gates

To obtain estimates for the rates of specific strand displacement reactions, we measured the kinetic behavior of join and fork gates subjected to a variety of treatment regimes. In the ideal case, the amount of translator strand <r tq> produced from the join gate should be exactly equal to the amount of the limiting reactant. We explored the behavior when each reactant was limiting. First, the kinetics across the join gate were observed after pre-mixing gates with input B and auxiliary <tr r>, then adding the input A (Fig. S17a). Next, the kinetics were observed following pre-mixing with A and <tr r> then treating with B (Fig. S17b), and also pre-mixing with A and B then treating with <tr r> (Fig. S17c). To characterize any delay between the release of <r tq> and observation of fluorescence, we also directly measured this step (Fig. S17d). For each experiment, the model produced dynamics that closely followed the observed kinetics. Leak and bad parameters were observable directly from the data, with leaks representing the fluorescent offset with 0x input, and the bads representing the fraction of dysfunctional input. Nevertheless, these parameters were inferred along with the kinetic parameters using our inference strategy, and produce the values expected from inspecting the data.

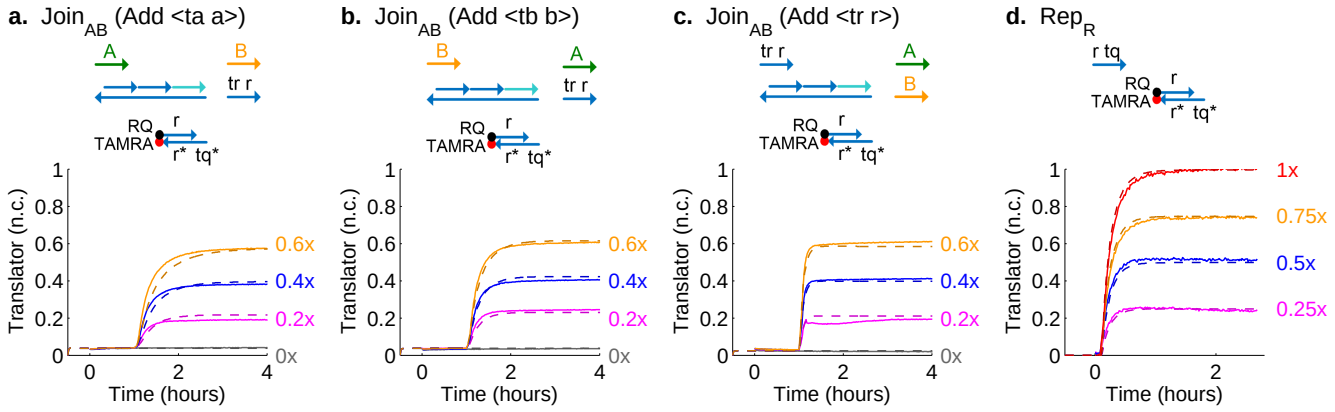


Figure S17: Kinetic behavior of Join gates. Shown is a comparison of model dynamics (dashed lines) with experimental observations (solid lines), for datasets that characterize strand displacement reactions on the Join_{AB} gate. The experiments directly observe the forward reactions (from left to right on the gate), **a**, <ta a> displacing <a tb>, **b**, <tb b> displacing <b tr>, **c**, <tr r> displacing <r tq>, and **d**, <r tq> binding the reporter gate. In a–c, 1x = 10 nM, and we start with 3x gates and reporters. All gates are pre-mixed with 10x of strands upstream and 3x of strands downstream of the target interaction. In d, 1x = 1 nM, and we use 3x reporter gates. In all cases, the model can produce a dynamical behavior that closely reproduces the experimental measurements. The corresponding parameter values can be found in Table S3.

A similar kinetic analysis was applied to the Fork_{C}, Fork_{CBB} and Fork_{BC} gates, along with the corresponding reporter for the output C (<tc c>) (Fig. S18). As for the join gate, measuring the kinetic behaviors at different stages of maturity enabled us to characterize the kinetic, leak and bad parameters associated with specific combinations of input strand and gate.

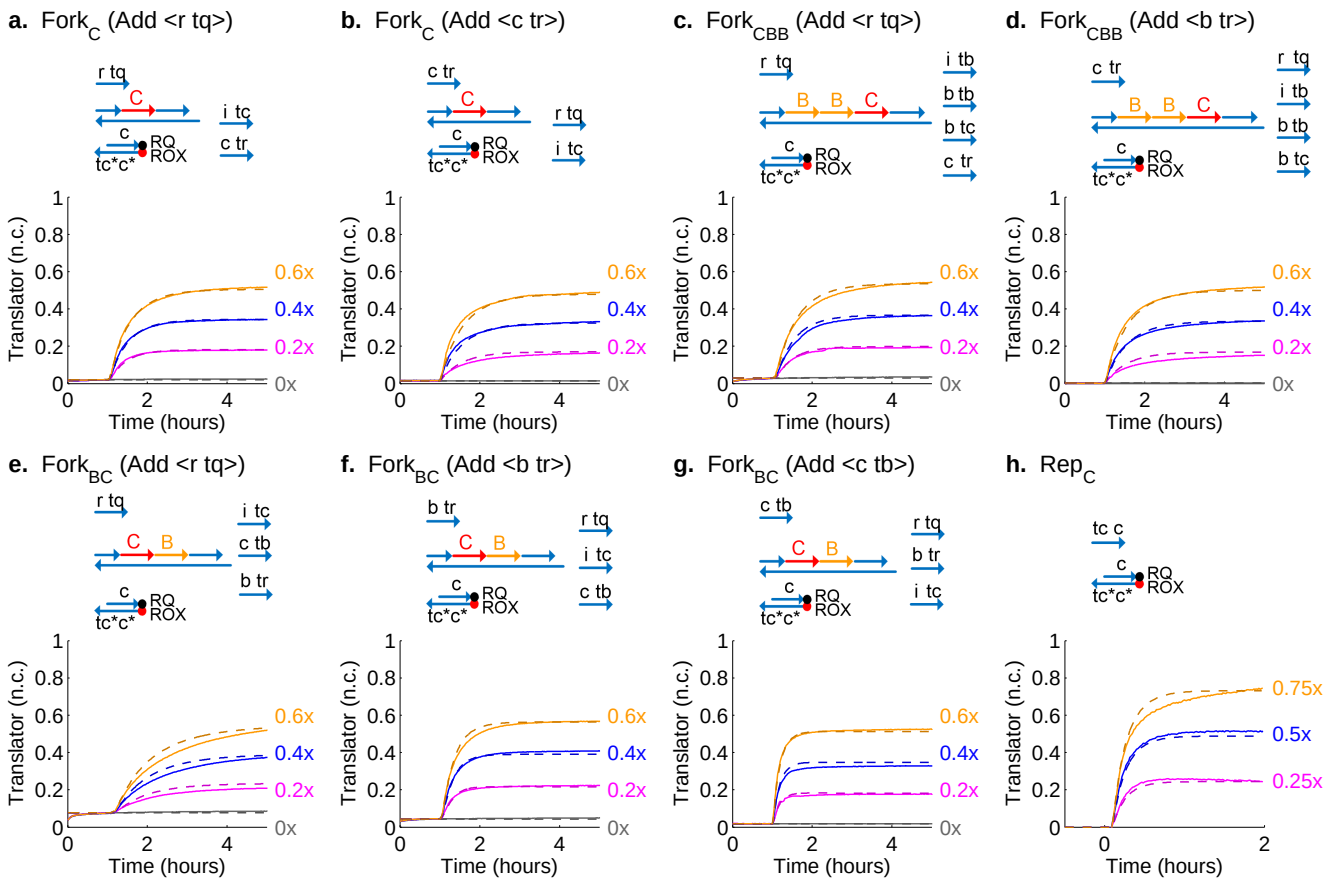


Figure S18: Kinetic behavior of Fork gates. Shown is a comparison of model dynamics (dashed lines) with experimental observations (solid lines), for datasets that characterize strand displacement reactions on individual Fork gates. The experiments directly observe the forward reactions (from right to left on the gate), **a**, <r tq> displacing <tr r> on Fork_{C}, **b**, <c tr> displacing <tc c> on Fork_{C}, **c**, <r tq> displacing <tr r> on Fork_{CBB}, **d**, <c tr> displacing <tc c> on Fork_{CBB}, **e**, <r tq> displacing <tr r> on Fork_{BC}, **f**, <b tr> displacing <tb b> on Fork_{BC}, **g**, <c tb> displacing <tc c> on Fork_{BC}, and **h**, binding the reporter gate. In a–g, 1x = 10 nM, and we start with 3x gates and reporters. All gates are pre-mixed with 10x of strands upstream and 3x of strands downstream of the target interaction. In d, 1x = 1 nM, and we use 3x reporter gates. In all cases, the model can produce a dynamical behavior that closely reproduces the experimental measurements. The corresponding parameter values can be found in Table S3.

S7.9.5 Kinetics of full circuits implementing CRNs

For completeness and to re-iterate that the parameter inference procedure was applied to multiple kinetic data simultaneously, we re-plot the experiments shown in the main text figures. These are the catalytic $A + B \rightarrow B + C$ circuit (Fig. S19a), the non-catalytic $A + B \rightarrow C$ circuit (Fig. S19b), the same non-catalytic circuit with reverse strands <a tb> and <b tr> added in various concentrations (Fig. S19c), and autocatalytic circuits $A + B \rightarrow C + B + B$ (Fig. S19d.i), $A + B \rightarrow B + C + B$ (Fig. S19d.ii) and $A + B \rightarrow B + B + C$ (Fig. S19d.iii).

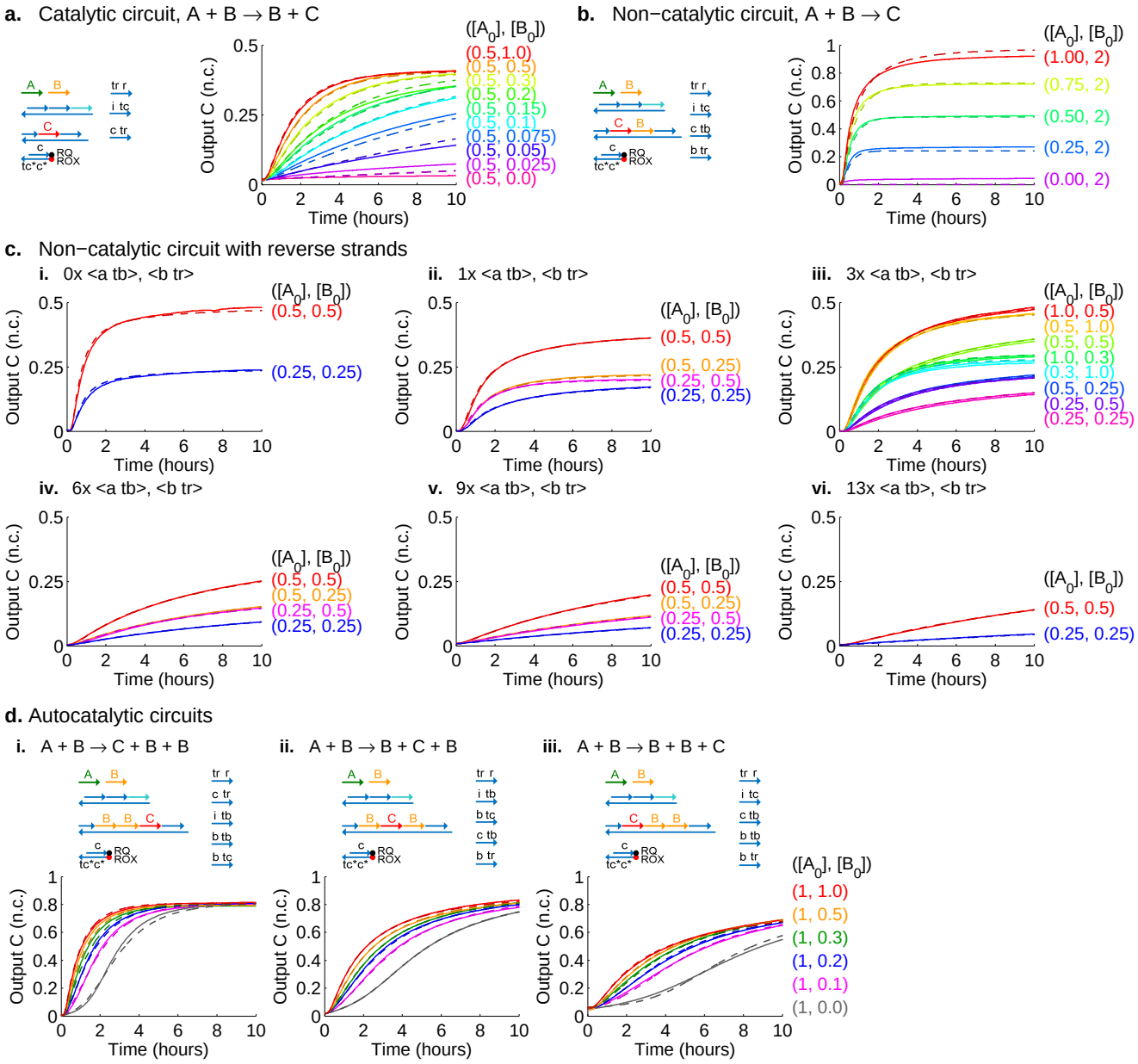


Figure S19: Kinetic behavior of CRN circuits. Shown is a comparison of model dynamics (dashed lines) with experimental observations (solid lines), for **a**, the catalytic $A + B \rightarrow B + C$ circuit, **b**, the non-catalytic $A + B \rightarrow C$ circuit, **c**, the non-catalytic circuit with kinetics modulated by addition of <a tb> and <b tr>, which slow the progress along the Join_{AB} gate, and **d**, autocatalytic circuits. In all cases, the model can produce a dynamical behavior that closely reproduces the experimental measurements. In d, panel (iv) shows a linear fit of the 15% completion time against the logarithm of the relative concentration of signal B (this extends Figure 2d (iii) in the main text). The data and model simulations are repeated from Figures 2b–d and 3 from the main text. The corresponding parameter values can be found in Table S3.

Table S3: Maximum likelihood parameter values for the strand displacement model. The maximum likelihood parameter values were determined using the Filzbach MCMC software. For kinetic parameters, the subscript denotes the strand for which the parameter applies: *taa* is <ta a>, *rtq* is <r tq>, etc. The superscript denotes specific gates that the parameter applies to: *J* is Join_{AB}, *C* is Fork_{C} and Fork_{CBB}, *BC* is Fork_{BC} and *R* is reporter. For leak and bad parameters, the subscript denotes the experiment for which the parameter value applies: $ABCr3$ is the non-catalytic $A + B \rightarrow C$ circuit with $3x$ reversibles, Cat is the catalytic $A + B \rightarrow B + C$ circuit, etc. The subscript distinguishes between multiple parameters in an experiment, $Join/Fork$ for the leaks, and A (<ta a>)/B (<tb b>) for the bads.

Kinetic parameters		**Leak parameters**		**Bad parameters**	
Name	Value ($M^{-1}s^{-1}$)	Name	Value	Name	Value
k_{taa}	39277	$leak^{J}_{ABCr3}$	2.0909×10^{-7}	bad^{A}_{ABCr3}	0.0677
k_{tbb}	121200	$leak^{F}_{ABCr3}$	0.0006	bad^{B}_{ABCr3}	2.6×10^{-5}
k_{trr}	279030	$leak^{J}_{ABCr0}$	0.0013	bad^{A}_{ABCr0}	0.0281
k^{J}_{rtq}	10197	$leak^{F}_{ABCr0}$	0.3475	bad^{B}_{ABCr0}	0.0279
k^{C}_{rtq}	208830	$leak^{J}_{ABCr1}$	0.0003	bad^{A}_{ABCr1}	0.2030
k^{BC}_{rtq}	12971	$leak^{F}_{ABCr1}$	0.3876	bad^{B}_{ABCr1}	0.1162
k_{btr}	158000	$leak^{J}_{ABCr6}$	0.0015	bad^{A}_{ABCr6}	0.1460
k_{ctr}	24785	$leak^{F}_{ABCr6}$	0.4387	bad^{B}_{ABCr6}	0.0002
k_{tcc}	33016	$leak^{J}_{ABCr9}$	0.0031	bad^{A}_{ABCr9}	0.1280
k_{btc}	1445800	$leak^{F}_{ABCr9}$	0.0012	bad^{B}_{ABCr9}	3.88×10^{-5}
k_{ctb}	117420	$leak_{ABC}$	0.0439	bad^{A}_{ABC}	0.0212
k_{atb}	510720	$leak^{J}_{Cat}$	2.34×10^{-5}	bad^{A}_{Cat}	0.1837
k_{btb}	20138	$leak^{F}_{Cat}$	0.0137	bad^{B}_{Cat}	0.4159
k_{itc}	47793	$leak^{J}_{BBC}$	0.0050	bad^{A}_{BBC}	0.1786
k_{itb}	10012	$leak^{F}_{BBC}$	0.0419	bad^{B}_{BBC}	0.4999
k^{R}_{rtq}	731720	$leak^{J}_{BCB}$	0.0192	bad^{A}_{BCB}	0.1521
k^{R}_{tcc}	599460	$leak^{F}_{BCB}$	0.0063	bad^{B}_{BCB}	3.17×10^{-5}
		$leak^{J}_{CBB}$	0.0166	bad^{A}_{CBB}	0.1498
		$leak^{F}_{CBB}$	4.57×10^{-5}	bad^{B}_{CBB}	5.89×10^{-5}
		$leak_{JoinAB-A}$	0.0126	$bad_{JoinAB-A}$	0.1072
		$leak_{JoinAB-B}$	0.0121	$bad_{JoinAB-B}$	0.0372
		$leak_{JoinAB-H}$	0.0077	$bad_{JoinAB-H}$	0.0636
		$leak_{ForkC-R}$	0.0066	$bad_{ForkC-R}$	0.1916
		$leak_{ForkC-H}$	0.0049	$bad_{ForkC-H}$	0.2277
		$leak_{ForkBC-R}$	0.0258	$bad_{ForkBC-R}$	0.2186
		$leak_{ForkBC-H1}$	0.0145	$bad_{ForkBC-H1}$	0.1342
		$leak_{ForkBC-H2}$	0.0058	$bad_{ForkBC-H2}$	0.1755
		$leak_{ForkCBB-R}$	0.0097	$bad_{ForkCBB-R}$	0.1561
		$leak_{ForkCBB-H}$	0.0007	$bad_{ForkCBB-H}$	0.1713
				bad_{RepC}	0.0249
				bad_{RepR}	0.0055

S8 Consensus network

S8.1 Behavior of the ideal consensus network CRN

The ideal consensus network works as a two-species molecular classifier that converges to either all X or all Y depending on which species made up a greater percentage of the input. It works by converting all of the signal initially present in the minority into the signal initially present in the majority. Here we demonstrate the ideal behavior of the consensus network by simulating the ideal CRN shown in Fig. 5:

$$X + Y \xrightarrow{k} 2B$$
$$B + X \xrightarrow{k} 2X$$
$$B + Y \xrightarrow{k} 2Y$$

Fig. S20 qualitatively matches our experimental data (Fig. 5) in the main text: intuitively, the minority and majority signals initially cancel each other producing the buffer signal B which is then converted back to the majority signal.

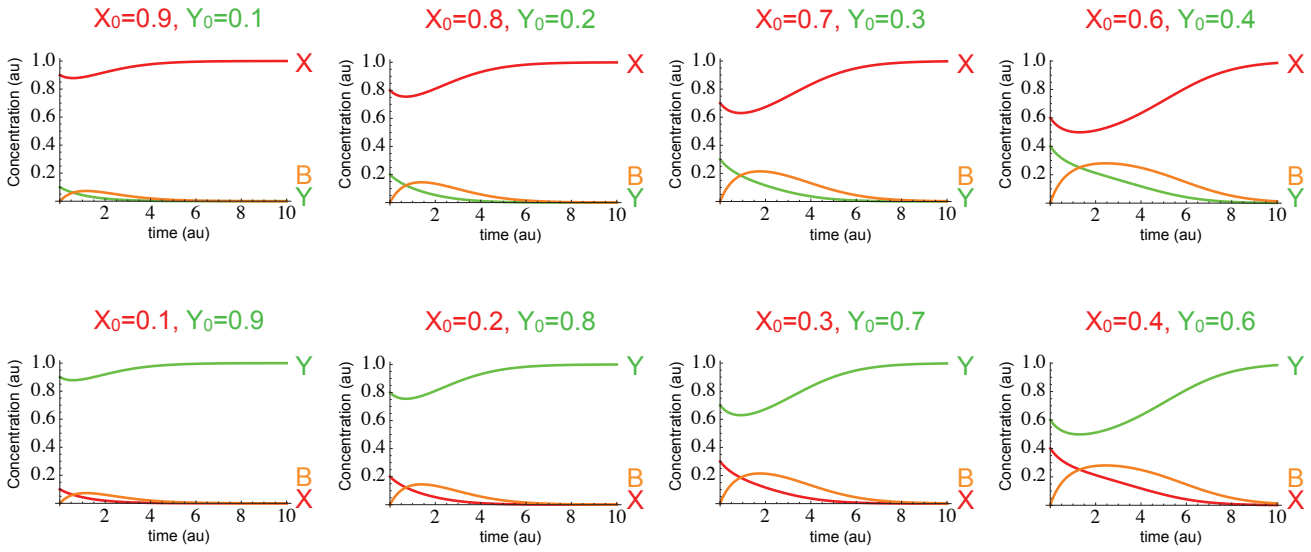


Figure S20: Ideal consensus network behavior. Unitless numerical simulation of the chemical reactions shown above and in Fig.5b, using $k = 1$. The behavior of the system qualitatively matches experimentally obtained data shown in Fig. 5: both the majority and minority species initially drop to form B and then the majority species recovers while the minority continues to decay.

S8.2 Characterization of individual reactions of the consensus network

The consensus network consists of three different chemical reactions: one non-catalytic reaction, and two autocatalytic reactions, and we characterized the kinetics of each individual reaction as shown in Fig. S21. The DNA implementation is shown in panel i, and sequences are detailed in Table S10. In the experiments of $X + Y \rightarrow 2B + PB$, we varied the concentrations of signal X and signal Y. The kinetics of output B is shown in panel ii, and the completion level of output B is shown in panel iii. In an ideal reaction, the output B should be twice as that of the limiting reactant, and the experimental data show that reaction completion levels are very close to a stoichiometrically correct bimolecular reaction. To follow the kinetics of autocatalytic reactions, we read out the signal PX and PY for $B + X \rightarrow 2X + PX$ and $B + Y \rightarrow 2Y + PY$, respectively (Fig. S21b(ii) and Fig. S21c(ii)). Both reactions demonstrated exponential growth in the initial phase and this is verified by the linear fit of the 15% completion time against the logarithm of the relative concentration of the catalyst signals (Fig. S21b(iii) and Fig. S21c(iii)). These two autocatalytic reactions showed different reaction speeds, and thus we adjusted the concentrations of auxiliary strands and gates for $B + Y \rightarrow 2Y + PY$ to balance the rates of the two autocatalytic reactions in the experiments of the consensus network.

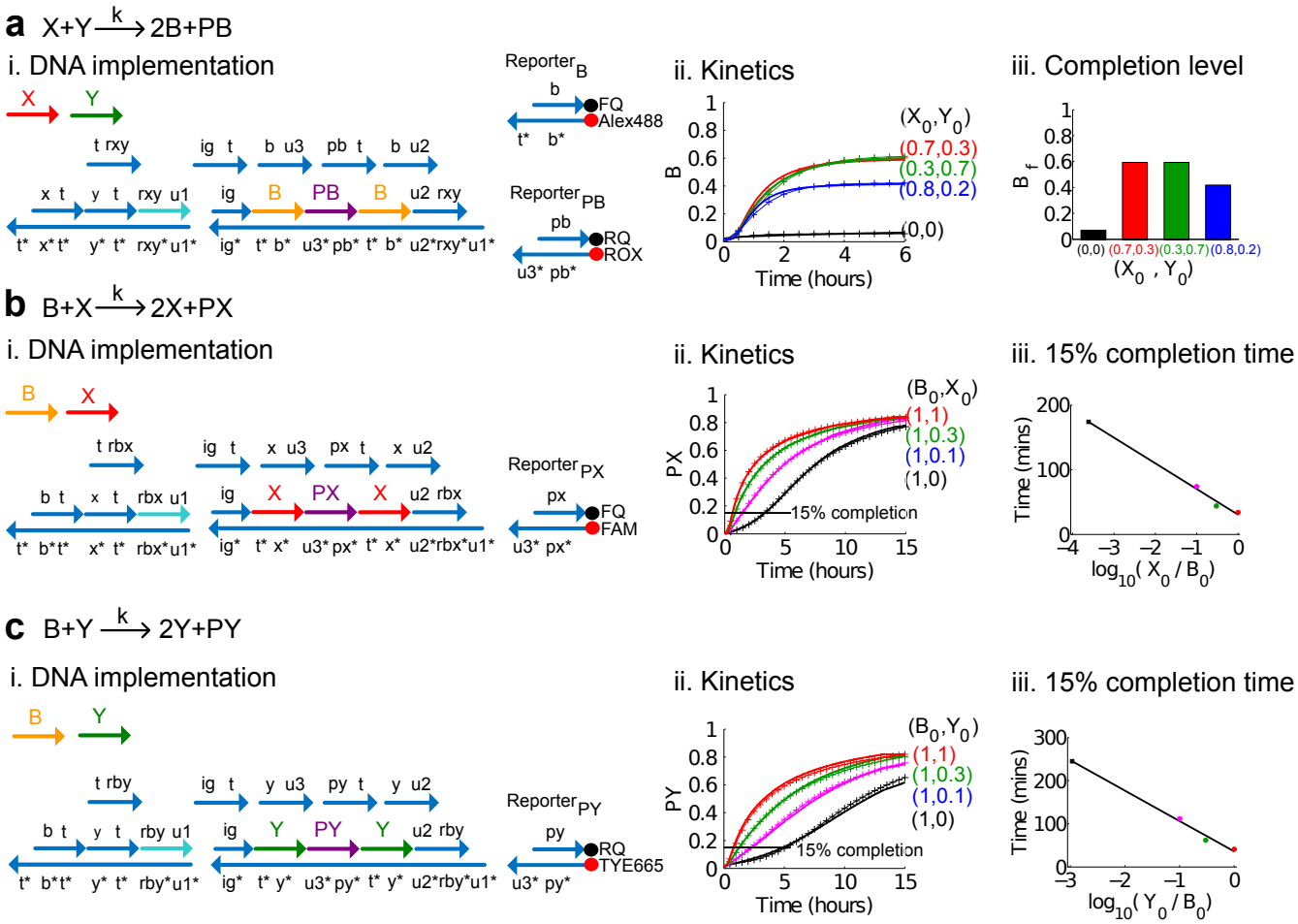


Figure S21: Kinetics data for individual chemical reactions of the consensus network. Panel i in each row shows the signal strands, auxiliary strands, ndsDNA gates and reporter gates used for the corresponding experiments. Experiments were run in 1x TAE/Mg^{++} and performed at 25°C. All auxiliary strands were at 100 nM (2x), and the gates were at 75 nM concentration (1.5x). Best fits of the strand displacement-level model to the data are shown as crossed lines. **a**, $X + Y \rightarrow 2B + PB$. Different amounts of signal X and signal Y were added. The $\text{Reporter}_{\text{B}}$ was used for the characterization of $X + Y \rightarrow 2B + PB$, and the $\text{Reporter}_{\text{PB}}$ was used for the consensus network. Panel ii shows the kinetics of the signal B. Panel iii shows the completion level of signal B. **b**, $B + X \rightarrow 2X + PX$. Signal B was fixed at 50 nM (1x). Different amounts of signal X were added. Panel ii shows the kinetics of the signal PX. Panel iii shows a linear fit of the 15% completion time against the logarithm of the relative concentration of signal PX. **c**, $B + Y \rightarrow 2Y + PY$. Signal B was fixed at 50 nM (1x). Different amounts of signal Y were added. Panel ii shows the kinetics of the output PY. Panel iii shows a linear fit of the 15% completion time against the logarithm of the relative concentration of signal PY.

S8.3 Prediction of consensus network behavior from bimolecular fits of the three reactions

How well can the behavior of the full consensus network (ie Fig. 5c) be explained in terms of the bimolecular models of the three reactions? The separate experiments on the three consensus network reactions were used to fit an effective bimolecular rate constant to each reaction separately. Then a prediction was made by simulating the formal consensus CRN with these three rate constants. We initially found that the prediction was significantly faster than the experimentally observed kinetics. However, as justified in Section S8.4.4, incorporating a phenomenological interference parameter yielded better agreement (Fig. S22). The prediction agrees reasonably well for the four conditions with the greatest difference between the initial amounts of X and Y, and in all cases captures the essential qualitative behavior. However, the detailed strand displacement-level models are more predictive (see Section S8.4.4). This is likely, at least in part, due to the departure from the "CRN regime" (see Section S5).

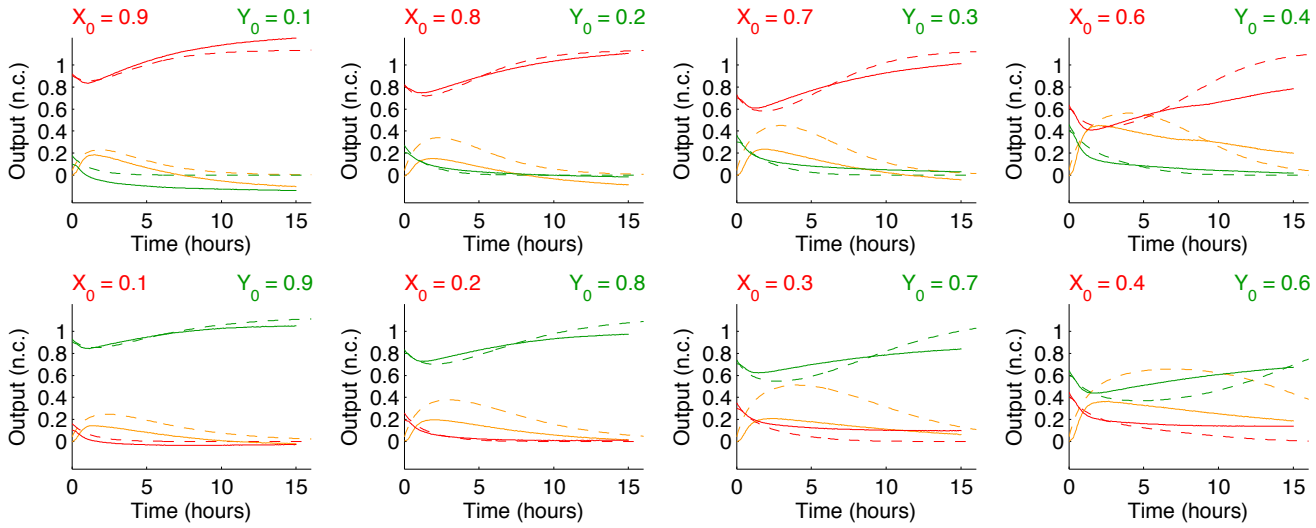


Figure S22: Comparison of the consensus network at $1x = 80$ nM with predictions from the formal CRN model. The three bimolecular rate constants fitted in separate experiments were multiplied by the best-fit interference parameter 0.37 to obtain: 9028/M/s, 2945/M/s, 1815/M/s, for the three formal reactions, respectively. Simulation of the formal consensus CRN with these rate constants is shown plotted dashed. The experiment data (solid) is as shown in Fig. 5c.

S8.4 DSD modeling of the consensus network

S8.4.1 Modeling strategy

In order to ensure that parameters were accurately inferred from experimental data, we used an estimation-prediction strategy. We fit model parameters to data measuring the components of the consensus algorithm, and then used these parameters to predict the behavior of the full system. In doing so we give credibility to the idea of being able to design arbitrarily complex networks from a toolbox of DNA parts.

S8.4.2 Model parameterization hypotheses

In Section S7 by default we assigned parameters to strand displacement reactions based on the binding and (where applicable) unbinding toeholds involved in the reaction. For the consensus network we also compared four different hypotheses for how to parametrize the strand displacement reactions:

- *Unique strands:* Distinct strands are assumed to have distinct reaction rates, where a given strand consists of a toehold domain followed by a long recognition domain.
- *Unique context:* As described in S7.1, distinct rates were assigned to reactions involving distinct binding and (where applicable) unbinding toeholds.
- *Internal-external join/fork*: Distinct *external* reaction rates were used for strands binding to either ends of a gate, with *internal* reaction rates for the remaining reactions. Here, distinct rates were used for Join and Fork circuits.
- *Internal-external*: As above, but the same rates were used for both Join and Fork circuits.

N.B. It was clear from the data measuring the reporters of <u3 pb>, <u3 px> and <u3 py> that sequence-specific rates were required to describe their kinetics, and so all hypotheses used separate rates for these reporters.

S8.4.3 Characterization of the kinetics of individual gates and corresponding model behavior

To obtain estimates of the parameter values for the consensus network components, we used our Bayesian parameter inference procedure described in Section S7.9 against data measuring the kinetics of each join and fork gate, and the data for the formal reactions in Fig. S21. All *forward* strand displacement reactions were characterized for the Join_{XY}, Join_{BX} and Join_{BY} gates using the same strategy as in Section S7.9.4 (Fig. S23). Additionally, all reverse strand displacement reactions were characterized for the Join gates, by preparing the gates as if they had already

received both inputs and helper, then stimulating with varying concentrations of the strands that are displaced in the forward orientation (Fig. S23). The first three forward strand displacement reactions were characterized for the $Fork_{BB}$, $Fork_{XX}$ and $Fork_{YY}$ gates (Fig. S24). We did not characterize the remaining forward rates nor reverse rates for the Fork gates. While this meant that the posterior distributions of these parameters were broader, we found that prediction of the full consensus network was unaffected by this level of uncertainty. The parameter inference procedure operated on all datasets in Figs. S21-S24 simultaneously, with the kinetic parameters shared between systems using equivalent components. Leak and bad strand parameters were also simultaneously inferred, similar to Section S7.9.

We found that we could not reproduce all of the kinetics with the model hypotheses we tested in their standard form. In particular, a conflict was observed between addition of the first input to fork gates (compare Fig. S24b.i with Fig. S24c.i) for the unique context and internal-external models. The unique strands model already contained different parameters for these interactions, but an alternative conflict was seen resulting from <t x> binding externally to the $Join_{XY}$ gate but internally to the $Join_{BX}$ gate, so an additional parameter was used to model internal versus external binding of <t x>. Therefore, a mixture of context and nucleotide-specifity was required to explain the kinetics data. To summarise the findings of applying the model parameterizations, we have shown the maximum likelihood score observed for each of the described hypotheses, with their corresponding modifications (Fig. S25). As expected, the ability to reproduce the experimental data correlated with the complexity of the parameterization, with the bars ordered in increasing numbers of kinetic parameters from left to right (unique strands - 36/37, unique context - 19/20, internal-external J/F - 10/11, internal-external - 8/9). However, the single parameter correction to the unique context hypothesis led to a better model than the default unique strands hypothesis, despite using almost half as many parameters (20 versus 37). The maximum likelihood parameter values in the unique context parameterization are shown in Table S4.

S8.4.4 Predicting the behavior of the consensus network

To determine the optimal parameterization, we used the estimates of the strand displacement reaction parameters to predict the kinetic behavior of the full consensus network. Initially, we consistently predicted behaviors that were faster than we had observed experimentally. Furthermore, even when attempting to fit the full consensus circuit with the kinetics data for its components simultaneously, we observed a conflict in the rates required by the full circuit and the individual components. None of our model parameterization hypotheses could reconcile this difference, as the full circuit is simply a composition of its components. We then simulated the full circuit with the kinetic rates scaled by a single factor in an attempt to resolve the kinetic mismatch. As expected, this had a large effect on the behavior of the model. We found that several of the parameterizations yielded very good predictions when the scale factor (from now on, the *interference parameter* f) was between 0.4 and 0.5 (Fig. S26). Slightly different values were found for optimally predicting the behavior of the network measured with $1x = 40$ nM (Fig. S27) and $1x = 80$ nM (Fig. 5), so here we report a weighted prediction score against both 40 nM and 80 nM regimes as a function of the interference parameter. The finding that slower rates are required to describe larger systems is consistent with previous work (e.g. [9] incorporated side reactions involving promiscuous toehold binding), though the exact underlying mechanism responsible for the apparent slowdown remains debated. We feel that the significantly improved performance resulting from a single homogeneous scaling to 17 kinetic parameters would not happen unless this was representative of a real biochemical effect.

Our predictions indicate that the unique context hypothesis represents the statistically optimal description of the consensus network, in terms of the data we had collected (Fig. S26). When attempting to fit the unique strands hypothesis, a more complex parameterization, we observe diminishing returns in predictive capability. This is a sign of over-fitting, a well-established (yet sometimes ignored) issue in parameter inference, which occurs when additional parameters start to model random fluctuations in the data arising from measurement or process error [8]. Therefore, while a unique strands parameterization may indeed be more realistic, our measurements are not sufficient to constrain all parameters. For example, we did not characterize the kinetics of reverse strand displacement reactions for the Fork gates. At the lower end of the complexity spectrum, we found that the internal-external join/fork hypothesis gave weaker predictions than the unique context hypothesis (Fig. S26). As the unique context hypothesis also incorporates the internal-external distinction, these results suggest that the difference between internal and external topology is a dominant source of variability in strand displacement reactions. Removing join/fork specificity in the reaction rates led to poorer behavior still, both in terms of reproducing kinetic behavior of individual gates (Fig. S25) and predicting the full network (Fig. S26).

S8.5 Experimental regime for the consensus network

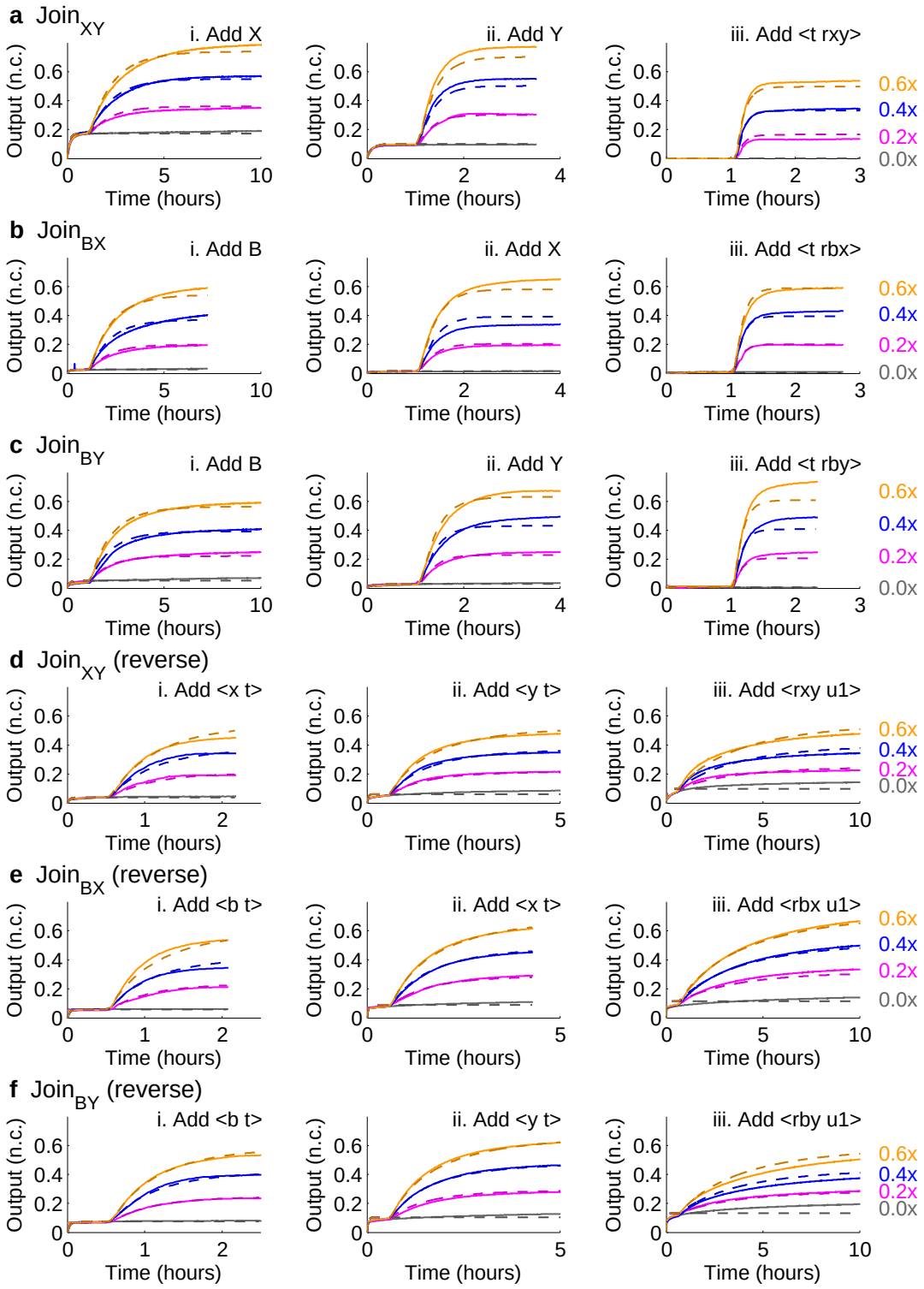


Figure S23: Characterisation and modelling of join gates for the consensus network. In all panels, kinetics data (solid lines) are compared with simulations of the unique context model (dashed lines) for **a**, Join$_{XY}$, **b**, Join$_{BX}$, **c**, Join$_{BY}$ gates, **d**, reversed Join$_{XY}$, **e**, reversed Join$_{BX}$ and **f**, reversed Join$_{BY}$ gates. Experiments were run in 1x TAE/Mg^{++} and performed at 25°C. The gates and reporters were at 30 nM concentration (3x), and all downstream or upstream auxiliary strands were at 100 nM concentration (10x). In a-c, for each row, panel i shows addition of the first (leftmost) input strands at indicated concentrations, panel ii shows addition of the second inputs, and panel iii shows addition of the helpers which displace the translator strands. *B* refers to a <t b> strand, *X* is <t x> and *Y* is <t y>. In d-f, the gates were prepared to measure strand displacement reactions in the reverse direction, with an equivalent measurement strategy used for monitoring the eventual displacement of *X* or *B* from the lefthand end of each gate.

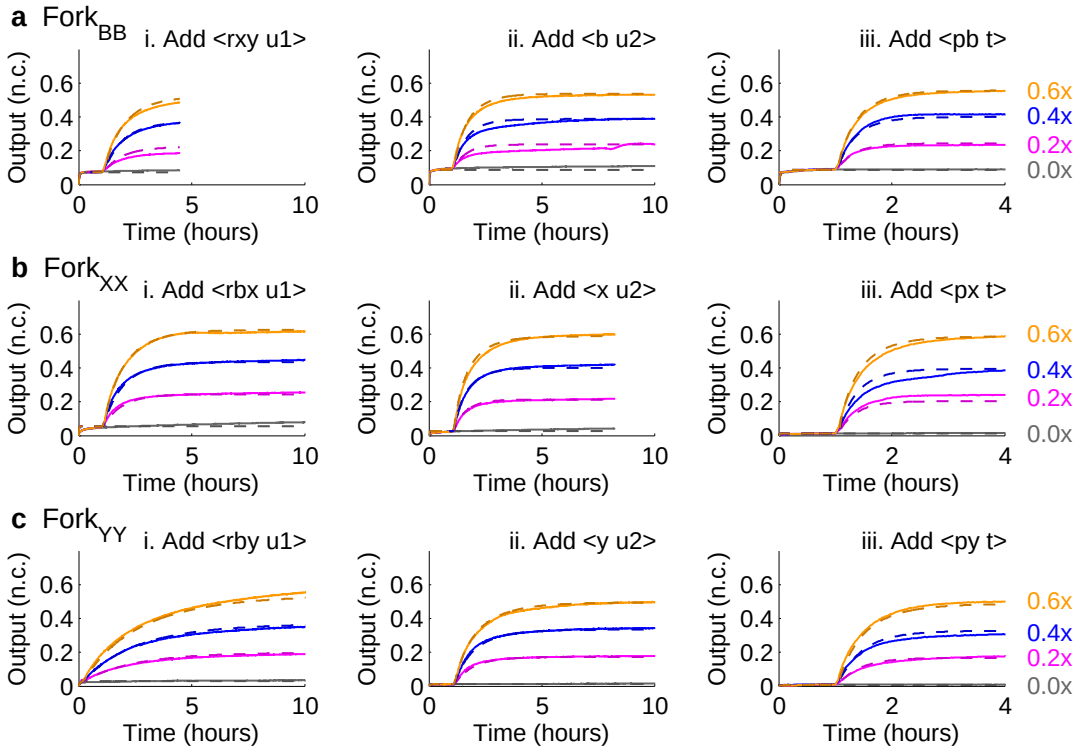


Figure S24: Characterisation and modelling of fork gates for the consensus network. In all panels, kinetics data (solid lines) are compared with simulations of the unique context model (dashed lines) for **a**, Fork$_{BB}$, **b**, Fork$_{XX}$ and **c**, Fork$_{YY}$ gates. Experiments were run in 1x TAE/Mg^{++} and performed at 25°C. The gates and reporters were at 30 nM concentration (3x), and all downstream or upstream auxiliary strands were at 100 nM concentration (10x). For each row, panel i shows addition of the first (rightmost) input strand at indicated concentrations, panel ii shows addition of the second input, and panel iii shows addition of the helpers which displace the reporter strands *PB*, *PX* and *PY*.

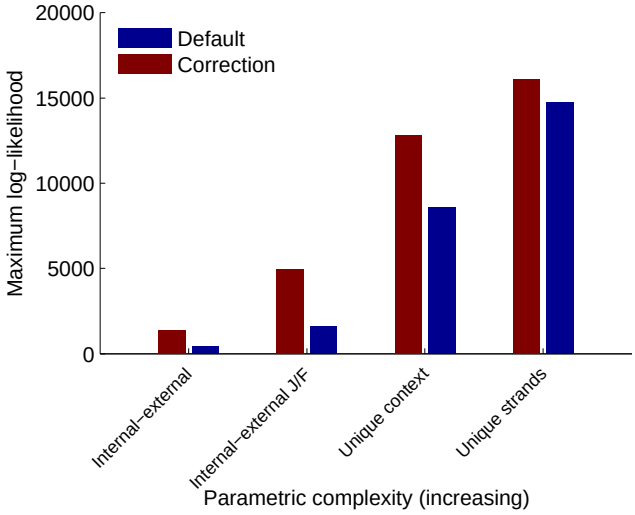


Figure S25: Comparison of model hypotheses in reproducing kinetics data for formal reactions and individual gates. The performance of each model parameterization hypothesis was evaluated by approximating the maximum log-likelihood score using the Filzbach MCMC software. Here, the MCMC runs comprised 100,000 burn-in iterations, followed by 500,000 sampling iterations to produce posterior densities, and then 100,000 further iterations to specifically attempt to maximize the log-likelihood. The blue bars indicate the default parameterization in each hypothesis, while the red bars show the effect of having a separate parameter for <t x> binding externally versus internally (for the unique strands hypothesis) or binding of the <rby u1> strand to the Fork$_{YY}$ gate (for the other hypotheses). The parameterizations are ordered by their number of kinetic parameters (increasing from left to right).

Table S4: Maximum likelihood parameter values in the unique context parameterization. Kinetic rate parameter names were chosen for Join (J) and Fork (F) circuits, and numbered based on the order in which the reactions take place as the circuits progress towards completion. For example, $kJ3$ denotes the rate constant for the third strand displacement reaction of the Join circuit. A subscript r was used to denote the reactions that oppose circuit completion. The names reflect the fact that all three Join and Fork gates have the same arrangement of toeholds, resulting in identical rate constants at corresponding positions along these gates. The exceptions to this were the use of a specific $kF1$ rate for the Fork$_{YY}$ gate, and strand-specific rates for the reporters for the outputs of the Fork gates (<u3 pb>, <u3 px> and <u3 py>). Leak and bad parameter names are of the form "leak{circuit}_{variant}", where the circuit was either XY2B, BX2X, BY2Y, JoinXY, JoinBX, JoinBY, ForkBB, ForkXX or ForkYY. The variants for XY2B, BX2X and BY2Y indicate leak and bad parameters associated with the Join (J) or Fork (F) gates. The variants for JoinAB and ForkCC circuits indicate the strand for which the kinetic rate was being characterized (see Figs. S23 and S24). All table entries are the maximum likelihood parameter values determined using the Filzbach MCMC software. A value is not provided for the 4th reverse rate on the Fork, $kF4_r$, as this rate was set equal to $kF2_r$ throughout.

Kinetic parameters		**Leak parameters**		**Bad parameters**	
Name	Value (M^{-1} s^{-1})	Name	Value	Name	Value
$kJ1$	9.34×10^3	leakXY2B_J	0.0142	badXY2B_X	0.0650
$kJ1_r$	5.89×10^4	leakXY2B_F	0.0098	badXY2B_Y	0.0881
$kJ2$	8.15×10^4	leakBX2X_J	0.0107	badBX2X_B	0.1239
$kJ2_r$	6.97×10^4	leakBX2X_F	0.0000	badBX2X_X	0.0700
$kJ3$	4.06×10^5	leakBY2Y_J	0.0088	badBY2Y_B	0.0000
$kJ3_r$	3.31×10^3	leakBY2Y_F	0.0175	badBY2Y_Y	0.2500
kJR	9.21×10^4	leakJoinXY_X	0.0572	badJoinXY_X	0.0545
kJR_r	2.98×10^4	leakJoinXY_Y	0.0338	badJoinXY_Y	0.0000
$kF1$	1.21×10^4	leakJoinXY_H	0.0000	badJoinXY_H	0.1742
$kF1^{YY}$	4.21×10^3	leakJoinXY_R1	0.0132	badJoinXY_R1	0.1628
$kF1_r$	1.30×10^5	leakJoinXY_R2	0.0205	badJoinXY_R2	0.2347
$kF2$	4.58×10^4	leakJoinXY_R3	0.0327	badJoinXY_R3	0.2500
$kF2_r$	4.79×10^5	leakJoinBX_B	0.0087	badJoinBX_B	0.1384
$kF3$	1.20×10^5	leakJoinBX_X	0.0049	badJoinBX_X	0.0564
$kF3_r$	1.20×10^3	leakJoinBX_H	0.0027	badJoinBX_H	0.0316
$kF4$	3.60×10^3	leakJoinBX_R1	0.0194	badJoinBX_R1	0.1156
$kF5$	7.20×10^3	leakJoinBX_R2	0.0293	badJoinBX_R2	0.0025
kFR^{PB}	3.14×10^5	leakJoinBX_R3	0.0381	badJoinBX_R3	0.0001
kFR^{PX}	6.65×10^5	leakJoinBY_B	0.0173	badJoinBY_B	0.1483
kFR^{PY}	1.56×10^5	leakJoinBY_Y	0.0101	badJoinBY_Y	0.0000
		leakJoinBY_H	0.0025	badJoinBY_H	0.0000
		leakJoinBY_R1	0.0248	badJoinBY_R1	0.1486
		leakJoinBY_R2	0.0347	badJoinBY_R2	0.0757
		leakJoinBY_R3	0.0441	badJoinBY_R3	0.2500
		leakForkBB_R	0.0242	badForkBB_R	0.2499
		leakForkBB_H1	0.0291	badForkBB_H1	0.2500
		leakForkBB_H2	0.0291	badForkBB_H2	0.2188
		leakForkXX_R	0.0182	badForkXX_R	0.0482
		leakForkXX_H1	0.0087	badForkXX_H1	0.0647
		leakForkXX_H2	0.0043	badForkXX_H2	0.0448
		leakForkYY_R	0.0099	badForkYY_R	0.1509
		leakForkYY_H1	0.0042	badForkYY_H1	0.1961
		leakForkYY_H2	0.0027	badForkYY_H2	0.2010

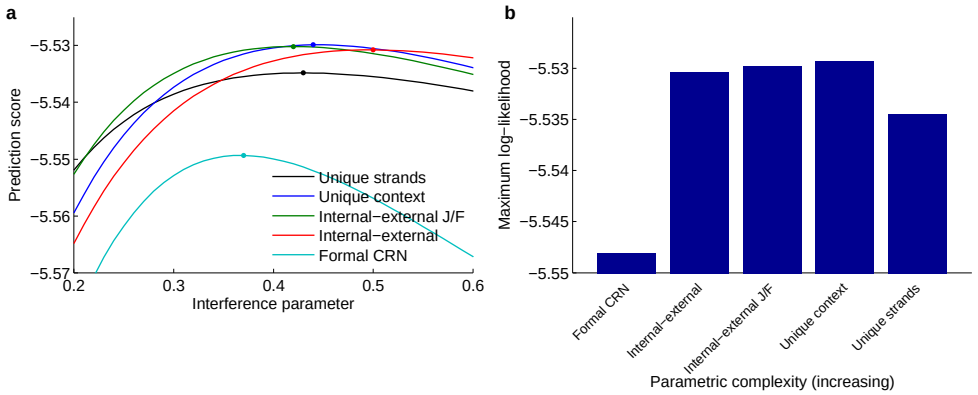


Figure S26: Comparison of model hypotheses in predicting the consensus network. The performance of each strand displacement model parameterization hypothesis and the formal CRN model was evaluated by calculating a weighted log-likelihood score using data for the consensus network when $1x = 80$ nM (Figure 4 of the main text) and when $1x = 40$ nM (Figure S27). The weights were the inverse of the number of data-points in each experiment. **a**, The *interference parameter* f was allowed to vary between 0.2 and 0.6 to determine the optimal prediction score as using the default value of 1 led to poor predictions in all cases. The maximum log-likelihood score is indicated by the circles. The "Formal CRN" prediction is as described in Section S8.3. **b**, Summary of the maximum log-likelihood scores for each model parameterization hypothesis, ordered from left to right by the number of kinetic parameters (increasing).

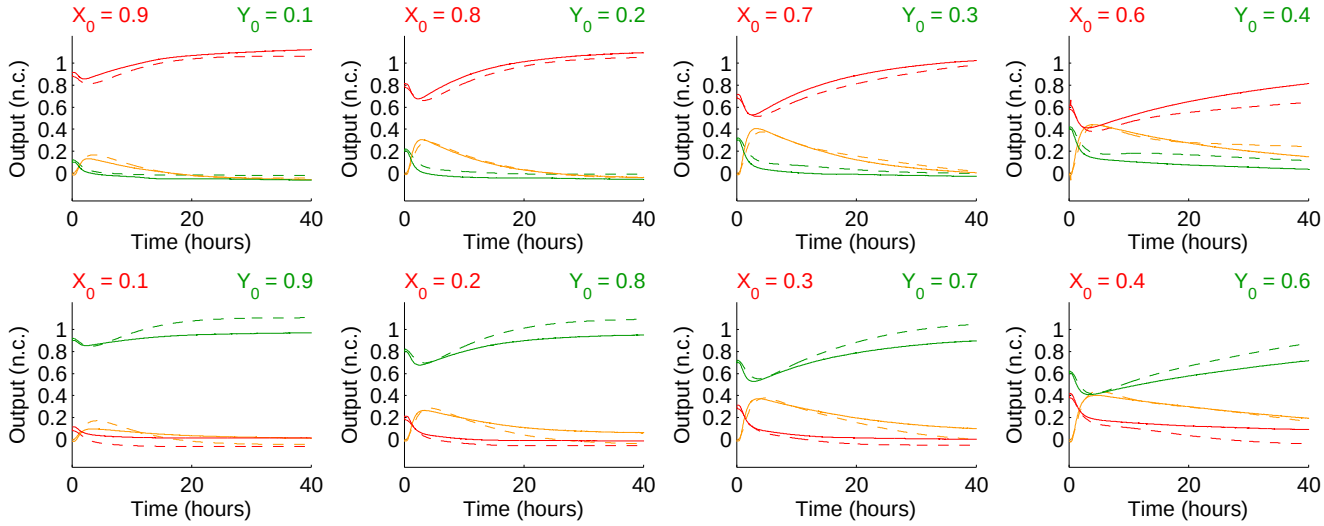


Figure S27: Comparison of the consensus network at $1x = 40$ nM with predictions from the unique context model. The consensus network was measured for $1x = 40$ nM, and compared with simulations of the strand displacement model with the unique context parameterization (see Table S4 for parameter values). Experimental regime is the same as for the 80 nM equivalent data in Figure 5 of the main text. In the simulations, the dynamics of the reporter strands PX, PY and PB were modelled explicitly. For both measurements and simulations, the signals X, Y and B were calculated according to $X = X_0 + PX - PB$, $Y = Y_0 + PY - PB$ and $B = 2PB - PX - PY$. The simulations used the interference parameter $f = 0.44$, which was the optimal compromized value for these data and the equivalent 80 nM set.

The graphical representation of all gates and auxiliary species of the consensus network is given in the panel (i) of Fig. S21. The network consisted of 3 join gates, 3 fork gates, 3 reporters, 13 auxiliary strands, and 3 signal strands. The consensus network was performed in two different standard concentrations (1x=40nM in Fig. S27, and 1x=80nM in Fig. 5). The initial concentrations of signal X and Y were indicated in each panel of Fig. S27 and Fig. 5, respectively. Reporters were at 3x. Reporters for PX, PY, and PB were used to follow the reaction kinetics without interfering with the dynamics of X, Y, and B. Join gates, fork gates and auxiliary strands for reaction $B + X \rightarrow 2X$ (Fig. S21a) and $B + Y \rightarrow 2Y$ (Fig. S21b) were at 2x. For reaction $B + Y \rightarrow 2Y$ (Fig. S21c), join gates, fork gates and auxiliary strands were at 2.4x. The auxiliary strand <ig t> was at 6.4x. No backward auxiliary strands were added to the reaction.

S9 Material and methods

S9.1 Sequence design

Strand displacement circuitry using double-stranded DNA was proposed in Ref. [3] by Cardelli. Here, we provide the first experimental verification of this design. Ref. [3] proposed a design where all toeholds have the same sequence but in our experimental work we used a variant of the original design where different toeholds have different sequence. The nicking enzyme recognition sites do not coincide with the toeholds but are part of the long domain sequences (Nb.BsrDI and Nt.BstNBI recognition sites are on the left and right side of the long domain, respectively, please see Fig. S3). An alternative design where nicking enzymes and toeholds coincide is shown in Fig. S4. In this case all toeholds are identical.

The use of a nicking enzyme to make breaks in one of the gate strands imposes certain sequence constraints but these constrains can be incorporated without loss of generality. There are enough unconstrained nucleotides in each strand or domain to ensure that all sequences can be made orthogonal, minimizing crosstalk. For a very large reaction network it may be desirable to further increase these differences. In this case, we can simply extend the length of the long domains and insert additional unconstrained nucleotides .

S9.2 Cloning stategy of plasmid-derived ndsDNA gates

In order to increase the yield of plasmid-derived ndsDNA gates, we cloned multiple identical ndsDNA gates into one plasmid. ndsDNA gate templetes were ordered as doubles-stranded gBlocks from Integrated DNA Technologies (IDT). Four different pairs of primers were used to amplify a gate complex, and an extra ~40 bp of overlapping sequence (red, orange, green, purple, and magenta) were added at each terminus (Fig. S28). The amplified parts were further purifed from 2% agarose gel. Purified fragments and the linearized vector were then combined with the Gibson Master Mix containing T5 exonuclease, Phusion DNA polymerase, and Taq ligase [5]. The reaction was incubated at 50°C for one hour, and during incubation, multiple copies of the gate complexes were assembled into the plasmid backbone.

S9.3 DNA synthesis and purification

DNA oligos were purchased from Integrated DNA Technologies (IDT). Inserts for gate cloning were ordered as double-stranded gBlocks. Long bottom strands of synthesized ndsDNA gates were ordered as polyacrylamide gel electrophoresis (PAGE) purified ultramers. Signal species X and Y were ordered purified by high-performance liquid chromatography (HPLC), and reporter strands with fluorophore/quencher modifications were also HPLC purified. Other single-stranded DNA were ordered as PAGE purified. Strands were re-suspended and 100 μM stock solutions were prepared in 1x TE.

S9.4 Preparation of synthesized gates

Join and fork gates were prepared from stock DNA solutions with 20% excess of top strands, in 1x Tris-acetate-EDTA buffer containing 12.5mM Mg^{++} (1x TAE/Mg^{++}). Resulting gate concentrations were typically in the range of 20 μM and total amounts were approximately 3 nmole. Gates were annealed in a thermal cycler, cooling from 95°C to 20°C at a rate of 0.1°C /min. This slow annealing process ensured the correct formation of complexes. Reporter complexes were annealed with a 20% excess of top strands, in 1x TAE/Mg^{++}, and a faster anneal was used, going from 90°C to 20°C at a rate of 1°C /min in order to save time. The excess of top strands with quenchers ensured complete quenching of fluorescent bottom strands. Join and fork gates were further purified to remove excess single-stranded DNA and poorly formed gate complexes using non-denaturing (ND) PAGE (gel preparation see S9.7). The proper bands were cut from the gel and soaked in 1x TAE/ Mg^{++} for two days at room temperature to elute the DNA.

S9.5 Kinetics experiments and fluorescence data normalization

Kinetics experiments were performed on a spectrofluorimeter (SPEX Fluorolog-3, Horiba) with 0.875 ml synthetic quartz cells (Starna catalog number 23-5.45-S0G-5). We used a 2.73 nm slit width for both excitation and emission monochromators. Excitation and emission wavelengths of different fluorophores were as follows: ROX (588 nm/ 608 nm), FAM (495 nm/ 520 nm), Cy3 (550 nm/ 564 nm), TYE665 (645 nm/ 665 nm), TAMRA (559 nm/ 583 nm), AL488 (492 nm/ 517 nm), and AL647 (650 nm/ 670 nm). The integration time was 10 sec for all experiments for

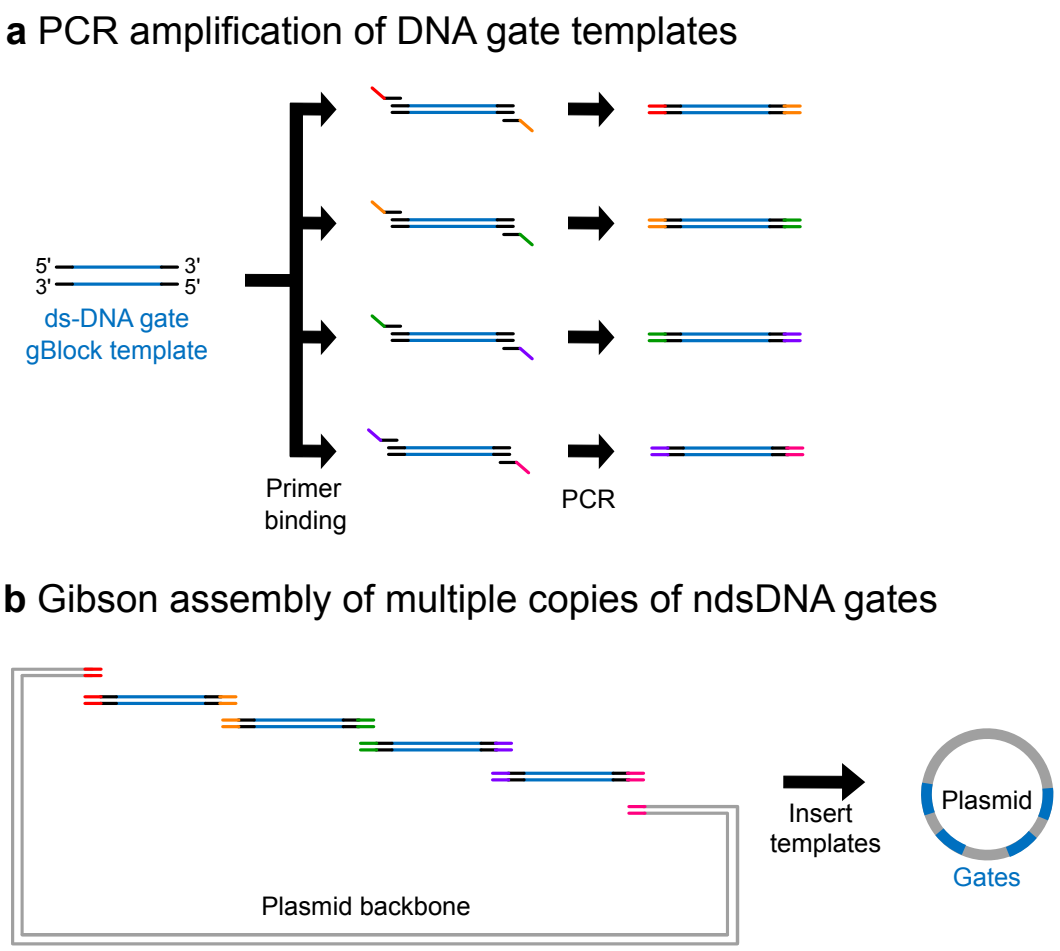


Figure S28: Cloning strategy of ndsDNA gates. **a**, Four pairs of primers were used to amply the ndsDNA gate template, and overlapping sequences were added to each end of the double-stranded gate complexes. Overlapping sequences are color-coded (red, orange, green, purple, and magenta). **b**, Four DNA fragments sharing terminal sequence overlaps were assembled into a linearized plasmid backbone using Gibson assembly cloning method.

every 60 sec time-point. Since DNA has been observed to non-specifically bind to pipette tips, a non-reactive 20 nt poly-T "carrier" strand at the concentration of 1 uM was used in all reactions so that the majority of DNA loss would be the carrier strands. Addition of carrier strands could improve the measurement accuracy.

In all kinetics experiments, arbitrary fluorescence units were converted to the concentrations of the corresponding signal strand using a calibration curve of each reporter complex (see S1.2 for details). Calibration experiments were run once for each reporter complex.

S9.6 Gate concentration quantification

The ndsDNA gate complex can be fully triggered by adding excess of all necessary signal and auxiliary strands, and then the concentration of the gate complex can be determined by reading off the produced fluoroscence signal. For example, in the case of Fork_C, a small amount of Fork_C (~0.5x) was mixed with 3x of the Reporter_C. Then excess amounts (~10x) of all necessary auxiliary strands (i.e. <i tc>, <c tr>, <r tq>) were added to trigger the Fork_C gate, After the reaction reached equilibrium, the concentration of Fork_C can be determined by reading off the concentration of the produced signal C.

S9.7 Gel electrophoresis

S9.7.1 Non-denaturing PAGE

10% non-denaturing PAGE gels were made by mixing 10 ml 19:1 40% acrylamide/bis, 4 mL 10x TAE/Mg^{2+}, and deionized water to 40 mL. Then 300 μL APS and 30 μL TEMED were added to help polymerization. 80% glycerol

was added to all annealed samples, achieving final glycerol concentration of 15% by volume. The samples were run on 10% ND PAGE at 140 V for 6 hrs at 25°C.

S9.7.2 Denaturing PAGE

10% denaturing PAGE gels were made by mixing 10 mL 19:1 40% acrylamide/bis, 4 mL 10x TBE, 16.8g urea, and deionized water to 40 mL, then adding 300 μL APS and 30 μL TEMED were added to help polymerization. For plasmid-derived sample, we loaded approximately 1 μg of DNA (including gates and plasmid backbone) and for synthetic DNA we loaded 150-200 ng of gate complex. 2x TBE/Urea denaturing loading buffer(Bio-Rad) was added to all samples in 1:1 ratio. Gels were run at 120 V for 3 hours at 55°C with the temperature controlled using an external temperature bath. Gels were stained with Sybr-Gold stain (Invitrogen) at room temperature for 20 minutes, and scanned with a Typhoon FLA 9000 biomolecular imager.

S10 Tables of sequences

Table S5: Domain sequences of ndsDNA gates with non-identical toeholds. All sequences start from the 5′ end.

Domain	Sequence	Length (nt)
ta	CTGCTA	6
tb	TTCCAC	6
tc	TACCCA	6
tr	TCCTAC	6
tq	AACCAG	6
a	CATTGCTTCTACGAGTCATCC	21
b	CATTGCACCTTAGAGTCCGAA	21
c	CATTGCCACATCGAGTCCCTT	21
r	CATTGCTTAACCGAGTCTCAC	21
i	CTGCCATCATAAGAGTCACCA	21

Table S6: Domain sequences of ndsDNA gates for consensus network.

Domain	Sequence	Length (nt)
t	CTGATC	6
u1	CTTCAG	6
u2	CCATAC	6
u3	ATACCC	6
x	CATTGCTTTATTTACCGAGTCTTAT	25
y	CATTGCCTAACCCACCGAGTCCTTT	25
b	CATTGCCAATTCCTACGAGTCTACC	25
pb	CATTGCATTATATTCCGAGTCCTAC	25
px	CATTGCCTTCCCACTAGAGTCTCAC	25
py	CATTGCACCACCCTAAGAGTCTAAC	25
rxy	CATTGCTACCACCTCCGAGTCTAAC	25
rbx	CATTGCCAAACCATTAGAGTCAAAC	25
rby	CATTGCACCCTAATACGAGTCTCAC	25
ig	CTGAAATAAATAAATAGAGTCTACC	25

Table S7: Domain sequences of ndsDNA gates for testing nicking enzymes in S3.1.

Domain	Sequence	Length (nt)
t	TCAGCT	6
a	CATTGCAACCTCAACCTAATCC	22
b	CATTGCATAACCACCTCATTCC	22
c	CATTGCCTTCCTAATTCTCACC	22
r	CATTGCCCAACATTAACCAACC	22

Table S8: Strand sequences of ndsDNA gates with non-identical toeholds (Domain sequences are in Table S5).

Strand	Domain	Sequence	Length (nt)
$Join_{AB}$-Bottom	tq* r* tr* b* tb* a* ta*	CTGGTT GTGAGACTCGGTTAAGCAATG GTAGGA TTCGGACTCTAAGGTGCAATG GTGGAA GGATGACTCGTAGAAGCAATG TAGCAG	87
$Fork_C$-Bottom	tq* r* tr* c* tc* i*	CTGGTT GTGAGACTCGGTTAAGCAATG GTAGGA AAGGGACTCGATGTGGCAATG TGGGTA TGGTGACTCTTATGATGGCAG	81
$Fork_{BC}$-Bottom	tq* r* tr* b* tb* c* tc* i*	CTGGTT GTGAGACTCGGTTAAGCAATG GTAGGA TTCGGACTCTAAGGTGCAATG GTGGAA AAGGGACTCGATGTGGCAATG TGGGTA TGGTGACTCTTATGATGGCAG	108
$Fork_{CBB}$-Bottom	tq* r* tr* c* tc* b* tb* b* tb* i*	CTGGTT GTGAGACTCGGTTAAGCAATG GTAGGA AAGGGACTCGATGTGGCAATG TGGGTA TTCGGACTCTAAGGTGCAATG GTGGAA TTCGGACTCTAAGGTGCAATG GTGGAA TGGTGACTCTTATGATGGCAG	135
$Fork_{BCB}$-Bottom	tq* r* tr* b* tb* c* tc* b* tb* i*	CTGGTT GTGAGACTCGGTTAAGCAATG GTAGGA TTCGGACTCTAAGGTGCAATG GTGGAA AAGGGACTCGATGTGGCAATG TGGGTA TTCGGACTCTAAGGTGCAATG GTGGAA TGGTGACTCTTATGATGGCAG	135
$Fork_{BBC}$-Bottom	tq* r* tr* b* tb* b* tb* c* tc* i*	CTGGTT GTGAGACTCGGTTAAGCAATG GTAGGA TTCGGACTCTAAGGTGCAATG GTGGAA TTCGGACTCTAAGGTGCAATG GTGGAA AAGGGACTCGATGTGGCAATG TGGGTA TGGTGACTCTTATGATGGCAG	135
<ta a>	ta a	CTGCTA CATTGCTTCTACGAGTCATCC	27
<tb b>	tb b	TTCCAC CATTGCACCTTAGAGTCCGAA	27
<tc c>	tc c	TACCCA CATTGCCACATCGAGTCCCTT	27
<a tb>	a tb	CATTGCTTCTACGAGTCATCC TTCCAC	27
<b tr>	b tr	CATTGCACCTTAGAGTCCGAA TCCTAC	27
<r tq>	r tq	CATTGCTTAACCGAGTCTCAC AACCAG	27
<i>	i	CTGCCATCATAAGAGTCACCA	21
<tr r>	tr r	TCCTAC CATTGCTTAACCGAGTCTCAC	27
<i tc>	i tc	CTGCCATCATAAGAGTCACCA TACCCA	27
<c tr>	c tr	CATTGCCACATCGAGTCCCTT TCCTAC	27
<c tb>	c tb	CATTGCCACATCGAGTCCCTT TTCCAC	27
<b tr>	b tr	CATTGCACCTTAGAGTCCGAA TCCTAC	27
<i tb>	i tb	CTGCCATCATAAGAGTCACCA TTCCAC	27
<b tb>	b tb	CATTGCACCTTAGAGTCCGAA TTCCAC	27
<b tc>	b tc	CATTGCACCTTAGAGTCCGAA TACCCA	27
<c tr>	c tr	CATTGCCACATCGAGTCCCTT TCCTAC	27
<b tr>	b tr	CATTGCACCTTAGAGTCCGAA TCCTAC	27
TYE665-<a* ta*>	a* ta*	/5TYE665/ GGATGACTCGTAGAAGCAATG TAGCAG	27
<a>-RQ	a	CATTGCTTCTACGAGTCATCC /3IAbRQSp/	21
Cy3-<b* tb*>	b* tb*	/5Cy3/ TTCGGACTCTAAGGTGCAATG GTGGAA	27
<b>-FQ	b	CATTGCACCTTAGAGTCCGAA /3IABkFQ/	21
ROX-<c* tc*>	c* tc*	/56-ROXN/ AAGGGACTCGATGTGGCAATG TGGGTA	27
<c>-RQ	c	CATTGCCACATCGAGTCCCTT /3IAbRQSp/	21
<tq* r*>-TAMRA	tq* r*	CTGGTT GTGAGACTCGGTTAAGCAATG /36-TAMTSp/	27
RQ-<r>	r	/5IAbRQ/ CATTGCTTAACCGAGTCTCAC	21

Table S9: Strands occur in ndsDNA gates with non-identical toeholds (Strand sequences are in Table S8)..

Gate	Strand	Length of bottom strand (nt)
$Join_{AB}$	$Join_{AB}$-Bottom, <a tb>, <b tr>, <r tq>	87
$Fork_{C}$	$Fork_{C}$-Bottom, <i>, <tc c>, <tr r>	81
$Fork_{BC}$	$Fork_{BC}$-Bottom, <i>, <tc c>, <tb b>, <tr r>	108
$Fork_{CBB}$	$Fork_{CBB}$-Bottom, <i>, <tb b>, <tb b>, <tc c>, <tr r>	135
$Fork_{BCB}$	$Fork_{BCB}$-Bottom, <i>, <tb b>, <tc c>, <tb b>, <tr r>	135
$Fork_{BBC}$	$Fork_{BBC}$-Bottom, <i>, <tc c>, <tb b>, <tb b>, <tr r>	135
Reporter-<ta a>	TYE665-<a* ta*>, <a>-RQ	27
Reporter-<tb b>	Cy3-<b* tb*>, <b>-FQ	27
Reporter-<tc c>	ROX-<c* tc*>, <c>-RQ	27
Reporter-<r tq>	<tq* r*>-TAMRA, RQ-<r>	27

Table S10: Strand sequences of ndsDNA gates for concensus network. (Domain sequences are in S6).

Strand	Domain	Sequence	Length (nt)
$Join_{XY}$-Bottom	u1* rxy* t* y* t* x* t*	CTGAAG GTTAGACTCGGAGGTGGTAGCAATG GATCAG AAAGGACTCGGTGGGTTAGGCAATG GATCAG ATAAGACTCGGTAAATAAAGCAATG GATCAG	99
$Join_{BX}$-Bottom	u1* rbx* t* x* t* b* t*	CTGAAG GTTTGACTCTAATGGTTTGGCAATG GATCAG ATAAGACTCGGTAAATAAAGCAATG GATCAG GGTAGACTCGTAGGAATTGGCAATG GATCAG	99
$Join_{BY}$-Bottom	u1* rby* t* y* t* b* t*	CTGAAG GTGAGACTCGTATTAGGGTGCAATG GATCAG AAAGGACTCGGTGGGTTAGGCAATG GATCAG GGTAGACTCGTAGGAATTGGCAATG GATCAG	99
$Fork_{BB}$-Bottom	u1* rxy* u2* b* t* pb* u3* b* t* ig*	CTGAAG GTTAGACTCGGAGGTGGTAGCAATG GTATGG GGTAGACTCGTAGGAATTGGCAATG GATCAG GTAGGACTCGGAATATAATGCAATG GGGTAT GGTAGACTCGTAGGAATTGGCAATG GATCAG GGTAGACTCTATTTATTTATTTCAG	155
$Fork_{XX}$-Bottom	u1* rbx* u2* x* t* px* u3* x* t* ig*	CTGAAG GTTTGACTCTAATGGTTTGGCAATG GTATGG ATAAGACTCGGTAAATAAAGCAATG GATCAG GTGAGACTCTAGTGGGAAGGCAATG GGGTAT ATAAGACTCGGTAAATAAAGCAATG GATCAG GGTAGACTCTATTTATTTATTTCAG	155
$Fork_{YY}$-Bottom	u1* rby* u2* y* t* py* u3* y* t* ig*	CTGAAG GTGAGACTCGTATTAGGGTGCAATG GTATGG AAAGGACTCGGTGGGTTAGGCAATG GATCAG GTTAGACTCTTAGGGTGGTGCAATG GGGTAT AAAGGACTCGGTGGGTTAGGCAATG GATCAG GGTAGACTCTATTTATTTATTTCAG	155
<t x>	t x	CTGATC CATTGCTTTATTTACCGAGTCTTAT	31
<t y>	t y	CTGATC CATTGCCTAACCCACCGAGTCCTTT	31
<t b>	t b	CTGATC CATTGCCAATTCCTACGAGTCTACC	31
<u3 px>	u3 px	ATACCC CATTGCCTTCCCACTAGAGTCTCAC	31
<u3 py>	u3 py	ATACCC CATTGCACCACCCTAAGAGTCTAAC	31
<u3 pb>	u3 pb	ATACCC CATTGCATTATATTCCGAGTCCTAC	31
<x t>	x t	CATTGCTTTATTTACCGAGTCTTAT CTGATC	31
<y t>	y t	CATTGCCTAACCCACCGAGTCCTTT CTGATC	31
<b t>	b t	CATTGCCAATTCCTACGAGTCTACC CTGATC	31

<rxy u1>	rxy u1	CATTGCTACCACCTCCGAGTCTAAC CTTCAG	31
<rbx u1>	rbx u1	CATTGCCAAACCATTAGAGTCAAAC CTTCAG	31
<rby u1>	rby u1	CATTGCACCCTAATACGAGTCTCAC CTTCAG	31
<ig>	ig	CTGAAATAAATAAATAGAGTCTACC	25
<u2 rxy>	u2 rxy	CCATAC CATTGCTACCACCTCCGAGTCTAAC	31
<u2 rbx>	u2 rbx	CCATAC CATTGCCAAACCATTAGAGTCAAAC	31
<u2 rby>	u2 rby	CCATAC CATTGCACCCTAATACGAGTCTCAC	31
<ig t>	ig t	CTGAAATAAATAAATAGAGTCTACC CTGATC	31
<b u3>	b u3	CATTGCCAATTCCTACGAGTCTACC ATACCC	31
<pb t>	pb t	CATTGCATTATATTCCGAGTCCTAC CTGATC	31
<b u2>	b u2	CATTGCCAATTCCTACGAGTCTACC CCATAC	31
<x u3>	x u3	CATTGCTTTATTTACCGAGTCTTAT ATACCC	31
<px t>	px t	CATTGCCTTCCCACTAGAGTCTCAC CTGATC	31
<x u2>	x u2	CATTGCTTTATTTACCGAGTCTTAT CCATAC	31
<y u3>	y u3	CATTGCCTAACCCACCGAGTCCTTT ATACCC	31
<py t>	py t	CATTGCACCACCCTAAGAGTCTAAC CTGATC	31
<y u2>	y u2	CATTGCCTAACCCACCGAGTCCTTT CCATAC	31
FAM-<px* u3*>	px* u3*	/56-FAM/ GTGAGACTCTAGTGGGAAGGCAATG GGGTAT	31
<px>-FQ	px	CATTGCCTTCCCACTAGAGTCTCAC /3IABkFQ/	25
TYE665-<py* u3*>	py* u3*	/5TYE665/ GTTAGACTCTTAGGGTGGTGCAATG GGGTAT	31
<py>-RQ	py	CATTGCACCACCCTAAGAGTCTAAC /3IAbRQSp/	25
ROX-<pb* u3*>	pb* u3*	/56-ROXN/ GTAGGACTCGGAATATAATGCAATG GGGTAT	31
<pb>-RQ	pb	CATTGCATTATATTCCGAGTCCTAC /3IAbRQSp/	25
ALEX488-<b* t*>	b* t*	/5Alex488N/ GGTAGACTCGTAGGAATTGGCAATG GATCAG	31
<b>-FQ	b	CATTGCCAATTCCTACGAGTCTACC /3IABkFQ/	25
ROX-<x* t*>	x* t*	/56-ROXN/ ATAAGACTCGGTAAATAAAGCAATG GATCAG	31
<x>-RQ	x	CATTGCTTTATTTACCGAGTCTTAT /3IAbRQSp/	25
ALEX647-<y* t*>	y* t*	/5Alex647N/ AAAGGACTCGGTGGGTTAGGCAATG GATCAG	31
<y>-RQ	y	CATTGCCTAACCCACCGAGTCCTTT /3IAbRQSp/	25
<u1* rxy*>-TAMRA	u1* rxy*	CTGAAG GTTAGACTCGGAGGTGGTAGCAATG /36-TAMTSp/	31
RQ-<rxy>	rxy	/5IAbRQ/ CATTGCTACCACCTCCGAGTCTAAC	25
<u1* rbx*>-TAMRA	u1* rbx*	CTGAAG GTTTGACTCTAATGGTTTGGCAATG /36-TAMTSp/	31
RQ-<rbx>	rbx	/5IAbRQ/ CATTGCCAAACCATTAGAGTCAAAC	25
<u1* rby*>-TAMRA	u1* rby*	CTGAAG GTGAGACTCGTATTAGGGTGCAATG /36-TAMTSp/	31
RQ-<rby>	rby	/5IAbRQ/ CATTGCACCCTAATACGAGTCTCAC	25

Table S11: Strands occur in ndsDNA gates for concensus network (The strand sequences are in Table S10)..

Gate	Strand	Length of bottom strand (nt)
$Join_{XY}$	$Join_{XY}$-Bottom, <x t>, <y t>, <rxy u1>	155
$Join_{BX}$	$Join_{BX}$-Bottom, <b t>, <x t>, <rbx u1>	155
$Join_{BY}$	$Join_{BY}$-Bottom, <b t>, <y t>, <rby u1>	155
$Fork_{BB}$	$Fork_{BB}$-Bottom, <ig>, <t b>, <u3 pb>, <t b>, <u2 rxy>	155
$Fork_{XX}$	$Fork_{XX}$-Bottom, <ig>, <t x>, <u3 px>, <t x>, <u2 rbx>	155
$Fork_{YY}$	$Fork_{YY}$-Bottom, <ig>, <t y>, <u3 py>, <t y>, <u2 rby>	155
Reporter-<u3 px>	FAM-<px* u3*>, <px>-FQ	31
Reporter-<u3 py>	TYE665-<py* u3*>, <py>-RQ	31
Reporter-<u3 pb>	ROX-<pb* u3*>, <pb>-RQ	31
Reporter-<t x>	ROX-<x* t*>, <x>-RQ	31
Reporter-<t y>	ALEX647-<y* t*>, <y>-RQ	31
Reporter-<t b>	ALEX488-<b* t*>, <b>-FQ	31
Reporter-<u1 rxy>	<u1* rxy*>-TAMRA, RQ-<rxy>	31
Reporter-<u1 rbx>	<u1* rbx*>-TAMRA, RQ-<rbx>	31
Reporter-<u1 rby>	<u1* rby*>-TAMRA, RQ-<rby>	31

Table S12: Strand sequences of ndsDNA gates for testing nicking enzymes in S3.1 (Domain sequences are in Table S7).

Strand	Domain	Sequence	Length (nt)
$Join_{AB}$-Bottom	r* t* r* t* b* t* a* t*	GGTTGGTTAATGTTGGGCAATG AGCTGA GGTTGGTTAATGTTGGGCAATG AGCTGA GGAATGAGGTGGTTATGCAATG AGCTGA GGATTAGGTTGAGGTTGCAATG AGCTGA	112
$Fork_C$-Bottom	t* r* t* c* t* a*	AGCTGA GGTTGGTTAATGTTGGGCAATG AGCTGA GGTGAGAATTAGGAAGGCAATG AGCTGA GGATTAGGTTGAGGTTGCAATG	84
<t a>	t a	TCAGCT CATTGCAACCTCAACCTAATCC	28
<t b>	t b	TCAGCT CATTGCATAACCACCTCATTCC	28
<t c>	t c	TCAGCT CATTGCCTTCCTAATTCTCACC	28
<a t>	a t	CATTGCAACCTCAACCTAATCC TCAGCT	28
<b t>	b t	CATTGCATAACCACCTCATTCC TCAGCT	28
<r t>	r t	CATTGCCCAACATTAACCAACC TCAGCT	28
<r>	r	CATTGCCCAACATTAACCAACC	22
<a>	a	CATTGCAACCTCAACCTAATCC	22
<t r>	t r	TCAGCT CATTGCCCAACATTAACCAACC	28

Table S13: Strands occur in ndsDNA gates for testing nicking enzymes in S3.1. (Strand sequences are in Table S12).

Gate	Strand	Length of bottom strand (nt)
$Join_{AB}$	$Join_{AB}$-Bottom, <a tb>, <b tr>, <r tq>	112
$Fork_C$	$Fork_C$-Bottom, <i>, <tc c>, <tr r>	84

References

[1] http://mstlab.org/eng/projects/pages/solvers.aspx.

[2] http://research.microsoft.com/science/tools.

[3] Luca Cardelli. Two-domain dna strand displacement. *Mathematical Structures in Computer Science*, 23(02):247–271, 2013.

[4] Xi Chen, Neima Briggs, Jeremy R. McLain, and Andrew D. Ellington. Stacking nonenzymatic circuits for high signal gain. *Proc Natl Acad Sci U S A*, 110(14):5386–5391, Apr 2013.

[5] Daniel G Gibson, Lei Young, Ray-Yuan Chuang, J Craig Venter, Clyde A Hutchison, and Hamilton O Smith. Enzymatic assembly of dna molecules up to several hundred kilobases. *Nature methods*, 6(5):343–345, 2009.

[6] Matthew R Lakin, Simon Youssef, Luca Cardelli, and Andrew Phillips. Abstractions for DNA circuit design. *J R Soc Interface*, 9(68):470–486, Mar 2012.

[7] Matthew R Lakin, Simon Youssef, Filippo Polo, Stephen Emmott, and Andrew Phillips. Visual DSD: a design and analysis tool for DNA strand displacement systems. *Bioinformatics*, 27(22):3211–3213, Nov 2011.

[8] Tom Mitchell. *Machine learning*. McGraw-Hill, 1997.

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

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

[11] David Soloveichik, Georg Seelig, and Erik Winfree. DNA as a universal substrate for chemical kinetics. *Proc Natl Acad Sci U S A*, 107(12):5393–5398, Mar 2010.

[12] David Yu Zhang and Erik Winfree. Control of DNA strand displacement kinetics using toehold exchange. *J Am Chem Soc*, 131(47):17303–17314, Dec 2009.