Electronic Supplementary Material (ESI) for Nanoscale.

Nicking Enzymes Controlled Toehold Regulation for DNA Logic Circuit

Supplementary Information

Linqiang Pan,[a,b] Zhiyu Wang,[a] Yifan Li, Fei Xu,[*a] Qiang Zhang,[*e] Cheng Zhang[*c]

a Key Laboratory of Image Information Processing and Intelligent Control of Education Ministry of China, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China. E-mail: fei_xu@hust.edu.cn

b School of Electric and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, Henan, China.

c Institute of Software, School of Electronics Engineering and Computer Science, Peking University. Key laboratory of High Confidence Software Technologies, Ministry of Education, Beijing 100871, China. E-mail: zhangcheng369@pku.edu.cn

d School of Control and Computer Engineering, North China Electric Power University, Beijing 102206, China.

e College of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China. E-mail: zhangq@dlu.edu.cn

Contents

Note S1: Experimental section

Materials. DNA oligonucleotides were PAGE gel-purified and purchased from Sangon Biotech Co., Ltd.(Shanghai, China). Nicking enzyme Nt.BbvCI, Nt.BsmAI, Nb.BtsI and CutSmart buffer were purchased from New England Biolabs Inc.

DNA assembly. DNA structures B:C, E:F, I and G were generated by mixing corresponding single strands with equal concentrations of 8 uM in 1×Cutsmart buffer and each of the final concentrations of B:C, E:F, I and G was 4 uM. The sample was annealed in a polymerase chain reaction (PCR) thermal cycler to control the temperature from 90°C to 22°C at a rate of -4°C every 5 minutes.

Catalysis of nicking enzymes. The catalysis of nicking enzymes was performed in 1×Cutsmart buffer at 25°C. For YES, NOT AND and INHIBIT gate, the catalysis time was 1 hour, and for SWITCH logic circuit, the catalysis time was 2.5 hours. After catalysis, it took 3.5 hours for strand displacement.

Native polyacrylamide gel electrophoresis. Samples were run on 12% native polyacrylamide gel in 1×TAE buffer at 85V for 2 hours at 4°C. Gels were scanned with a Fluorchem FC2 gel scanner.

Quantitative real-time polymerase chain reaction. The fluorescent experiments were implemented using real-time PCR (Agilent, G8830A) equipped with a 96-well fluorescence plate reader. In a typical 25-uL reaction volume, the 1× reaction concentration was 0.4 uM. The reactions were performed in NEB Cutsmart buffer. Fluorescence intensity was measured every 3 minutes for 300 minutes (YES, NOT, AND and INHIBIT) or 390 minutes (SWITCH). Each experiment was repeated three times to ensure reproducibility. In each reaction, the fluorescence intensity was calculated by subtracting the initial intensity baseline at each time point. The error bars were also drawn on the fluorescence curves in each figure.

Note S2: Reaction simulation

A three-step cleaving-displacement reaction with a nicking enzyme can be modeled as:

$$\mathrm{BC} + \mathrm{n1} \xrightarrow{K_n} \mathrm{B1B2C} + \mathrm{n1} \quad ①$$

$$\mathrm{B1B2C} \xrightarrow{K_1} \mathrm{B1} + \mathrm{B2C} \quad ②$$

$$\mathrm{B2C} + \mathrm{A} \xrightarrow{K_2} \mathrm{AC} + \mathrm{B2} \quad ③$$

Therefore, the rate equation of B1B2C can be derived from reactions ① and ② as:

$$d[\mathrm{B1B2C}]/dt = K_n[\mathrm{BC}][\mathrm{n1}] - K_1[\mathrm{B1B2C}] \quad (1)$$

The rate equation of B2C can be derived from reactions ② and ③ as:

$$d[\mathrm{B2C}]/dt = K_1[\mathrm{B1B2C}] - K_2[\mathrm{B2C}][\mathrm{A}] \quad (2)$$

The rate equation of AC can be derived from reaction ③ as:

$$d[\mathrm{AC}]/dt = K_2[\mathrm{B2C}][\mathrm{A}] \quad (3)$$

When the initial condition is $[BC]_0$, $[A]_0$, [n1], the mass balance equations are:

$$[\mathrm{BC}]_0 = [\mathrm{B1B2C}] + [\mathrm{B1}] \quad (4)$$

$$[\mathrm{B1}] = [\mathrm{B2C}] + [\mathrm{AC}] \quad (5)$$

$$[\mathrm{A}]_0 = [\mathrm{A}] + [\mathrm{AC}] \quad (6)$$

$$[\mathrm{AC}] = [\mathrm{B2}] \quad (7)$$

The curve of [AC] can be obtained either by integration[2] or solving difference equations. For example, the differential equations above could be described by the following difference equations:

$$\mathrm{B1B2C}(k+1) = \mathrm{B1B2C}(k) + K_n \cdot \mathrm{BC}(k) \cdot [\mathrm{n1}] - K_1 \cdot \mathrm{B1B2C}(k) \quad (8)$$

$$\mathrm{B2C}(k+1) = \mathrm{B2C}(k) + K_1 \cdot \mathrm{B1B2C}(k) - K_2 \cdot \mathrm{B2C}(k) \cdot \mathrm{A}(k) \quad (9)$$

$$\mathrm{AC}(k+1) = \mathrm{AC}(k) + K_2 \cdot \mathrm{B2C}(k) \cdot \mathrm{A}(k) \quad (10)$$

The model above has good agreement with the data (**Figure S1a**) and K_n, K_1 and K_2 were fit to 2.67×10^{-9} L·unit^{-1}·s^{-1}[①], 8.43×10^{-3} M^{-1}·s^{-1} and 1.03×10^{4} M^{-1}·s^{-1}, respectively. According to the parameter values above, the reaction simulation for all the products were performed (**Figure S1b**). The olive curve (AC) corresponds to the red curve in (a). The concentration curves may appear in three types: rising, decreasing and rising followed by decreasing.

The concentrations of B1 and AC showed rising curves, because they were generated products. In addition, the increase of AC appeared a little delay after that of B1, for the reason that the reaction ② occurred prior to reaction ③. The concentrations of BC and A showed decreasing curves for they only served as consumed reactants.

Specially, in the simulation, the concentration of B1B2C increased at the beginning, but after reached its summit, the concentration began to decrease. This is because the reaction rate is a result depending on the production and consumption according to equation (1). In the equation (1), the

[①]As the amount of nicking enzyme is not measured with concentration in most cases, we introduced "unit concentration" to describe the concentration of enzyme. For example, the unit concentration of the nicking enzyme in this reaction was 2×10^5 units/L (According to NEB, one unit is defined as the amount of enzyme required to convert 1 μg of supercoiled plasmid DNA to open circular form in 1 hour at 37° C in a total reaction volume of 50 μl.). Therefore, the unit of K_n is L·unit^{-1}·s^{-1}.

production rate depends on the concentration of BC, which decreased all the time. On the other hand, the consumption rate depends on the concentration of B1B2C, which kept increasing at the initial stage. As a result, in the whole reaction process, the concentration of B1B2C increased initially and then decreased, because consumption rate eventually surpassed the production rate. The concentration curve of B2C was similar to that of B1B2C.

In the simulation, the concentration of B2C reached a summit of 2.26×10^{-8}M at 8th minute, while the reaction rate of AC reached a summit of 1.05×10^{-8}M/s at 7th minute. The reason for the time difference between the two summits is that, in equation (3), the concentrations of B2C and A varied asynchronously.

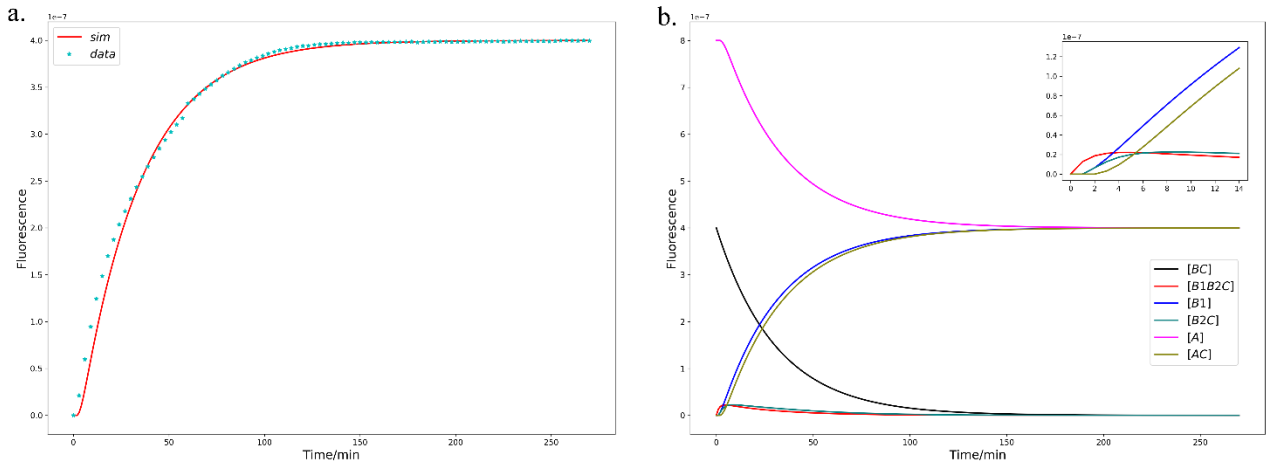

Figure S1. Reaction rate constants for the cleaving-displacement reaction and simulative results for the products in the reaction. (a) Simulation for the reaction in which the nicking enzyme and the displacement strand were added simultaneously (line 6 in **Figure 2a**). [BC]=400nM, [A]=800nM, [n1]=2×10^{5} units/L. (b) Simulative results for the products according to the values of K_n, K_1 and K_2 in (a). The concentration of B1B2C and B2C stayed a low level during the whole reaction because the consumption rate approached the production rate. The concentration of AC showed a delay form that of B1 due to the reaction order.

To investigate the details of the reaction, the simulation for the reaction rate of each product was performed based on the parameters obtained above (**Figure S2**). The simulative results showed that the maximum reaction rate of BC was -1.28×10^{-8}M/s (at the beginning), of B1B2C was 1.28×10^{-8}/s (at the beginning), of B1 was 1.12×10^{-8}M/s (at 5th minute), of B2C was 0.65×10^{-8}M/s (at 1st minute), of A was -1.05×10^{-8}M/s (at 7th minute), of AC was 1.05×10^{-8}M/s (at 7th minute). Specially, in the

reactions, both the products B1B2C and B2C possessed producing and consuming procedures. For B1B2C, it came from BC, and was converted into B1 and B2C. Due to the low concentration of B1B2C at the beginning, the consumption rate was extremely low, and thus the concentration of B1B2C increased. As the concentration of B1B2C increased, the consumption rate increased and finally the consumption rate surpassed the production rate. Therefore, the concentration of B1B2C presented consuming trend. In the simulation, the consuming rate of B1B2C reached a maximum of 6.21×10^{-10}M/s at 9th minute. For B2C, the production rate depended on the concentration of B1B2C. Initially the concentration of B1B2C was zero so that the initial production rate of B2C was zero. According to equation (2), the consumption rate of B2C depended on the concentrations of both A and B2C. With the increasing concentration of B2C, the consumption rate of B2C increased dramatically, and eventually the concentration of B2C presented a consuming trend. In the simulation, the consuming rate of B2C reached a maximum of 3.73×10^{-10}M/s at 16th minute.

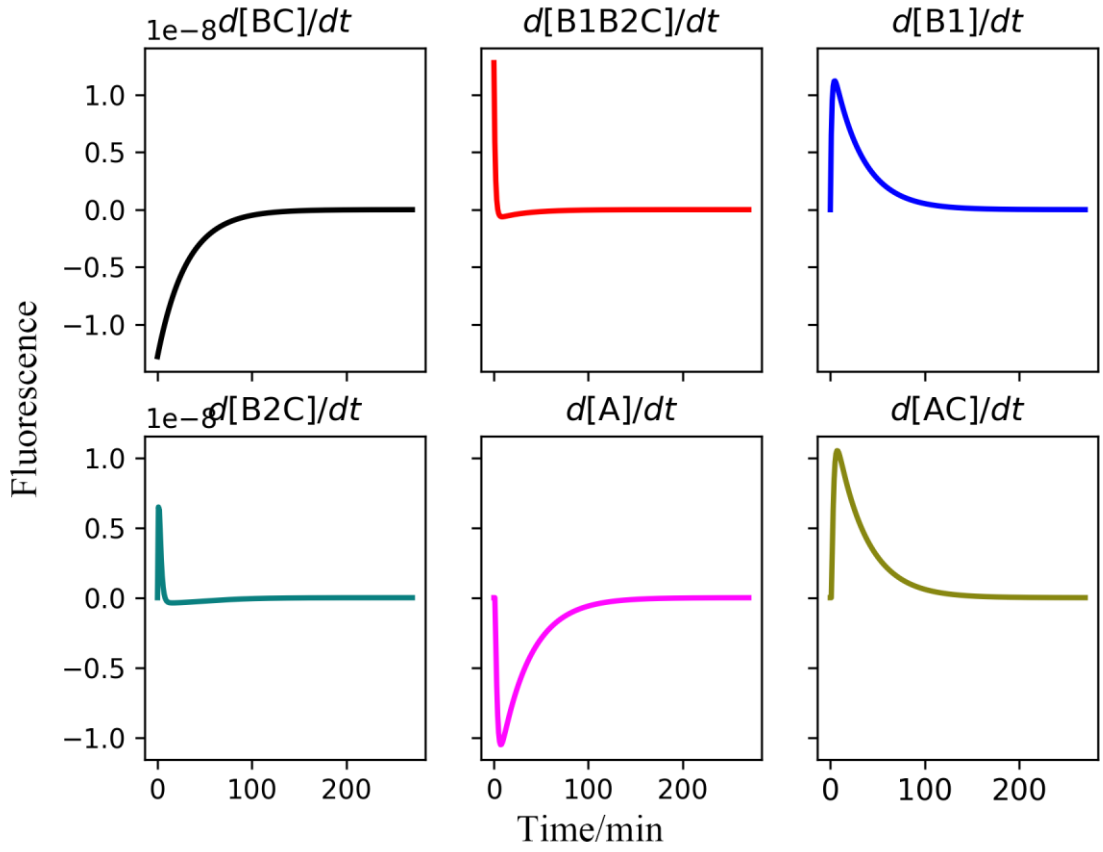

Figure S2. Simulative results for reaction rates of each product. As BC and A were always consumed, their reaction rates show negative values. As B1 and AC were always produced, their reaction rates show positive values. Since B1B2C and B2C would be consumed after being produced, their reaction rate shows positive values then negative values.

Note S3: NOT gate

A NOT gate performs the operation that reverts the input value, for example, a True input leads to a False output. So we can define the input as True if the nicking enzyme is present, as False if the nicking enzyme is absent. The output is defined as True if strong fluorescent signal is observed, as False if no strong fluorescent signal can be observed.

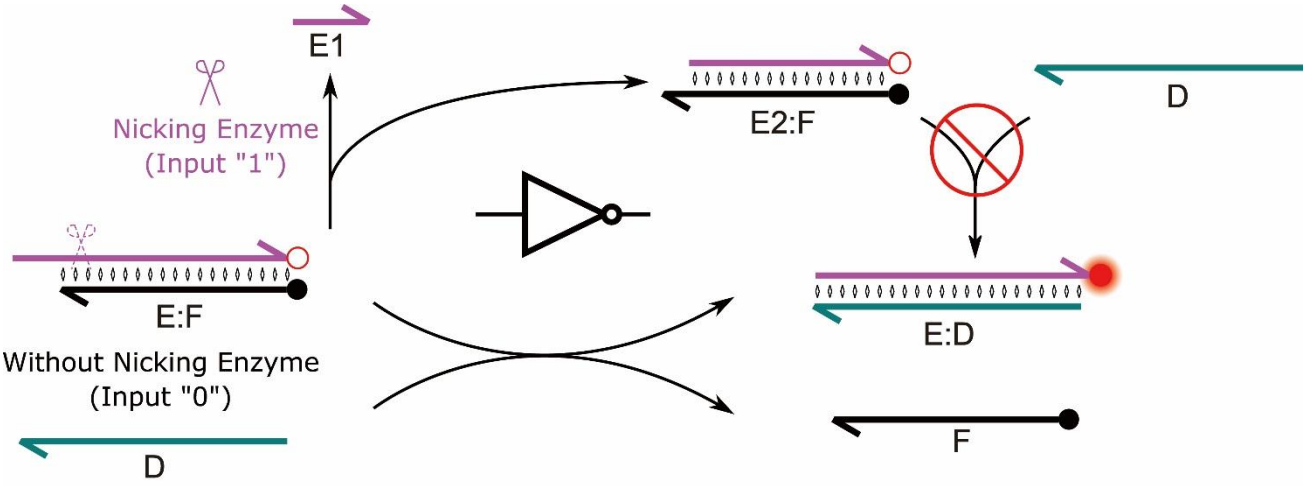

Figure S3. NOT gate. In the absence of the nicking enzyme, strand displacement between D and E:F occurs, leading to strong fluorescent signal. In the presence of the nicking enzyme, the toehold will be removed from E:F, thus no further reaction.

Based on Toehold Removal Mechanism (TRM), NOT logic can be readily constructed (**Figure S3**). In the absence of the nicking enzyme (input as False), quencher strand F in complex E:F would be freed through the strand displacement reaction, leading to the increase of the fluorescent signal (output as True). In the presence of the nicking enzyme (input as True), the toehold would be cut off via the cleavage by the nicking enzyme. Thus no further reaction occurs and no strong fluorescent signal can be observed (output as False).

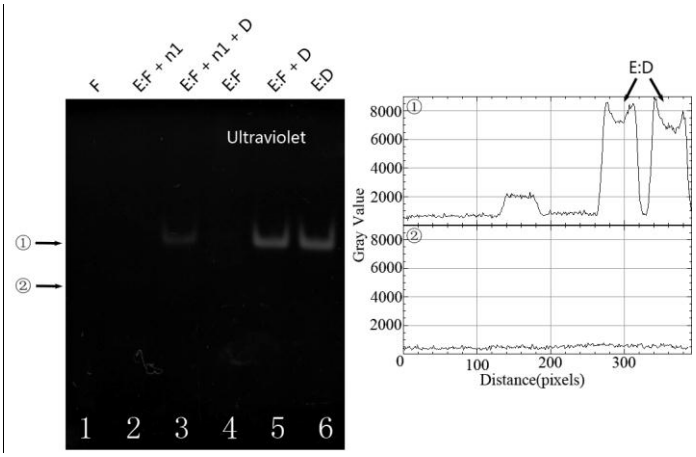

Figure S4. The grey intensity analysis of PAGE of NOT gate under ultraviolet. The right two height maps correspond to two transections ① and ② on the left image. Bright area corresponds to higher grey value. Compared

with lane 5, the addition of nicking enzyme n1 led to peak of grey value disappearing (lane 3).

Note S4: INHIBIT gate

Table S1. Truth table for INHIBIT gate.

Inhibit	Input	Output
0	0	0
0	1	1
1	0	0
1	1	0

INHIBIT gate contains a data input (Input in Table S1) and a control input (Inhibit in Table S1). The control input decides whether the gate is active. For example, if the control input is true, the gate is inactive (i.e. the output is always false). If the control input is false, the output is depended on the data input. INHIBIT gate can be constructed by cascading a NOT gate to one input of AND gate.

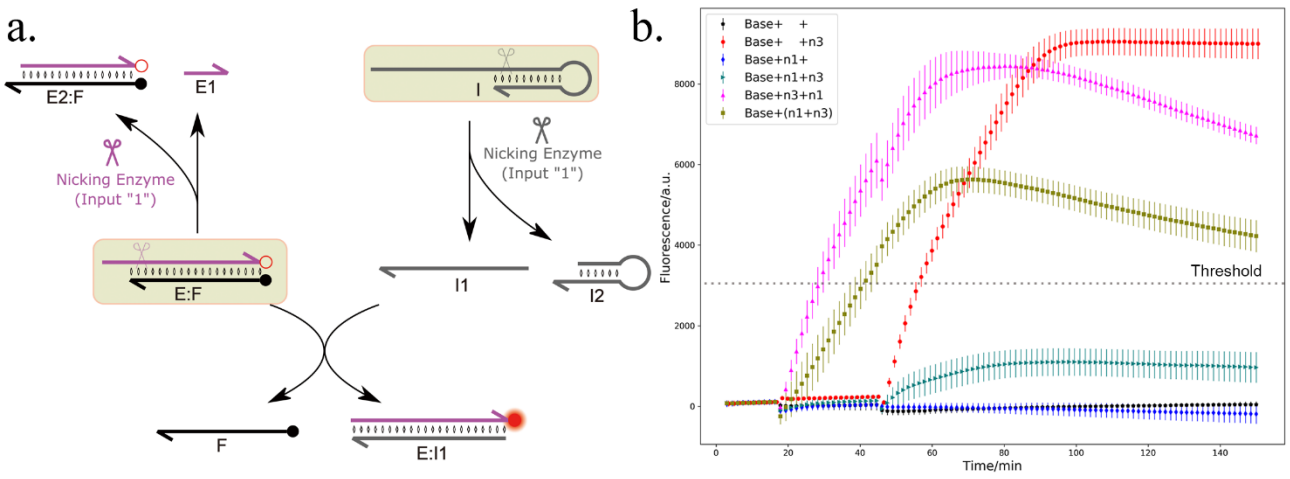

Figure S5. INHIBIT gate. (a) Reaction scheme of INHIBIT gate. Nicking enzyme n1 could inhibit strand displacement by removing the toehold of E:F, and nicking enzyme n3 could free a displacement strand I1 from hairpin I to trigger strand displacement if the toehold of E:F still exists. Only when n1 is absent and n3 is present could strand displacement take place, leading to strong fluorescence signal observed. (b) Kinetic characterization of INHIBIT gate. Only when n1 was absent and n3 was present could strand displacement take place yielding durable strong fluorescence signal (red dots). Other combination of inputs (black pentagons, blue diamonds and teal triangles) did not show significant fluorescence increase. Error bars represent one standard deviation from duplicate experiments.

The construction of INHIBIT gate is consisted by NOT gate and a DNA hairpin structure (**Figure S5a**). The addition of nicking enzyme n1 can remove the toehold that is necessary in the further strand displacement. Therefore, the strand displacement is inhibited after nicking digestion. Nevertheless, if n1 is not added, the addition of nicking enzyme n3 can cleave the hairpin and release a single strand that could trigger the strand displacement, leading to significant fluorescence signal observed. In the kinetic experiment (**Figure S5b**), only when n1 was absent and n3 was present, could strand displacement take place with the yield of durable fluorescence signal (red dots). Other combinations of inputs (black pentagons, blue diamonds and teal triangles) did not show significant fluorescence increase. As the reaction rate of strand displacement reaction is always faster than that of nicking enzyme cleavage reaction, when n1 and n3 are both added, strand displacement will still take place for n1 does not have enough time to eliminate all the toeholds (olive squares).

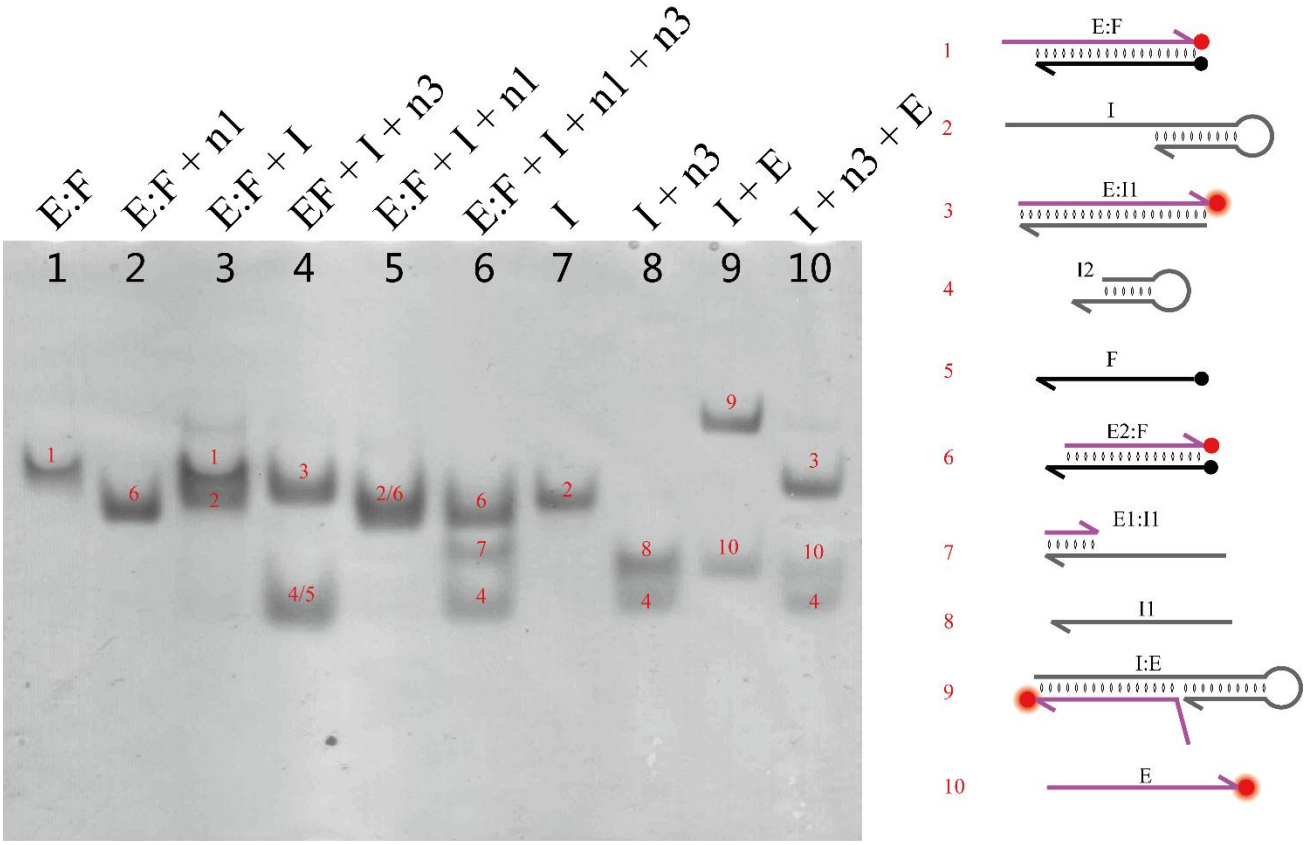

Figure S6. Nondenaturing gel electrophoresis of INHIBIT gate. True output (DNA structure 3) only existed in the presence of n3 and absence of n1 (lane 4). The reactions with other combinations of inputs (lane 3, 5 and 6) showed false outputs.

According to the gel electrophoresis in **Figure S6**, the toehold removal mechanism has a little

side effect. The toehold is only removed from the DNA duplex, not completely removed from the solution. So after the toehold was removed, it would cling to a strand with complement sequence to it (DNA structure 7), and this in rare cases may cause the formation of unwanted structures.

Note S5: Nicking enzyme

The nicking enzyme is a kind of restriction enzyme that only operates on one strand of a DNA duplex and introduces a nick. This property makes the nicking enzyme possible to regulate the DNA toehold in strand displacement.

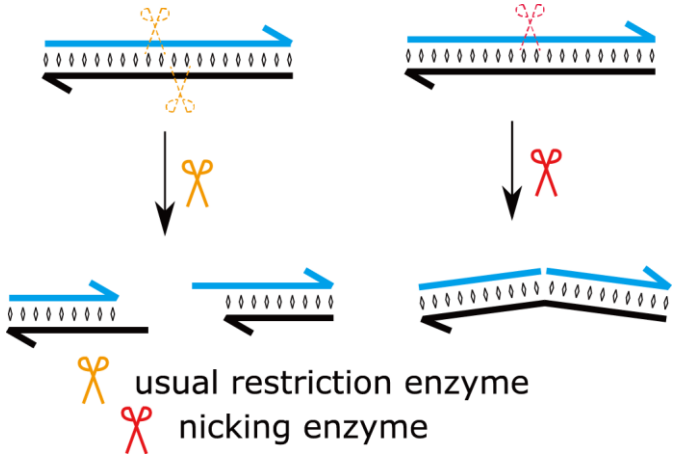

Figure S7. The characteristic of usual restriction enzyme and nicking enzyme.[1] Usual restriction enzymes will operate on both strands of the duplex and cleave the duplex into two parts (left). Each part is with a short single-stranded overhang (~3nt), and the sequence of the overhang relates to the recognition site of the enzyme. Unlike the usual restriction enzymes, nicking enzymes will operate on one strand of the duplex only. A nick will be introduced on the duplex with the opposite strand remaining unchanged.

Note S6: Concentration for TGM

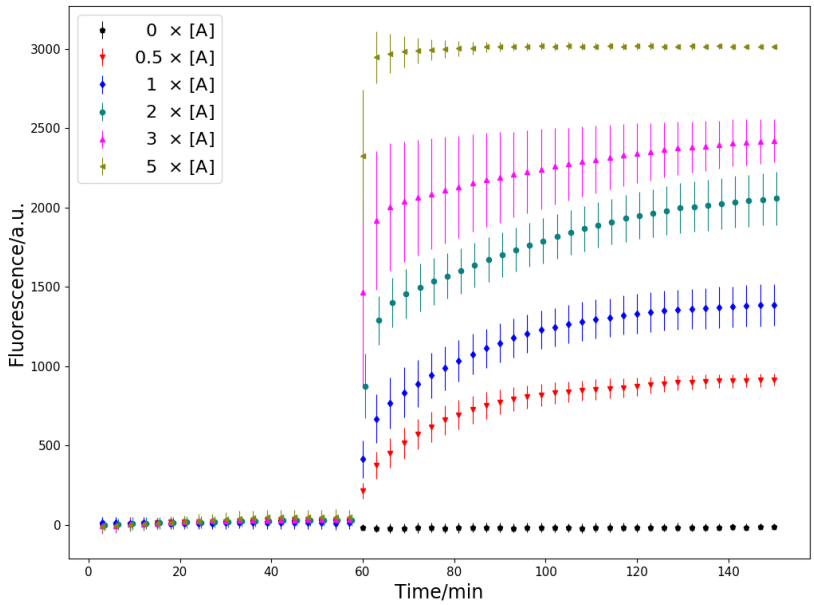

Figure S8. Fluorescence data for varying concentrations of displacement strand A. Displacement strand [A]: 400nM = 1x(10pmol at 25μL); substrate [B:C] = 400nM; nicking enzyme n1 = 10 units; 25°C. In order to get a high fluorescence curve with a relatively moderate slope (it would help to get more reaction details), 2 × [A] was selected for further experiment. Error bars represent one standard deviation from duplicate experiments.

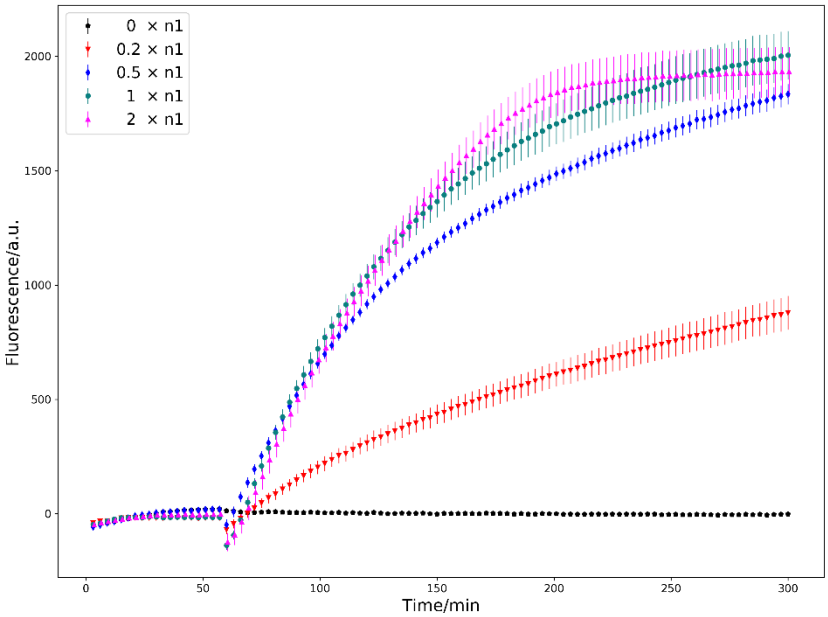

Figure S9. Fluorescence data for varying concentrations of nicking enzyme Nt.BbvCI. The amount of Nt.BbvCI: 0.5μL = 1× (5 units at 25μL); substrate [B:C] = 400nM; displacement strand [A] = 800nM; 25°C. 1× Nt.BbvCI shows an ideal performance. Error bars represent one standard deviation from duplicate experiments.

Note S7: YES gate

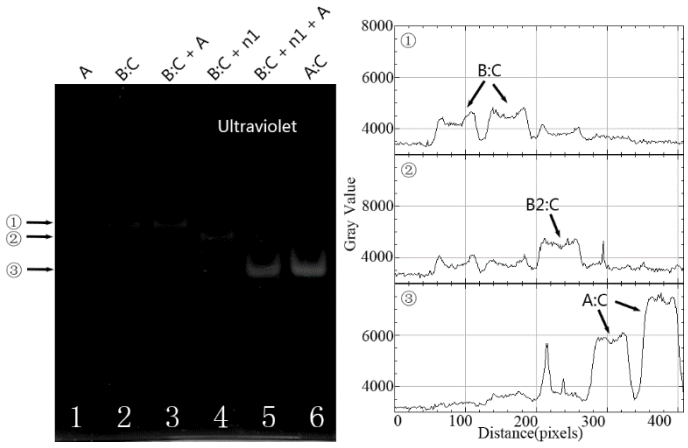

Figure S10. The grey intensity analysis of PAGE of YES gate under ultraviolet. The right three height maps correspond to three transections ①,② and ③ on the left image. Bright area corresponds to higher grey value. In the presence of displacement strand A, the addition of nicking enzyme n1 led to peak of grey value (lane 5).

Note S8: AND gate

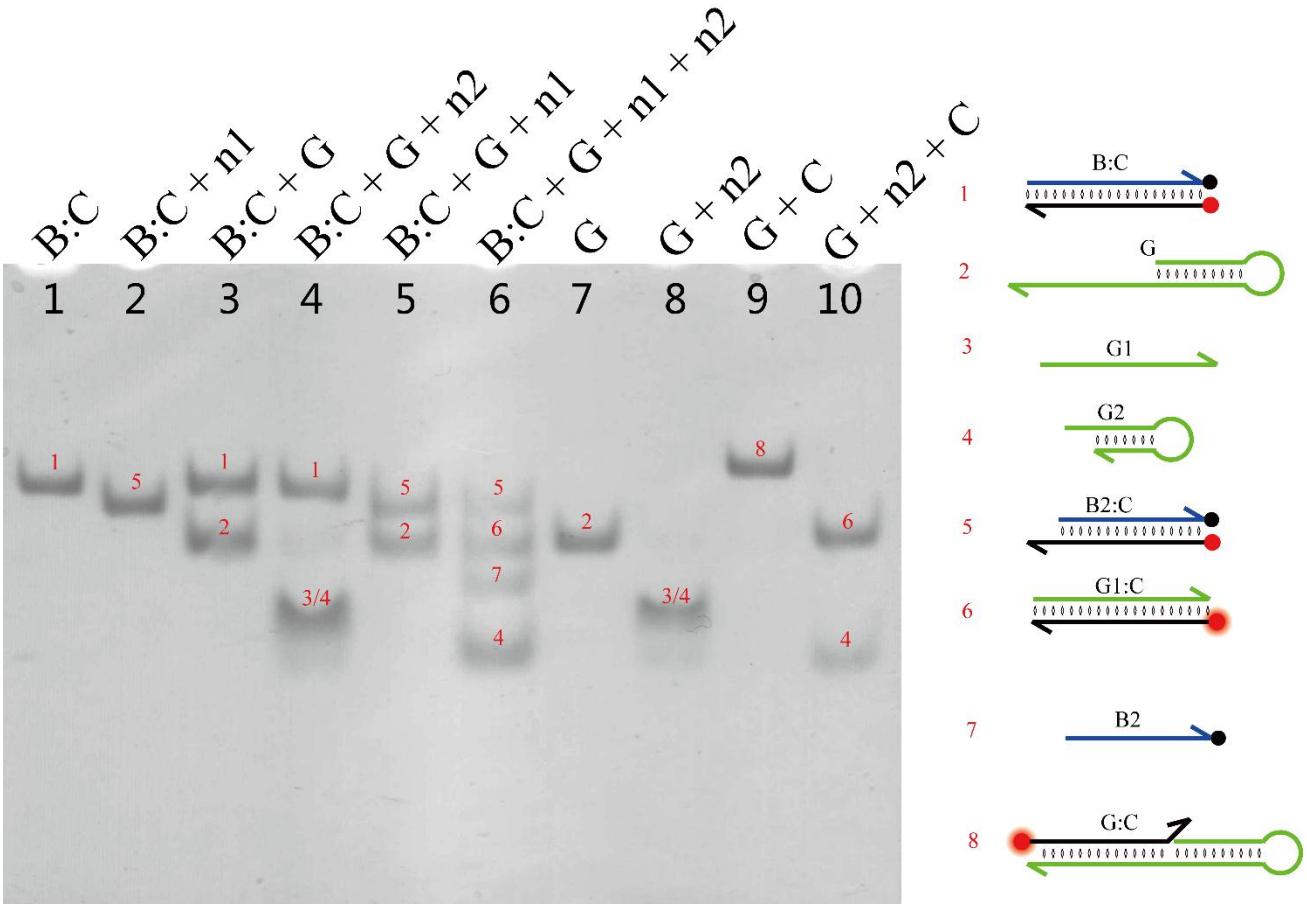

Figure S11. Nondenaturing gel electrophoresis of AND gate. True output (DNA structure 6) only existed in the presence of both inputs, nicking enzymes n1 and n2 (lane 6). The reactions with either input (nicking enzyme n1

or n2, lane 4 and lane 5) or no input (lane 3) showed false outputs.

Note S9: SWITCH logic circuit

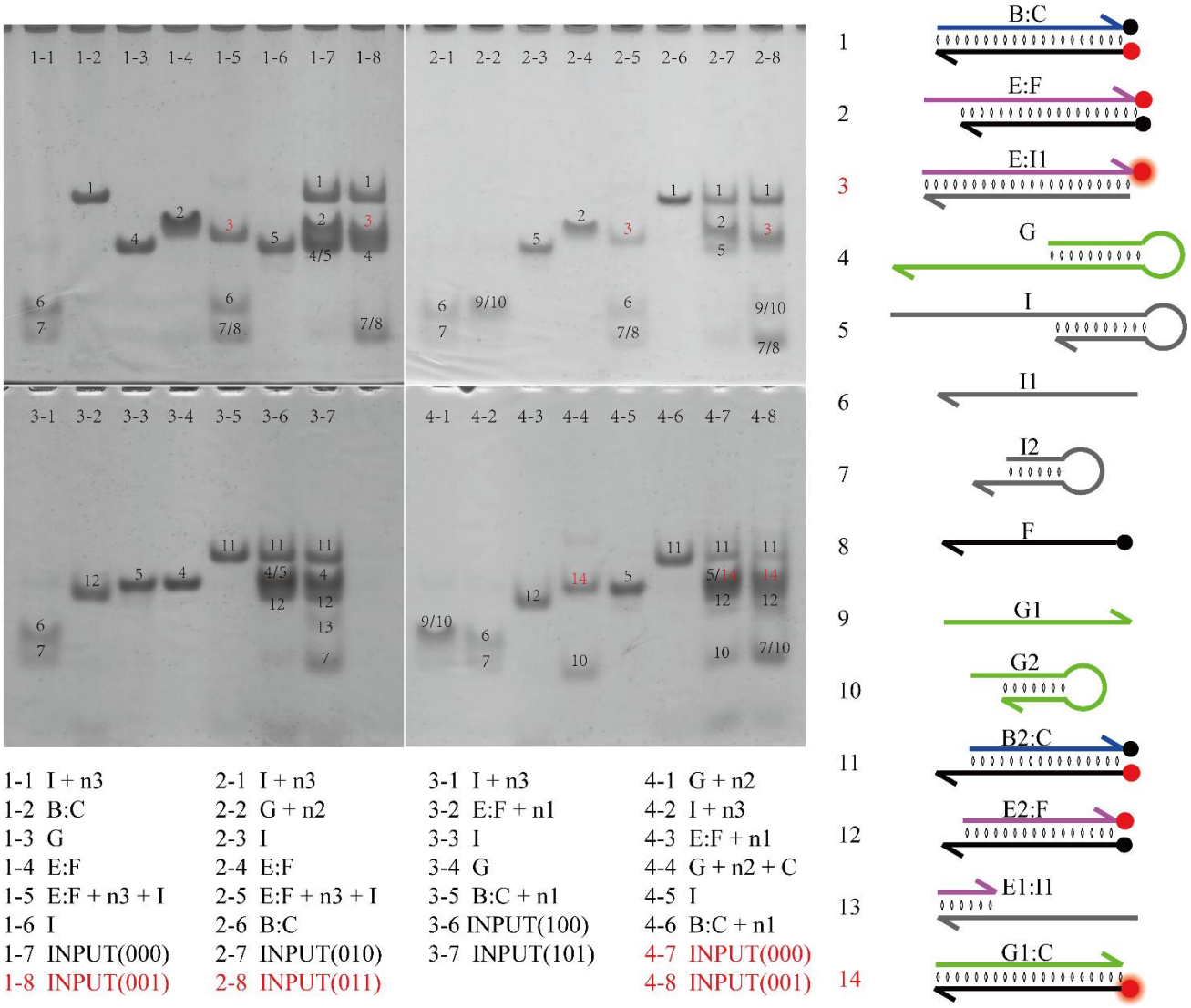

Figure S12. Nondenaturing gel electrophoresis of SWITCH logic circuit. The combination of the inputs was presented as three digits, $D_1D_2D_3$. $D_1D_2D_3$ stand for nicking enzymes n1, n2 and n3; 0 for being not added and 1 for being added. The 8 combinations of inputs were in lane 1-7, 1-8, 2-7, 2-8, 3-6, 3-7, 4-7 and 4-8. Red number means that this structure could emit fluorescence signal as true output. Red INPUT indicates that this combination of inputs will lead to true output. We can conclude that when n1 was not added, the output value was the same as whether n3 was added, and when n1 was added, the output value was the same as whether n2 was added.

Note S10: SWITCH logic statement

In computer programming languages, a switch statement is a type of selection control mechanism used to allow the value of a variable or expression to change the control flow of program execution via a

multiway branch. Switch statements exist in most high-level imperative programming languages such as Pascal, Ada, C/C++, C# and Java, and in many other types of language, using such keywords as switch, case, select or inspect. Switch statements come in two main variants: a structured switch, as in Pascal, which takes exactly one branch, and an unstructured switch, as in C, which functions as a type of goto. The main reasons for using a switch include improving clarity, by reducing otherwise repetitive coding, and (if the heuristics permit) also offering the potential for faster execution through easier compiler optimization in many cases.

A switch statement is often represented in the following form (**Figure S13**): an expression and a number of code blocks. According to the value of the expression, the program would proceed to a certain branch.

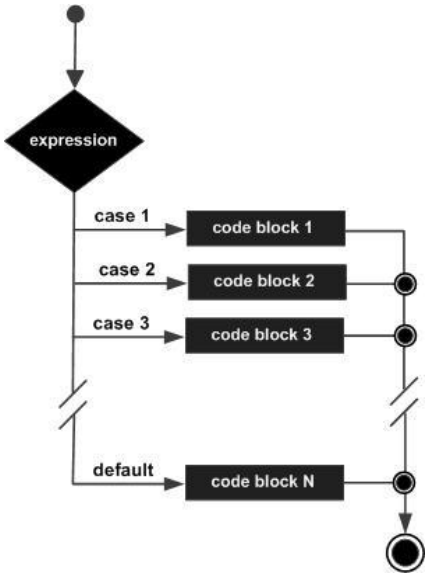

Figure S13. The control flow of switch statement. The value of the expression will be matched with the according case, then the codes for the case will be performed and proceed to the very end.

The SWITCH logic circuit constructed here could perform switch statement: three nicking enzymes are three inputs, and the expression is nicking enzyme n3; case 1 and case 2 correspond to true and false respectively; code block 1 is that the output is equal to n1; code block 2 is that the output is equal to n2. It could be represented as following (in the form of C language):

```
switch (n3) {
  case 1:   output = n1;     break;
  case 0:   output = n2;     break;
}
```

Thus, this construction for employing DNA circuit to operate computer program fragment is a beneficial attempt to realize molecular level programming.

Note S11: Sequences

In order to differentiate the different structures in gel analysis, multi-thymine (T) was employed in strand B and strand D. All the sequences were designed using Nupack to avoid unwanted structures.

Table S2. DNA sequences for YES, NOT, AND, INHIBIT and SWITCH (5' to 3').

A	CAGCCTCAGCAGTTGGATACATCTCAAGC
B	TTTTTTTCAGCCTCAGCAGTTGGATACATCTCAAGCTTTTTTTTTTTTTTT-BHQ
C	HEX-GCTTGAGATGTATCCAACTGCTGAGGCTG
D	TTTTTTTTTTTTATCCGTTCCTTGCAGTTGCTGAGGTGGCCAT
E	TTATGGCCACCTCAGCAACTGCAAGGAACGGATCA-FAM
F	BHQ-TGATCCGTTCCTTGCAGTTGCTGAGGT
G	GGCTGCGAGACTCGGTTTTCCGAGTCTCGCAGCCTCAGCAGTTGGATACATCTCAAGC
I	ATCCGTTCCTTGCAGTTGCTGAGGTGGCCATCACTGCTTGTTTTCAAGCAGTGATGGC

Table S3. Strands combination for YES, NOT, AND, INHIBIT and SWITCH circuit.

Circuit	Strands
YES	A, B, C
NOT	D, E, F
AND	B, C, G
INHIBIT	E, F, I
SWITCH	B, C, E, F, G, I

Table S4. DNA sequences for effect of length of toeholds on the performance of YES gate[a)].

T-2-A	CCTCAGCAGTTGGATACATCTCAAGC
T-2-B	TTTTTTTCCTCAGCAGTTGGATACATCTCAAGCTTTTTTTTTTTTTTT
T-2-C	GCTTGAGATGTATCCAACTGCTGAGG
T-3-A	GCCTCAGCAGTTGGATACATCTCAAGC
T-3-B	TTTTTTTGCCTCAGCAGTTGGATACATCTCAAGCTTTTTTTTTTTTTTT
T-3-C	GCTTGAGATGTATCCAACTGCTGAGGC
T-6-A	GCAGCCTCAGCAGTTGGATACATCTCAAGC
T-6-B	TTTTTTTGCAGCCTCAGCAGTTGGATACATCTCAAGCTTTTTTTTTTTTTTT
T-6-C	GCTTGAGATGTATCCAACTGCTGAGGCTGC
T-8-A	ATGCAGCCTCAGCAGTTGGATACATCTCAAGC
T-8-B	TTTTTTTATGCAGCCTCAGCAGTTGGATACATCTCAAGCTTTTTTTTTTTTTTT
T-8-C	GCTTGAGATGTATCCAACTGCTGAGGCTGCAT

[a)]T-5-A, T-5-B and T-5-C are the same sequence as A, B and C in **Supplementary Table s2**.

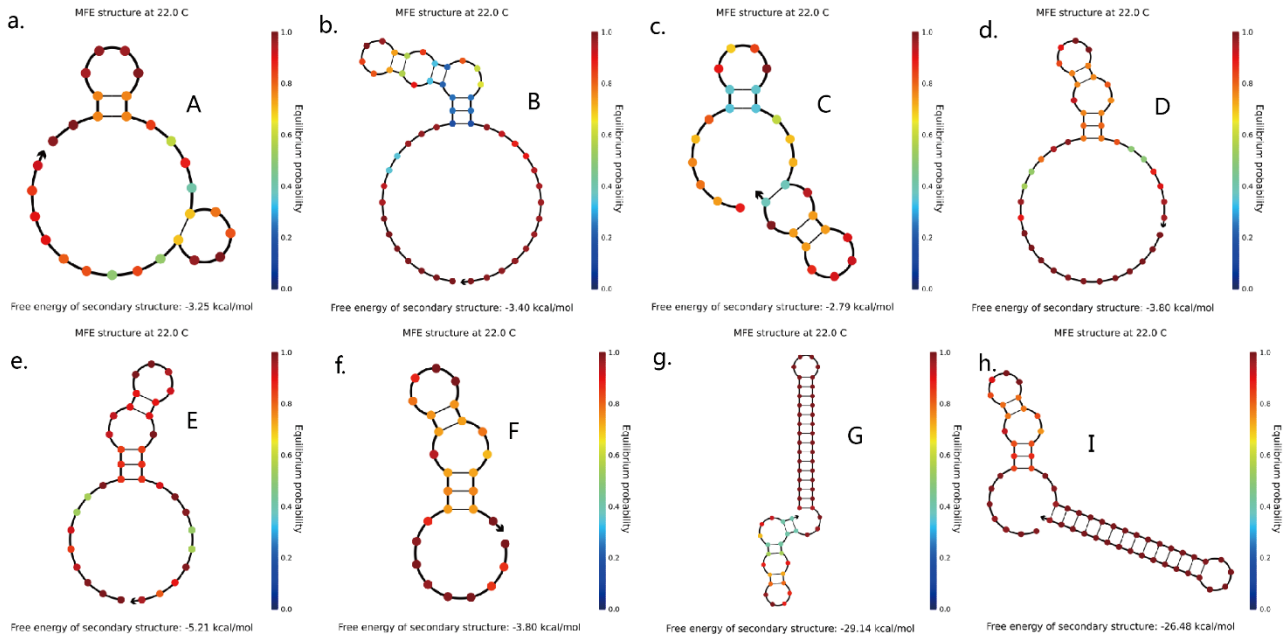

Figure S14. Nupack simulations for each strand in **Supplementary Table s2**. The unstable hairpin structures in a to f will not hinder the formation of DNA duplex such as B:C.

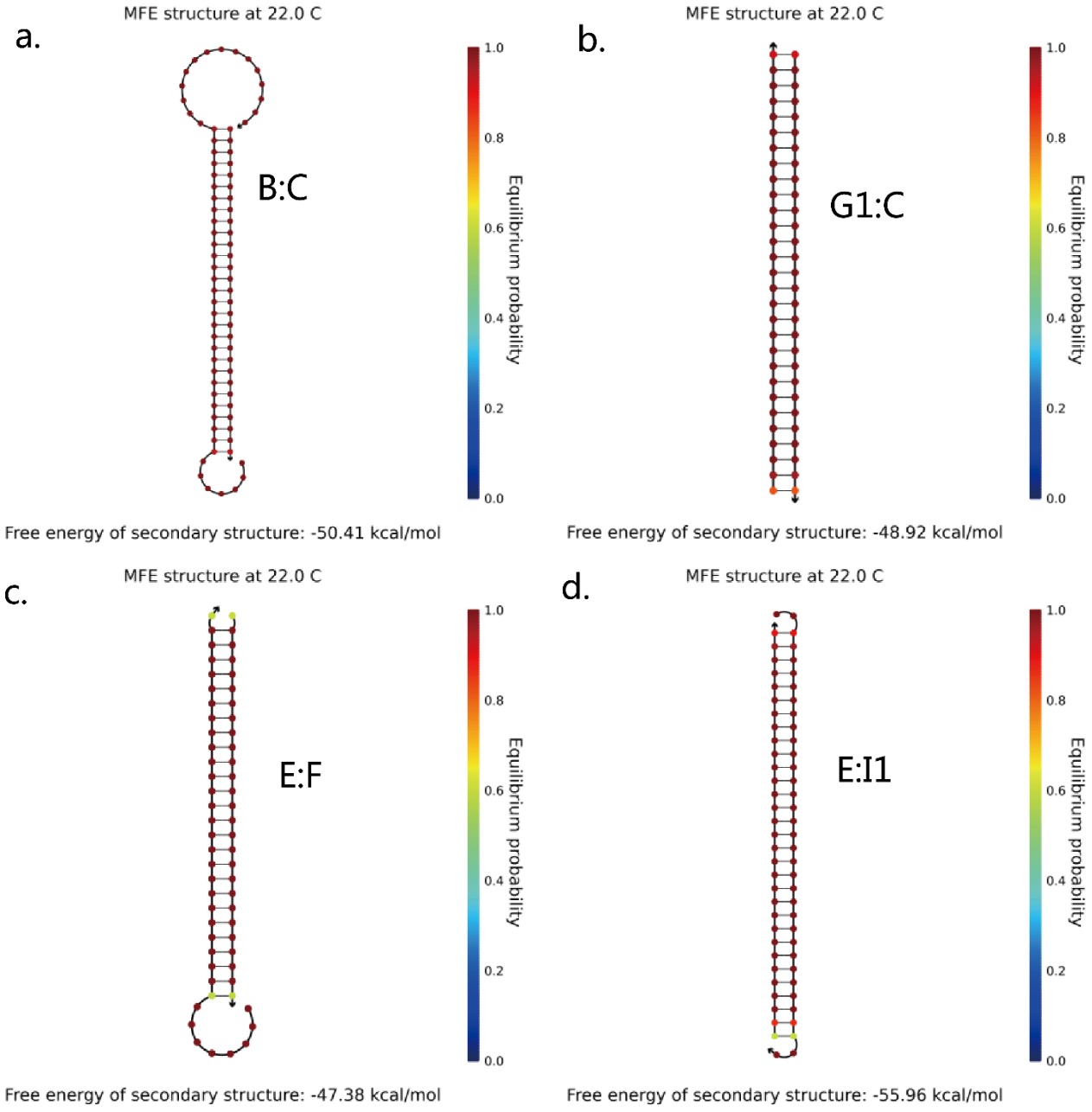

Figure S15. Nupack simulations for double-stranded structures used in YES, NOT, AND, INHIBIT and SWITCH circuits. The duplexes are stable without the cleavage of nicking enzymes.

Note S12: Python code for simulating reactions and fitting rate constants

The simulation was performed using python 3.6, and the three parameters kn, k1, k2 corresponded to r1.p0[0], r1.p0[1] and r1.p0[2], respectively.

```
import numpy as np

from scipy.optimize import leastsq
```

```
class Reaction_simu:
    def __init__(self, data):
        self.data = data
        self.p0 = [1e-7, 1.5, 1e+6]
        self.n1 = 2e+5
        self.ini = np.array([4e-7, 0, 0, 0, 0, 8e-7])

    def differ(self, kn, k1, k2, time=91, wreturn='ac'):
        bc = np.linspace(0,0,time)
        b1 = np.linspace(0,0,time)
        b2c = np.linspace(0,0,time)
        b1b2c = np.linspace(0,0,time)
        ac = np.linspace(0,0,time)
        a = np.linspace(0,0,time)

        sbc = np.linspace(0,0,time)
        sb1 = np.linspace(0,0,time)
        sb2c = np.linspace(0,0,time)
        sb1b2c = np.linspace(0,0,time)
        sac = np.linspace(0,0,time)
        sa = np.linspace(0,0,time)

        bc[0] = self.ini[0]
        b1[0] = self.ini[1]
        b2c[0] = self.ini[2]
        b1b2c[0] = self.ini[3]
        ac[0] = self.ini[4]
        a[0] = self.ini[5]

        for n in range(time-1):
```

```
            sbc[n] = -kn*bc[n]*self.n1
            sb1[n] = k1*b1b2c[n]
            sb2c[n] = k1*b1b2c[n] - k2*b2c[n]*a[n]
            sb1b2c[n] = kn*bc[n]*self.n1 - k1*b1b2c[n]
            sac[n] = k2*b2c[n]*a[n]
            sa[n] = -k2*b2c[n]*a[n]

            bc[n+1] = bc[n] + sbc[n]
            b1[n+1] = b1[n] + sb1[n]
            b2c[n+1] = b2c[n] + sb2c[n]
            b1b2c[n+1] = b1b2c[n] + sb1b2c[n]
            ac[n+1] = ac[n] + sac[n]
            a[n+1] = a[n] + sa[n]

        sbc[n] = -kn*bc[n]*self.n1
        sb1[n] = k1*b1b2c[n]
        sb2c[n] = k1*b1b2c[n] - k2*b2c[n]*a[n]
        sb1b2c[n] = kn*bc[n]*self.n1 - k1*b1b2c[n]
        sac[n] = k2*b2c[n]*a[n]
        sa[n] = -k2*b2c[n]*a[n]

        if wreturn == 'ac':
            return ac
        elif wreturn == 'bc':
            return bc
        elif wreturn == 'b1':
            return b1
        elif wreturn == 'b2c':
            return b2c
        elif wreturn == 'b1b2c':
```

```
            return b1b2c
        elif wreturn == 'a':
            return a
        elif wreturn == 'sac':
            return sac
        elif wreturn == 'sbc':
            return sbc
        elif wreturn == 'sb1':
            return sb1
        elif wreturn == 'sb2c':
            return sb2c
        elif wreturn == 'sb1b2c':
            return sb1b2c
        elif wreturn == 'sa':
            return sa

    def funcerror(self, p, y):
        return y - self.differ(p[0], p[1], p[2])

    def get_p(self):
        self.p0 = leastsq(self.funcerror, self.p0, args=(self.data))[0]
        return self.p0

r1 = Reaction_simu(np.load('data.npy'))
print(r1.get_p())
```

Reference

(1) Heiter, D. F.; Lunnen, K. D.; Wilson, G. G. *JOURNAL OF MOLECULAR BIOLOGY* **2005**, *348*, 631-640.

(2) Kotani, S.; Hughes, W. L. *JOURNAL OF THE AMERICAN CHEMICAL SOCIETY* **2017**, *139*, 6363-6368.