IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 27, 2020, accepted June 13, 2020, date of publication June 19, 2020, date of current version July 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3003636

DNA Logic Circuits Based on Accurate Step Function Gate

CONGZHOU CHEN[1], WEI XIAO[2], JIWEI ZHAO[3], ZHENG ZHANG[3], AND XIAOLONG SHI[2,3]

[1]School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China
[2]Institute of Computing Science and Technology, Guangzhou University, Guangzhou 510006, China
[3]School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China

Corresponding authors: Xiaolong Shi (xlshi@gzhu.edu.cn) and Zheng Zhang (leaf@mail.hust.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFA0706402, and in part by the National Natural Science Foundation of China under Grant 61632002, Grant 61772376, and Grant 61672248.

ABSTRACT DNA molecular as the material for logical computing and information storage has been widely applied. An important mechanism for DNA logical computing is the toehold-mediated DNA Strand Displacement (DSD). Researchers have used the DSD mechanism and designed various DNA circuits to perform logical computing processes. However, the logy reaction speed in some logic gates limited the speed of response, the output response is not sensitive enough. Therefore, no output response is often misjudged to logic ''0'', high output response is not accurate to logic ''1''. In this paper, we proposed a highly creditable step function gate, which consists of a step function module and a threshold module. The output signal in this proposed gate is highly sensitive and accurate to logic ''0'' & ''1''. The proposed gate can recognize the difference between ''no input'' and ''low input''. Besides, we built the OR, AND, NOT, XOR circuits by the proposed step function gate, and the simulation results are highly supported.

INDEX TERMS DNA strand displacement, step function gate, logic circuits, DNA computing.

I. INTRODUCTION

DNA as the carrier of heredity information can be used as nanomaterial [1]. It is natural and biocompatible; the four types of canonical bases can be coded for programing. Researchers have used DNA and fulfilled various applications. Such as, DNA nano architectures [2]–[5], information storage [6], [7], logic computing [8]–[10], etc.

Recently, programmable DNA computing is thriving. Notably, the toehold-mediated DNA Strand Displacement (DSD) plays a vital role in DNA computing. DSD is the Enzyme-free tech, with toehold and immigration domains [11]–[13]. It can be used in the second general Gene test [14], [15], intelligent molecular detection [16], [17], etc. Currently, some DNA processing systems have been generated by using DSD technology, such as acid logic circuits [18], [19], evolvable molecular machines [20], Hopfield network with automatic associative memory calculation [21], etc.

DSD based logic circuits is a promising research area for constructing nanocircuits. Researchers have proposed various methods to construct the circuits with stable output.

The associate editor coordinating the review of this manuscript and approving it for publication was Navanietha Krishnaraj Rathinam.

Many DNA logic gates have been proposed [22]–[25]. Fan *et al.* constructed the DSD based digital switching circuits, and completed a full-adder logic. Wang *et al.* [26] designed a poll with three voters via a four-way junction. Zhu *et al.* [27] designed three inputs two-layer DNA multiplexer. Zhu *et al.* [28] constructed a novel Bio-sensor based on the DSD system [29].

A notable work is the Seesaw gate [30], it is modularized and functionally robust. Each logic gate functions as a seesaw node. Those nodes can be cascaded and formed the multi-layers circuits.

However, the logic outputs in the above-mentioned methods are not accurate. The concentration of report strands (or final output logics) is negative correlation with the number of layers. Besides, the signal response (reaction speed) is not highly sensitive. Because the larger the DSD based circuits, the more reaction paths it will take. Then, the efficiency of displacement decreases, the signal response rusted. We compared the DSD simulation results of several logic circuits at their best performance, which gives a direct sense of the problems. (Table 1).

The best performance is the highest or the lowest concentration of output strands in each method (Table 1), we can

TABLE 1. The Logic output "1/0" in each method.

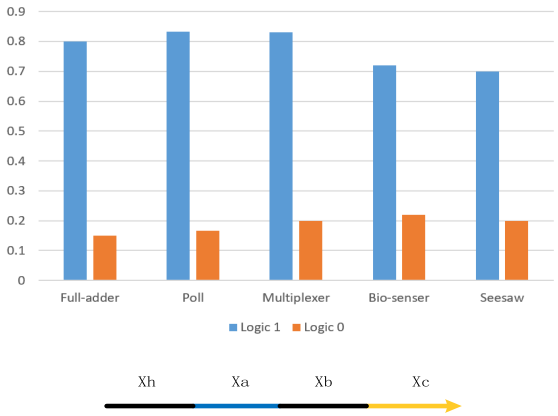


FIGURE 1. Four-domain chain.

find that the highest concentration for logic "1" is around 0.7∼0.8 (normalized); the lowest concentration for logic "0" is around 0.15∼0.2.

Moreover, the signal sensitivity in the above-mentioned methods is dull. We take the note in the Seesaw gate for a more specific example. First, when the threshold is 0.5×, the input is 0.6×, then the output signal of the node should be 1×. But in fact, the steady output signal is 0.5×. Second, when the input strand concentration is around 0< x <0.2× (low input), the output signal is not 0. The Seesaw gate cannot distinguish the difference between low input "0" and no input. Therefore, the NOT gate constructed by the seesaw gate is not accurate.

In this paper, we proposed an accurate step function gate with high sensitivity, the final output strands are accurate to be "0" or "1". The proposed logic gate can distinguish the difference between low input and no input. Besides, the reaction speed is fast and can be controlled. The designing of the Step function gate is described in the following text.

II. STEP FUNCTION GATE

A. INPUT STRAND

We used the "four-domain" strand to encode the input strand [29]. As shown in Figure 1, the four-domain chain X composes of four parts: the history domain Xh, the recognition domain Xb, the two toehold domains Xa and Xc. The history domain (Xh) is used for indicating the previous generation of chains. Assuming a chain is generated by multiple reactions. Their historical domains are different, but their identification domains are the same, then we consider those chains belong to the same input signal.

The step function gate contains two modules: the step function module and the threshold module. Both modules use the input chain X as the input signal. By controlling the reaction rate between the two modules. The threshold module and step module can be integrated.

B. THE STEP FUNCTION MODULE

We define the initial concentration of chain X as $[X]_0$, the steady-state concentration of chain X is $[X]_\infty$(Chain X can be produced from the step function module).

$$[X]_\infty = \int_0^{+\infty} X(t)\,dt \tag{1}$$

The definition of the step function is:

$$[Y]_\infty = \begin{cases} [G]_0, & [X]_0 > 0 \\ 0, & otherwise \end{cases} \tag{2}$$

The initial chain X as the input signal determines the existence or absence of the step signal, G is the gain signal which determines the amplitude of the step signal. The step signal module contains four reactions (As shown in formula 3). X is the input signal chain, G is the gain signal chain, S1, S2, and S3 are auxiliary chains, I1, I2, and B are intermediate products, and Ø represents no signal chain, null.

$$S1 + G \leftrightarrow I1 + B \tag{3a}$$
$$S2 + B \rightarrow \emptyset \tag{3b}$$
$$X + I1 \rightarrow I2 \tag{3c}$$
$$S3 + I2 \rightarrow X + Y \tag{3d}$$

The auxiliary chain S1 and the gain signal chain G products I1 and B through a reversible chain displacement reaction. To push forward the reaction (3a), S2 will be irreversibly consumed by the intermediate product B. Subsequently, the input signal X reacts with I1, produces chain I2. Finally, chain I2 reacts with S3 and produces chain Y. Since X is cyclic reacted in the system. If X is present, it continuously produces chain Y, until G is consumed. To maintain those reactions, S1, S2, and S3 are oversupplied. The reactions in the step function module are shown in Fig.1.

C. THE THRESHOLD MODULE

We designed the threshold strand T, T can annihilate input signal X with the help of auxiliary strand A1 and A2, like a seesaw node (formula 4a). The annihilate reactions are as follows:

$$A1 + T \leftrightarrow I1 + B \tag{4a}$$
$$A2 + B \rightarrow \emptyset \tag{4b}$$
$$X + I1 \rightarrow \emptyset \tag{4c}$$

The annihilation method is not unique. Soloveichik *et al.* used the bimolecular reaction method [31], B Yordanov and Lakin MR implemented the two-domain and the three-domain annihilation, respectively [32], [33]. In addition, Soloveichik *et al.* proposed the concept of replace ratio, and Yordanov *et al.* defined the ration as complementarity [31], [33], [34].

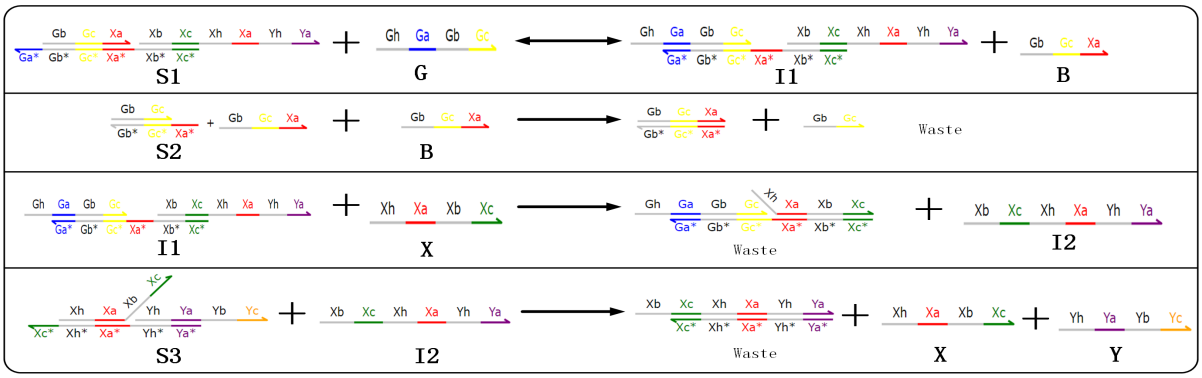


FIGURE 2. The reactions of the step function module.

The chain displacement speed is between $10^{-7}\text{nM}^{-1}\text{S}^{-1}$ and $10^{-3}\text{nM}^{-1}\text{S}^{-1}$ [35]. Assuming that the full domain replacement speed of X is q_{max} then x^*c is the replace speed with replace ratio c (c is between 0 and 1).

In the seesaw gate, the reaction between input and the logic gate is reversible. The reaction between input and threshold gate is irreversible. The irreversible reaction has a higher reaction speed. Input strands inclined to react with the threshold gate. When the threshold gates are all consumed, further reactions will produce output strands. Therefore, the step function works. We generate the step function gate by adjusting the replace ratio of chain X, and fulfilled the highly sensitive and accurate signal response. The reaction network is shown in Fig.2. The auxiliary strand S1, S2, and S3 are all oversupplied.

D. RESULTS AND ANALYSIS

We used the Visual DSD software to simulate the Step functions gate and the Seesaw gate. The four-domain input chains are built based on the toehold-mediated DND strands. The unit concentration of every strand in each group is 1×10^{-7}nM. We adjusted the concentration of input strands from 0.1 to 1.0×, every 0.1× per step. The concentration of threshold, output strands, fuel strands in the Seesaw gate is 0.5×, 1×, 2×, separately. The speed of forwarding reaction is $0.002nM^{-1}s^{-1}$, the speed of reverse reaction is $0.0013nM^{-1}s^{-1}$. The threshold in the Step function is 0.5×, and auxiliary strands are all 100×.

Figure. 3(a) and 3(b) show the results of the step-signal response. The dashed line shows the response of low input (input concentration is lower than threshold). The full line is the response of high input (inputs concentration is higher than threshold).

In the seesaw gate, the concentration of output response between 0∼0.3 is the low output, between 0.7∼1.0, is the high output. When the concentration of the input strand is between 0.1∼0.4, they quickly consumed. When the input strand raised to 0.5 (equal to the threshold gate), the output is enhanced, but not obvious. (Fig 3.a). When the input strand far overweighs the threshold gate, the output responds quickly reach to a balance with high concentration.

The lower input responses (in the seesaw gate or the proposed step function gate) are almost the same. However, the high input responses are different, the output responses in the proposed methods are all 1×. The different responses of low/high input are obvious. The output responses dispersed at two poles, away from the middle baseline (0.5× threshold).

Figure 4(a) and 4(b) shows the output curve (x-axis is the concentration of input strand) of the Seesaw gate and the Step function gate, separately. We found that: First, the input chain at the range of 0.1 ∼ 0.4 produces low output in the Seesaw gate, and the output signal slowly gains with the increase of the input signal. Second, when the input is around the threshold, the gradient gain is lower, and the mutation is smooth. Third, when the input excesses the threshold, the output cannot be stabilized, still increases with the input signal.

In contrast, the results of the step function gate are highly credible. The output responses of low inputs are all 0, the output responses of high input are all 1×. (Fig 4.b) The gradient gain is high, which means the response around the threshold is sensitive.

Table 1 and Table 2 list the steady-state output of the seesaw gate and step function gate in low input and high input situations, separately. When the input is less than or equal to the threshold: the average steady output of the seesaw door is 0.0816, the variance is 0.0104; the average steady output of the step function gate is 0.0074, the variance is 1.256×10^{-4}. The output value of the Step function reduced by 91.3%, the variance drops 98.79%, compared with the Seesaw gate. When the input is higher than the threshold: the average steady output of the Seesaw gate is 0.6964, the variance is 2.124×10^{-4}; the average steady output of the step function gate is 0.9995, and the variance is 1.214×10^{-6}. The mean value increased by 30.33%, and the variance reduced by 99.43%. Our proposed method significantly improves the accuracy.

III. LOGIC AND/OR/NOR/XOR GATES

We set the concentration of input strands at 0.1× as the lowest signal, the input concentration 0.9× as the highest signal. The main idea to construct those gates by the step function is

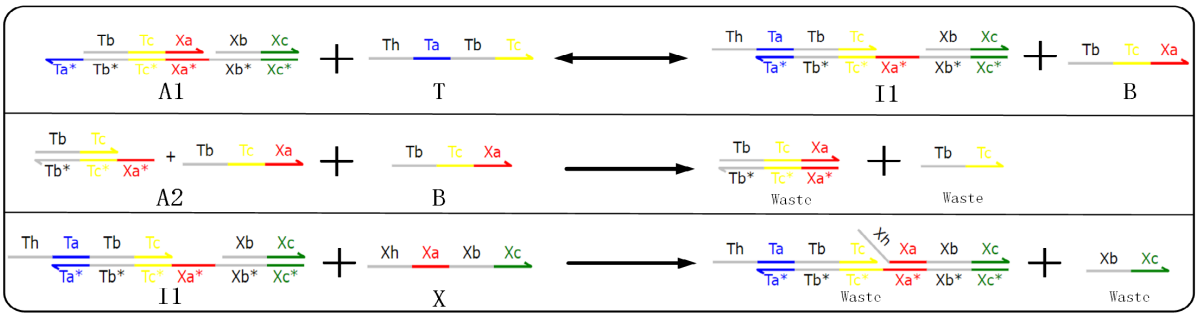


FIGURE 3. The reactions of the threshold module.

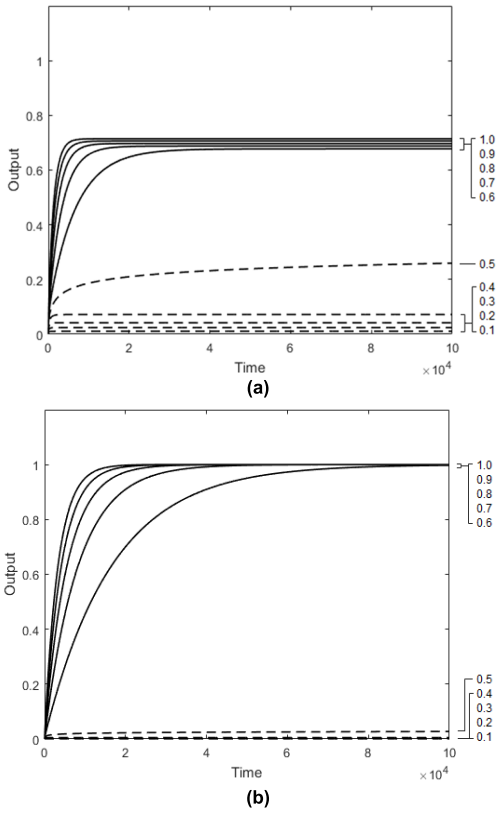


FIGURE 4. (a) Signal response of the Seesaw gate. (b) Signal-response of the step function gate.

TABLE 2. Response of low input.

Input	0.1	0.2	0.3	0.4	0.5
Seesaw gate	0.0105	0.0238	0.0420	0.0718	0.2597
Step function	6.6923×10^{-4}	0.0015	0.0027	0.0048	0.0273

converting multiple input chains into different signals. Then transferring those different signals to generate the low bit or high bit signals.

The signal molecular reactions are defined as following [21]:

$$\mathrm{U1} + \mathrm{X} \rightarrow \mathrm{I} \tag{5a}$$

$$\mathrm{U2} + \mathrm{I} \rightarrow \mathrm{Y1} + \mathrm{Y2} \tag{5b}$$

TABLE 3. Response of high input.

Input	0.6	0.7	0.8	0.9	1.0
Seesaw gate	0.6774	0.6875	0.6970	0.7059	0.7143
Step function	0.9975	1.0000	1.0000	1.0000	1.0000

First, input strand X will react with U1 and produce I, and then I will react with U2 and produce Y1 and Y2. Those two reactions are integrated as:

$$\mathrm{X} + \mathrm{U1} + \mathrm{U2} \rightarrow \mathrm{Y1} + \mathrm{Y2} \tag{5c}$$

A. AND/OR GATES

Figure 5(a) is the flow chart of AND/OR gate. We set the threshold gate at 1.2× (t=1.2) to form the AND gate, and adjust the threshold gate to 0.4× (t=0.4) to form the OR gate.

The reactions: X1 → T1 + T2 and X2 → T1 + T2 produce T1 and T2. Either existence of T1 or T2, imply that the signal molecular reacted. T1 transfers to the step function gate (threshold =0; gain=0.1) and then produce the output strand (0.1 Y). T2 transfers to another step function gate (threshold =1.2; gain=0.8), when the input strands X2 is over 1.2, the output Y is "1" (0.1Y+0.8Y). In contrast, if t=0.4, either X1 or X2 is high input, the output is Y is "1".

We used the Visual DSD software to simulate the AND/OR gates. The forward reaction speed is $1.3\mathrm{nM}^{-1}\mathrm{s}^{-1}$, bind ratio is 0.003. Each logic gate has three types of input signals "00", "01" and "11", respectively. Those three types of inputs represent the low-level inputs, a low input & a high input, and high-level inputs.

As shown in Fig. 5(b), the AND gate, when the inputs are "00" and "01" the concentration of output is 0.1×. When the input is "11", the step function AND gate produce 0.9× high output signal. Fig. 5(c), the OR gate, when input strands are "01" or "11" the output strand concentration is 0.9×. The trajectory line of "11" is slower than the others, but the final output is also 0.9×.

B. NOT GATES

The traditional logic gate used the presence or absence of the output strand to indicate "1" or "0". For example, the seesaw gate regards the output concentration in the range of 0 ∼ 0.3

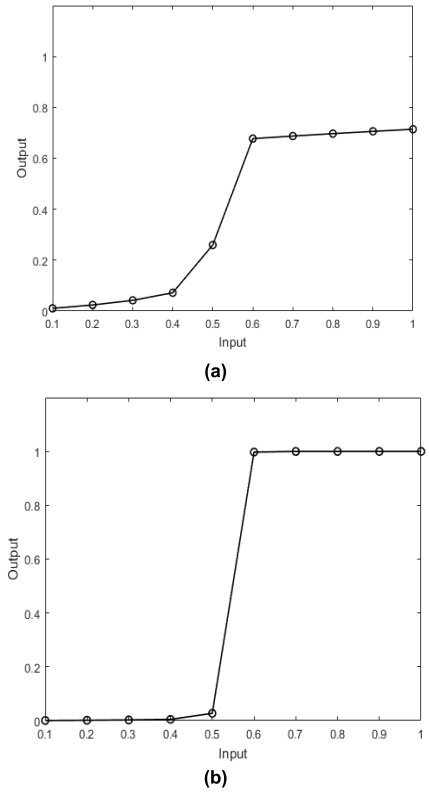


FIGURE 5. (a) Output curve of the Seesaw gate. (b) Output curve of the step function gate.

TABLE 4. NOR Gate represented by dual-rail logic.

x_1^0	x_1^1	x_2^0	x_2^1	y_1^0	y_1^1
1	0	1	0	0	1
1	0	0	1	1	0
0	1	1	0	1	0
0	1	0	1	1	0

as output "0". Output concentration in the range of $0.7 \sim 1$ is "1". But, the absence of the output strand (or input strand) is misrecognized as output (or input) "0". Also, the definition of "1" is not precise. Consider those short comes. Usually, the dual-rail logic method was utilized to design the NOR gate.

The dual-rail logic method uses a pair of input stands to represent logic 1 and logic 0. Taking the NOR gate for example, the truth value X is dived into X1 and X2, which are dual values. If x_1^0 is "1", x_1^1 is "0", then x1 is logic "0"; If x_2^0 is "1", x_2^1 is "0", then x2 is logic "0". As a result, the output y is logic "1" (y_1^1 is "1", y_1^0 is "0"). The truth table of the NOR gate is illustrated in Table 3.

However, the disadvantage of dual-rail logic is obvious, it cannot represent the NOT gate, directly. Whereas, we can clearly clarify the low input, high input, and no input. Therefore. We can design the NOT gate, directly.

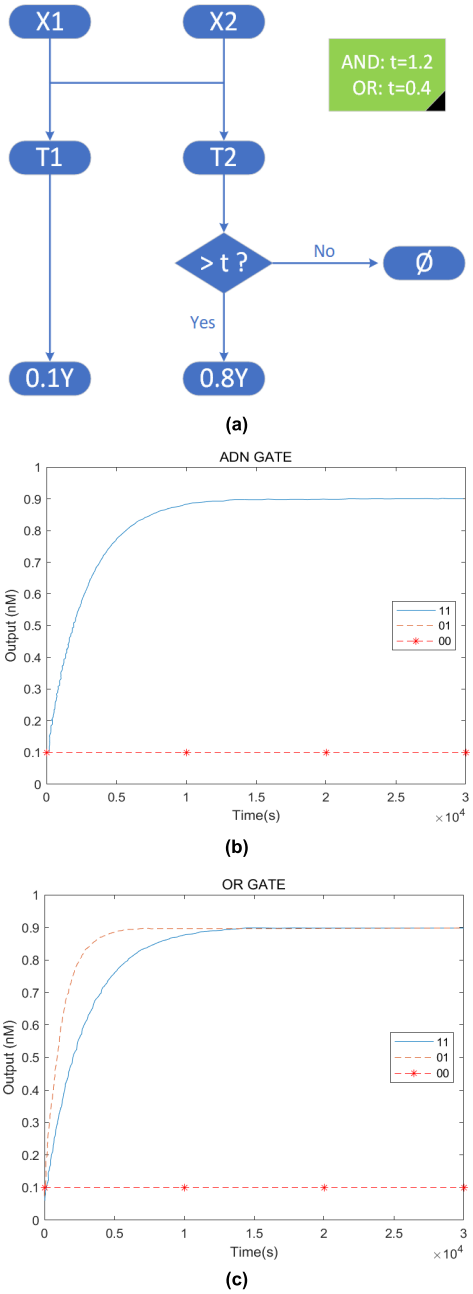
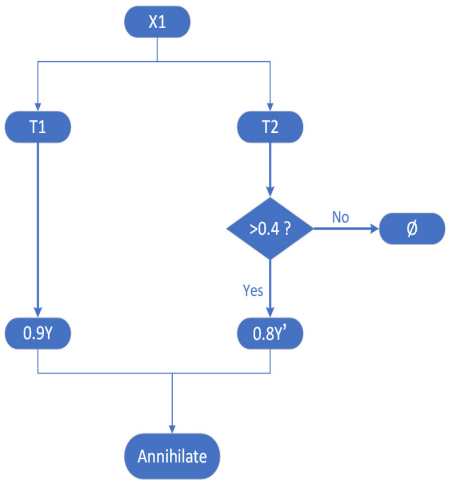
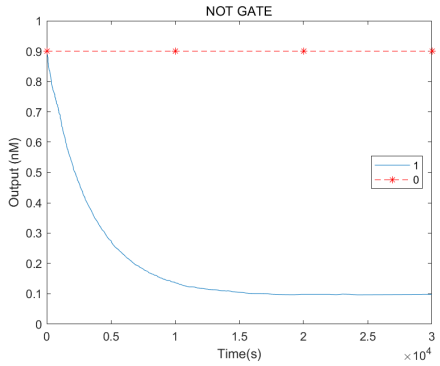


FIGURE 6. (a) The flow chart of AND/OR gate. (b) Simulation results of AND gate. (c) Simulation results of OR gate.

The flow chart of the NOT gate is shown in Fig. 6 (a). First, the input chain X transfers to the single-molecule reaction $X \rightarrow T1 + T2$. When the input chain is low signal, (input concentration is 0.1). Then, the output concentration is determined by the step signal gate with the threshold gate 0 and 0.9 $\times$ gain (0.1 to 0.9). When the input chain is a high-level signal (input concentration is 0.9), the step signal gate with threshold gate 0.4, produced high output Y (Y is 0.8). Then, auxiliary strand Y' will consume Y, haul back the output strand to low strand concentration 0.1 (0.9 to 0.1).

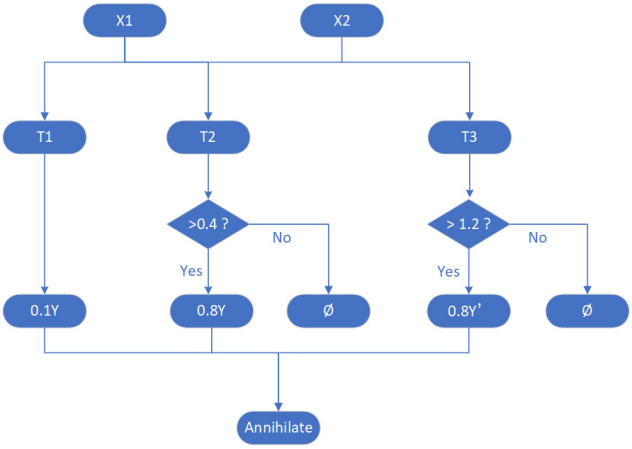


(a)

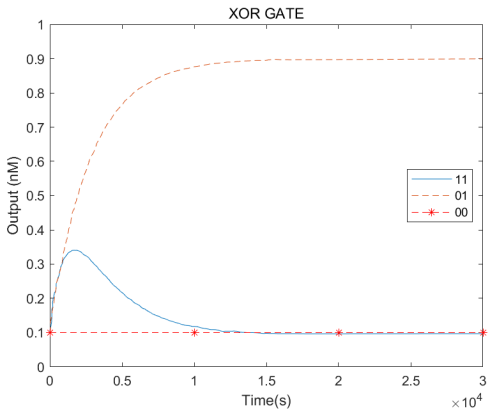


(b)

FIGURE 7. (a) The flow chart of NOT gate. (b) The simulation results of NOT gate.



(a)



(b)

FIGURE 8. (a) The flow chart of XOR gate. (b) The flow chart of XOR gate.

Fig. 6(b) is the results of step function NOT gate, which reverse the input signals.

C. XOR GATES

Generally, the XOR gate consists of multiple AND, OR, AND gates. Here, we used the dual-rail method to generate the XOR gate, directly. Therefore, the number of required gates is reduced. The flow chart of the XOR gate is shown in Fig. 7.

As shown in Fig. 7(a), X1 and X2 produced T1, T2, T3 via single molecular reactions:X1→T1+T2+T3, X2→T1+T2+T3. When X1 and X2 are low inputs, the output is 0.1Y. When X1 or X2 is high input, the output is a high signal. (0.1Y+0.8Y, or 0.8Y'-0.1Y). When X1 and X2 are both high inputs, Y and Y' strands will be quenched: $Y + Y' \rightarrow \emptyset$ (0.8Y-0.8 Y'), the output is a low signal.

Fig. 7(b) is the XOR gate. When the input is "00" or "11", the final output strand is 0.1×. When the input is "01", the output strand is 0.9×. In the "11" trajectory, the output strands raise to 0.3× and then drop to 0.1×. This is the effect of the two threshold gates (t=0.4 and t=1.2), Y is produced at the beginning (raise to 0.3×), then consumed by its paired strand Y'.

IV. CONCLUSION

In this paper, we designed a step function gate, and used this gate to construct the AND, OR, NOT, XOR logic circuits. We used the four-domain strand as the input signal, and adjusted the reaction speed and thresholds to control the reaction orders, so that the accurate step function gate can be achieved.

Visual DSD simulation results show that when the input is less than or equal to the threshold, the average steady-state output is 0.0074, the variance is 1.256×10^{-4}, which is 91.3% lower than that of the seesaw gate. When the input is over the threshold, the average steady-state output is 0.9995, the variance is 1.214×10^{-6}, the mean is increased by 30.33%, and the variance is reduced by 99.43%. Since the reaction sequence is controlled by adjusting the concentration of auxiliary strands. Leakage is possible in this system.

In the proposed method, we can clearly identify the high output (>0.9), low outputs (<0.1), and no output (zero). Therefore, the proposed gate clarifies the tangled output signals, and can be utilized to form the NOT gate directly. In addition, the saltation around the threshold is rapid, the output is stately. Based on those characterizations, the step function gate can be used to assemble the XNOR gate, tri-states gates, and complex programable DNA circuits.

ACKNOWLEDGMENT

(Congzhou Chen, Wei Xiao, and Jiwei Zhao contributed equally to this work.)

REFERENCES

[1] N. C. Seeman, "DNA in a material world," *Nature*, vol. 421, no. 6921, pp. 427–431, Jan. 2003.
[2] P. W. K. Rothemund, "Folding DNA to create nanoscale shapes and patterns," *Nature*, vol. 440, no. 7082, pp. 297–302, Mar. 2006.
[3] C. Chen, J. Xu, and X. Shi, "Adjusting linking strands to form size-controllable DNA origami rings," *IEEE Trans. Nanobiosci.*, vol. 19, no. 2, pp. 167–172, Apr. 2020.
[4] X. Shi, X. Wu, T. Song, and X. Li, "Construction of DNA nanotubes with controllable diameters and patterns using hierarchical DNA sub-tiles," *Nanoscale*, vol. 8, no. 31, pp. 14785–14792, 2016.
[5] X. Shi, W. Lu, Z. Wang, L. Pan, G. Cui, J. Xu, and T. H. LaBean, "Programmable DNA tile self-assembly using a hierarchical sub-tile strategy," *Nanotechnology*, vol. 25, no. 7, Feb. 2014, Art. no. 075602.
[6] N. Hampp, "Bacteriorhodopsin as a photochromic retinal protein for optical memories," *Chem. Rev.*, vol. 100, no. 5, pp. 1755–1776, May 2000.
[7] K. Chen, J. Kong, J. Zhu, N. Ermann, P. Predki, and U. F. Keyser, "Digital data storage using DNA nanostructures and solid-state nanopores," *Nano Lett.*, vol. 19, no. 2, pp. 1210–1215, Feb. 2019.
[8] L. M. Adleman, "Computing with DNA," *Sci. Amer.*, vol. 279, no. 2, pp. 54–61, 1998.
[9] J. Xu, X. Qiang, K. Zhang, C. Zhang, and J. Yang, "A DNA computing model for the graph vertex coloring problem based on a probe graph," *Engineering*, vol. 4, no. 1, pp. 61–77, Feb. 2018.
[10] Z. Yin, J. Cui, and C. Zhen, "Molecular beacon computing model for maximum weight clique problem," *Math. Comput. Simul.*, vol. 151, pp. 147–155, Sep. 2018.
[11] G. T. Walker, M. S. Fraiser, J. L. Schram, M. C. Little, J. G. Nadeau, and D. P. Malinowski, "Strand displacement amplification—An isothermal, in vitro DNA amplification technique," *Nucleic Acids Res.*, vol. 20, no. 7, pp. 1691–1696, 1992.
[12] P. Nielsen, M. Egholm, R. Berg, and O. Buchardt, "Sequence-selective recognition of DNA by strand displacement with a thymine-substituted polyamide," *Science*, vol. 254, no. 5037, pp. 1497–1500, Dec. 1991.
[13] M. C. Little *et al.*, "Strand displacement amplification and homogeneous real-time detection incorporated in a second-generation DNA probe system, BDProbeTecET," *Clin. Chem.*, vol. 45, no. 6, pp. 777–784, Jun. 1999.
[14] T. H. LaBean and H. Li, "Constructing novel materials with DNA," *Nano Today*, vol. 2, no. 2, pp. 26–35, Apr. 2007.
[15] X. Shi, X. Li, Z. Zhang, and J. Xu, "Improve capability of DNA automaton: DNA automaton with three internal states and tape head move in two directions," in *Proc. Int. Conf. Intell. Comput.* Berlin, Germany: Springer, 2005, pp. 71–79.
[16] S. Xiaolong, P. Linqiang, and X. Jin, "General DNA automaton model with R/W tape," in *Proc. Int. Conf. Intell. Comput.* Berlin, Germany: Springer, 2006, pp. 258–266.
[17] Y. Wang, P. Hu, X. Shi, and G. Cui, "DNA self-assembly for graph vertex 3-coloring problem," *J. Comput. Theor. Nanosci.*, vol. 9, no. 12, pp. 2086–2092, Dec. 2012.
[18] G. Seelig, D. Soloveichik, D. Y. Zhang, and E. Winfree, "Enzyme-free nucleic acid logic circuits," *Science*, vol. 314, no. 5805, pp. 1585–1588, Dec. 2006.
[19] A. Eshra, S. Shah, T. Song, and J. Reif, "Renewable DNA hairpin-based logic circuits," *IEEE Trans. Nanotechnol.*, vol. 18, pp. 252–259, 2019.
[20] R. Poli, N. F. McPhee, and J. E. Rowe, "Exact schema theory and Markov chain models for genetic programming and variable-length genetic algorithms with homologous crossover," *Genet. Program. Evolvable Mach.*, vol. 5, no. 1, pp. 31–70, Mar. 2004.
[21] L. Qian, E. Winfree, and J. Bruck, "Neural network computation with DNA strand displacement cascades," *Nature*, vol. 475, no. 7356, pp. 368–372, Jul. 2011.
[22] X. Song, A. Eshra, C. Dwyer, and J. Reif, "Renewable DNA seesaw logic circuits enabled by photoregulation of toehold-mediated strand displacement," *RSC Adv.*, vol. 7, no. 45, pp. 28130–28144, 2017.
[23] A. Eshra and A. El-Sayed, "An odd parity checker prototype using DNAzyme finite state machine," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 11, no. 2, pp. 316–324, Mar./Apr. 2014.
[24] S. Garg, S. Shah, H. Bui, T. Song, R. Mokhtar, and J. Reif, "Renewable time-responsive DNA circuits," *Small*, vol. 14, no. 33, Jul. 2018, Art. no. 1801470.
[25] C. Chen, H. Wang, E. Zhu, X. Shi, and J. Xu, "DNA logic multiplexing using toehold-mediated strand displacement," *IEEE Access*, vol. 8, pp. 88108–88114, 2020.
[26] F. Wang, H. Lv, Q. Li, J. Li, X. Zhang, J. Shi, L. Wang, and C. Fan, "Implementing digital computing with DNA-based switching circuits," *Nature Commun.*, vol. 11, no. 1, p. 121, Jan. 2020.
[27] J. Zhu, L. Zhang, S. Dong, and E. Wang, "Four-way junction-driven DNA strand displacement and its application in building majority logic circuit," *ACS Nano*, vol. 7, no. 11, pp. 10211–10217, Nov. 2013.
[28] E. Zhu, C. Chen, Y. Rao, and W. Xiong, "Biochemical logic circuits based on DNA combinatorial displacement," *IEEE Access*, vol. 8, pp. 34096–34103, 2020.
[29] X. Shi, Z. Wang, C. Deng, T. Song, L. Pan, and Z. Chen, "A novel biosensor based on DNA strand displacement," *PLoS ONE*, vol. 9, no. 10, Oct. 2014, Art. no. e108856.
[30] L. Qian and E. Winfree, "Scaling up digital circuit computation with DNA strand displacement cascades," *Science*, vol. 332, no. 6034, pp. 1196–1201, Jun. 2011.
[31] D. Soloveichik, G. Seelig, and E. Winfree, "DNA as a universal substrate for chemical kinetics," *Proc. Nat. Acad. Sci. USA*, vol. 107, no. 12, pp. 5393–5398, Mar. 2010.
[32] B. Yordanov, J. Kim, R. L. Petersen, A. Shudy, V. V. Kulkarni, and A. Phillips, "Computational design of nucleic acid feedback control circuits," *ACS Synth. Biol.*, vol. 3, no. 8, pp. 600–616, 2014.
[33] M. R. Lakin, S. Youssef, L. Cardelli, and A. Phillips, "Abstractions for DNA circuit design," *J. Roy. Soc. Interface*, vol. 9, no. 68, pp. 470–486, Mar. 2012.
[34] A. Phillips and L. Cardelli, "A programming language for composable DNA circuits," *J. Roy. Soc. Interface*, vol. 6, no. 4, pp. S419–S436, Aug. 2009.
[35] M. R. Lakin, S. Youssef, F. Polo, S. Emmott, and A. Phillips, "Visual DSD: A design and analysis tool for DNA strand displacement systems," *Bioinformatics*, vol. 27, no. 22, pp. 3211–3213, Nov. 2011.
[36] D. Y. Zhang and E. Winfree, "Control of DNA strand displacement kinetics using toehold exchange," *J. Amer. Chem. Soc.*, vol. 131, no. 47, pp. 17303–17314, Dec. 2009.

CONGZHOU CHEN received the B.S. degree in automation from the Wuhan University of Science and Technology, Wuhan, China, in 2014, and the M.S. degree in automation from the Huazhong University of Science and Technology, Wuhan, in 2016. He is currently pursuing the Ph.D. degree with Peking University, Beijing, China. His research interests include DNA nanotechnology and DNA biocomputing.

WEI XIAO received the bachelor's degree from Fujian Normal University, in 2018. He is currently pursuing the master's degree in computer science with Guangzhou University. His research interest includes DNA computing and biocomputing.

JIWEI ZHAO received the bachelor's degree from the Huazhong University of Science and Technology (HUST), in 2017, where he is currently pursuing the master's degree in artificial intelligence.

His research interest includes DNA computing and biocomputing.

ZHENG ZHANG received the bachelor's and Ph.D. degrees from the Huazhong University of Science and Technology (HUST), in 1998 and 2007, respectively. He is currently an Associate Professor with HUST. His main research interest includes pervasive computing and biocomputing.

XIAOLONG SHI received the Ph.D. degree in system engineering from the Huazhong University of Science and Technology. He was a Tenured Professor with the Huazhong University of Science and Technology. He is currently the Dean/Professor of the Institute of Computer Science and Technology, Guangzhou University.