Logic Synthesis of Recombinase-Based Genetic Circuits

Tai-Yin Chiu and Jie-Hong R. Jiang

Abstract—A synthetic approach to biology is a promising technique for various applications. Recent advancements have demonstrated the feasibility of constructing synthetic two-input logic gates in Escherichia coli cells with long-term memory based on DNA inversion induced by recombinases. On the other hand, recent evidences indicate that DNA inversion mediated by genome editing tools is possible; powerful genome editing technologies, such as CRISPR-Cas9 systems, have great potential to be exploited to implement large-scale recombinase-based circuits. What remains unclear is how to construct arbitrary Boolean functions based on these emerging technologies. In this paper, we lay the theoretical foundation formalizing the connection between recombinase-based genetic circuits and Boolean functions. It enables systematic construction of any given Boolean function using recombinase-based logic gates. We further develop a methodology leveraging existing electronic design automation (EDA) tools to automate the synthesis of complex recombinase-based genetic circuits with respect to area and delay optimization. Experimental results demonstrate the feasibility of our proposed method.

I. INTRODUCTION

The development of synthetic biology shows the feasibility to implement computing devices with DNA genetic circuits in living cells. Synthetic cellular designs often intended to implement certain functions that make cells respond to specific environmental stimuli or even change their growth and cellular development. For instance, synthetic toggle switches [1] and genetic oscillators [2]–[5] can be used to control cell metabolism, synthetic counters [6] can be potentially applied to the regulation of telomere length and cell aggregation, and genetic logic gates [7]–[10] can achieve digital computation in response to stimulus input signals. In addition to these transcription-based DNA circuits, new emerging translational mRNA circuits [11] are likely to have impact on mammalian regenerative medicine and gene therapy. Through the genetic engineering, synthetic cellular circuits with integrated logic and memory was proposed, such as CRISPR-Cas9 systems, have great potential to be duplicated in cell divisions. In recent work [12], a more efficient scheme for constructing synthetic cellular circuits with integrated logic and memory was proposed, where the computational result was automatically stored in the computing circuit configuration and the changes of configuration can be propagated to its descendant cells. The implemented circuits were built based on recombinases and tested in Escherichia coli cells and they showed a long-term memory for at least 90 cell generations. More recently, recombinase-based logic circuits has been applied in clinical uses. E.g., in [13] the authors demonstrate that biosensor made of recombinase-based logic gates can be used to detect pathological glycosuria in urine from diabetic patients. The ability to build complex recombinase-based logic circuits is an important step to enable widespread biomedical applications.

Specifically the synthetic cellular circuits proposed in [12] used serine recombinases Bxb1 and phiC31 to implement various two-input logic gates. A serine recombinase targeting a pair of non-identical recognition sites known as attB (attachment site bacteria) and attP (attachment site phage) is able to induce irreversible DNA inversion. As illustrated...
in Fig. 1(A), since the inversion makes the recognition sites become hybrid sites called attR and attI, which cannot be targeted by the recombinase, no further inversion is allowed afterwards.

We illustrate how recombinases take part in the implementation of two-input logic gates with the two-input AND gate example shown in Fig. 1B. (As a convention, in this paper we read a DNA sequence from left to right assuming the 5′-to-3′ direction of the coding strand.) Let molecules AHL and aTc be the stimulus inputs to a cell and act as inducers activating the expressions of recombinases Bxb1 and phiC31, respectively. These recombinases when activated will irreversibly invert (flip) the DNA sequences flanked by their recognition sites (denoted by the colored triangle pairs). The DNA sequences being flanked can be a promoter, a transcription terminator, or a reporter, e.g., a green fluorescent protein (GFP). Inverting these DNA sequences will alter the output gene expression. In Fig. 1B, two terminators were flanked by the recognition sites of recombinases Bxb1 and phiC31, and the output green fluorescent reporter is highly expressed only when both inducers AHL and aTc are in high concentration to activate Bxb1 and phiC31 which together further flip and disable both terminators (denoted by letter “T”). Therefore, the circuit of Fig. 1B effectively implements a two-input AND gate. Note that such DNA sequence changes will survive through cell divisions and can be inherited to descendant cells in different generations. Hence the so-implemented logic function can achieve a long-term transgeneration memory.

Note that the feasibility of constructing large recombinase-based circuits is limited to available recombinases. Nevertheless, with the advances of biotechnology, DNA inversion techniques mediated by genome editing approaches, such as ZFNs [14], TALENs [15]–[17], and CRISPR-Cas9 nucleases [17]–[20] have already been reported. It is envisaged that these genome editing tools could be alternatives scalable to realize large recombinase-based circuits [21]. Motivated by the viability and applicability of recombinase-based circuits, in this paper we formalize the construction of a general multi-input logic gate with its DNA sequence composed of series of promoters and transcription terminators targeted by multiple recombinases. We further characterize the set of Boolean functions realizable under such logic gates. In addition, we show a design flow for arbitrary Boolean function construction with cascaded recombinase-based logic gates. This automated design methodology is demonstrated by leveraging synthesis tool ABC [22], an electronic design automation (EDA) tool developed at UC Berkeley, to synthesize cascaded multi-level recombinase-based circuits.

The rest of the paper is organized as follows. In Section II some examples of multi-input recombinase-based logic gates are shown to motivate this work. In Section III the syntax and semantics of recombinase-based logic gates are formalized. In Section IV we propose a method to synthesize logic circuits composed of recombinase-based gates using conventional logic synthesis tools. In Section V experimental results are evaluated. Finally, conclusions and future work are remarked in Section VI.

II. PRELIMINARIES

To formalize the general multi-input gate construction, we use the three-input logic gates in Fig. 2 as an example to illustrate. Fig. 2A shows a realization of a 3-input AND gate using three recombinases \( R_1, R_2, \) and \( R_3 \), where molecule \( I_i \) is a stimulus input that activates the expression of recombinase \( R_i \), for \( i = 1, 2, 3 \). Then \( R_i \)'s induce the inversions of their corresponding DNA sequence fragments. In order to express GFP in this gate, first we require \( R_1 \) to invert the inverted promoter so that the RNA polymerase can bind to it and begin the transcription of the downstream DNA sequence in which the GFP gene resides. Second, \( R_2 \) is needed to flip the terminator to avoid the termination of transcription before reaching the GFP gene. Third, \( R_3 \) is demanded to upright the GFP gene for the RNA polymerase to initiate GFP production. Collectively, to have GFP highly expressed all \( R_i \)'s must exist, and thus this circuit implements a 3-input AND gate. Note that this 3-input AND gate, where the promoter and the reporter gene GFP can be flipped by recombinases, is designed in a different fashion from the 2-input AND gate in Fig. 1B, where only transcription terminators are inverted by recombinases. The additional choice of flipping the DNA fragments of promoter and GFP gives more flexibility for logic gate construction.

In Fig. 2B)-(H) we present seven other basic 3-input gates implemented with recombinases. Special implementations with nested targeting sites are applied on the XOR gate in (G) and the XNOR gate in (H). In the XOR gate in (G), the existence of one or three recombinases results in one or three times of GFP gene flipping and thus making the upside-down gene become upright, while the existence of two recombinases makes the GFP gene flip twice and remain upside down. Similar situations happen in the XNOR gate in (H).

Since the implementations of multi-input gates are possible, we are not constrained to using only 3-input gates and basic gate types, such as AND, OR, NAND, NOR, XOR, and XNOR gates. Rather, we can construct complex logic gates with more inputs. Fig. 3A shows an example of a 4-input logic circuit

\[
O = (R_1 + \bar{R}_2 \oplus R_3) \bar{R}_4,
\]

which can be directly realized by a single 4-input complex logic gate as shown in Fig. 3B, instead of cascading multiple two-input gates.

III. FORMALISM OF RECOMBINASE-BASED LOGIC GATES

A. Syntax of Well-Formed Sequences

We define the following syntax to formalize the DNA sequences of logic gates constructed with recombinases. Here the basic elements composing a legal DNA sequence of a recombinase-based logic gate are “atomic terms,” including (inverted/non-inverted) transcription factors, (inverted/non-inverted) promoters, (inverted/non-inverted) genes, and targeting sites of recombinases. The syntax of DNA sequence forming a legal recombinase-based logic gate can be defined as follows.

**Definition 1:** An atomic term in a DNA sequence is a transcription terminator \( T \), a promoter \( P \), a gene \( G \), an
GFP
GFP
GFP
T
GFP
I1 R1
I3 R3
I2 R2
I4 R4
O
R1
R2
R3
R4
O

Fig. 2. Implementation of basic 3-input logic gates using recombinases. The inputs of each gate from top to down are recombinases $R_1$, $R_2$, and $R_3$, respectively; inducer $I_i$ monitored by the cell activates the expression of $R_i$; the red, blue, and orange triangles denote the targeting sites of $R_i$, $i = 1, 2, 3$, respectively.

In this paper we concentrate on the special case of one-gene $wfs$ (1g-$wfs$), where only one gene $G$, which is neither inverted nor sandwiched by targeting sites, appears in the $wfs$ at the end of the sequence serving as the output. For example, $\{T\}_{r_1} G$, $d \{T\}_{r_2} G$, $\{d \{T\}_{r_1}\}_{r_2} G$, and $\{\{P\}_{r_3} \{L\}_{r_4}\}_{r_5} \{d \{T\}_{r_1}\}_{r_2} G$ are 1g-$wfs$’s. Notice that under the 1g-$wfs$ setting, the logic gate has a single output and the gene can only be transcribed in one direction from left to right.

A pair of targeting sites of a recombinase is called basic if it only flanks an atomic term. Otherwise, it is called non-basic. We call a 1g-$wfs$ basic if it contains only basic pairs of targeting sites, and non-basic if it contains some non-basic pair of targeting sites. For example, $\{P\}_{r_1} \{T\}_{r_2} \{L\}_{r_3} \{d\}_{r_4} G$ is a basic 1g-$wfs$. In contrast, $\{P\}_{r_1} \{T\}_{r_2} G$, $\{d\}_{r_1} G$, and $\{T\}_{r_1} \{P\}_{r_2} G$ are non-basic 1g-$wfs$’s.

Furthermore, a non-basic pair of targeting sites can be nested. That is, a non-basic pair of targeting sites can be flanked by another pair of targeting sites. For instance, $\{P\}_{r_1} \{T\}_{r_2} \{P\}_{r_3} \{d\}_{r_4} G$ has nested two pairs of targeting sites targeted by the recombinases $r_3$ and $r_4$.

We discuss the logic functions induced by basic and non-basic 1g-$wfs$’s in the following.

B. Semantics of Well-Formed Sequences

1) Basic well-formed sequences: We first study some reduction rules of basic 1g-$wfs$’s. Let $\sigma$ be the DNA sequence of a basic 1g-$wfs$ excluding the output gene, that is, $\sigma$ is a basic $wfs$ without any gene. We denote a $wfs$ without any gene as 0g-$wfs$. Because $\sigma$ is made of components $P$, $d$, $T$, $L$, $\{P\}_{r_1}$, $\{d\}_{r_1}$, $\{T\}_{r_1}$, and $\{L\}_{r_1}$, for any component $C$ in $\sigma$, the sequence $\sigma$ can be decomposed into

$$\sigma = \sigma_1 C \sigma_2,$$

where $\sigma_1$ and $\sigma_2$ are two 0g-$wfs$’s, if non-empty. We show that the logic gate induced by the 1g-$wfs$ $\sigma G$ can be further reduced to an equivalent form according to the type of the component $C$. 

Let the targeting sites $attP$ and $attB$ of recombinase $r$ in a DNA sequence be denoted as “{” and “}”, respectively. In the sequel, the subscripts of $\{r$ and $\}$, may be omitted for brevity when they are clear from the context or immaterial to the discussion. Note that targeting sites “{” and “}” of a recombinase must appear in a pair.

Definition 2: The syntax of a well-form sequence ($wfs$) is recursively defined as follows.

$$\langle wfs \rangle := \langle atomic \ term \rangle \ | \ \{ \langle wfs \rangle \}_{r_1} \ | \ \langle wfs \rangle \langle wfs \rangle.$$ 

(1)
When $C$ is a transcription terminator $T$, then $\sigma$ equals
\[ \sigma_1 T \sigma_2 G \equiv \sigma_2 G. \] (3)
This equivalence holds because any transcription that starts from $\sigma_1$ to gene $G$ is always blocked by the transcription terminator $T$ in the middle, making $\sigma_1 T$ a don’t-care and thus removable.

When $C$ is an inverted terminator $L$, then $\sigma$ equals
\[ \sigma_1 L \sigma_2 G \equiv \sigma_1 \sigma_2 G. \] (4)
This equivalence holds because the inverted terminator $L$ never blocks the transcription and is thus removable.

When $C$ is a promoter $P$, then $\sigma$ equals
\[ \sigma_1 P \sigma_2 G \equiv P \sigma_2 G. \] (5)
This equivalence holds because no matter whether there is a transcription that starts from $\sigma_1$ to $G$ or not, a transcription can always start from the promoter $P$. Therefore, $\sigma_1$ is a don’t-care and thus removable.

When $C$ is an inverted promoter $d$, then $\sigma$ equals
\[ \sigma_1 d \sigma_2 G \equiv \sigma_1 \sigma_2 G. \] (6)
This equivalence holds because the transcription that begins at $d$ proceeds across $\sigma_1$ in the direction from right to left, it does not pass through $G$. As a result, the expression of $G$ can not be initiated by $d$ and thus $d$ can be removed from the sequence.

When $C$ is $\{P\}_{r_i}, \{d\}_{r_i}, \{T\}_{r_i}$, or $\{L\}_{r_i}$, since an atomic term $A$ is equivalent to $\{A\}_r$ for recombinase $r$ being in low concentration (denoted $R = 0$ by treating $r$ as a Boolean variable $R$ of value 0) or $\{V\}_r$ for recombinase $r$ being in high concentration (denoted $R = 1$ by treating $r$ as a Boolean variable $R$ of value 1), the reduction rules for $C$ can be easily extended from the previous rules as summarized below.

\[ \sigma_1 \{T\}_{r_i} \sigma_2 G \equiv \begin{cases} \sigma_2 G, & \text{for } R = 0 \\ \sigma_1 \sigma_2 G, & \text{for } R = 1 \end{cases} \] (7)

\[ \sigma_1 \{L\}_{r_i} \sigma_2 G \equiv \begin{cases} \sigma_1 \sigma_2 G, & \text{for } R = 0 \\ \sigma_2 G, & \text{for } R = 1 \end{cases} \] (8)

\[ \sigma_1 \{P\}_{r_i} \sigma_2 G \equiv \begin{cases} P \sigma_2 G, & \text{for } R = 0 \\ \sigma_1 \sigma_2 G, & \text{for } R = 1 \end{cases} \] (9)

\[ \sigma_1 \{d\}_{r_i} \sigma_2 G \equiv \begin{cases} \sigma_1 \sigma_2 G, & \text{for } R = 0 \\ P \sigma_2 G, & \text{for } R = 1 \end{cases} \] (10)

With the above analysis, we can derive the corresponding Boolean function of a given 1g-wfs. Consider the 1g-wfs $\sigma G$ with the sequence $\sigma$ targeted by recombinases $r_i, i = 1 \cdots n$. Activating the expression of gene $G$ requires the recombinases $r_i$’s have adequate (high or low) concentrations so that the 1g-wfs $\sigma G$ effectively reduces to $PG$. The Boolean function induced by $\sigma G$ is determined through a series of decisions made by $r_i$’s. In essence, it corresponds to a decision list [23]. To illustrate, consider the example $\sigma = \{T\}_{r_1} \{P\}_{r_1} \{L\}_{r_3} \{d\}_{r_2} \{T\}_{r_1}$. The decision list induced by the 1g-wfs $\sigma G$ is shown in Fig. 4. Note that given a sequence without non-basic targeting sites, the decisions always start from the rightmost to the leftmost components because a component closer to the gene may overwrite the effects imposed by the components on its left and thus it is of higher priority. Therefore, the Boolean function of $\sigma G$ is determined starting from $R_1$ to $R_5$. In order to reduce $\sigma$ to $P$ to express gene $G$, we must require $R_1$ to be 1. Otherwise if $R_1 = 0$, $\sigma$ becomes equivalent to a null sequence no matter what other $R_i$’s are. Next, if we let $R_2$ be 1, we can have an equivalent sequence equal to $P$ as wished. Otherwise we can let $R_2$ be 0 and look for other possibilities for the reduction to $P$. If $R_2 = 0$, we can easily tell that the only possibility occurs when $R_3$ and $R_4$ are both 0 and that the logic of $R_5$ never affects the reduction. Collectively, the logic function of the gate $G$ is derived as $R_1 \cdot (R_2 + R_3 \cdot R_4)$, where symbol “$+$” denotes Boolean disjunction, symbol “$\cdot$” denotes Boolean conjunction, and symbol “$\cdot$” denotes Boolean negation. In the sequel, we sometimes omit the conjunction symbol “$\cdot$” in a Boolean expression.

\[
\begin{align*}
(T)_{r_5}(P)_{r_4}(L)_{r_3}(d)_{r_2}(T)_{r_1} & \rightarrow R_1 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \\
(T)_{r_2}(P)_{r_4}(L)_{r_3}(d)_{r_2}(T)_{r_1} & \rightarrow R_2 \cdot 1 \cdot 1 \cdot P \\
(T)_{r_3}(P)_{r_4}(L)_{r_3}(d)_{r_2}(T)_{r_1} & \rightarrow R_3 \cdot 1 \cdot 0 \cdot 0 \\
(T)_{r_5}(P)_{r_4} & \rightarrow R_4 \cdot 0 \cdot 1 \cdot P \\
(T)_{r_5} & \rightarrow R_5 \cdot 0 \cdot 0 \cdot 0 
\end{align*}
\]

Fig. 4. Decision list corresponding to 1g-wfs $\{T\}_{r_5} \{P\}_{r_4} \{L\}_{r_3} \{d\}_{r_2} \{T\}_{r_1}$ of gene $G$. Node labelled $R_i$ is the decision for the logic value of $R_i$. Nodes labelled 0 (resp. 1) stand for gene $G$ cannot (resp. can) be expressed. The sequences beside nodes are the equivalent sequences after the corresponding (partial) decisions.

In general, we can systematically convert any basic 1g-wfs to its corresponding logic function. To achieve this conversion, the operator $\Omega$ over a 1g-wfs is defined in Table [1]. For an empty sequence $\perp$, we define $\Omega[\perp] = 0$. E.g., for the 1g-wfs $\{T\}_{r_5} \{P\}_{r_4} \{L\}_{r_3} \{d\}_{r_2} \{T\}_{r_1}$, $G$, its Boolean function is derived by

\[
\Omega[\{T\}_{r_5} \{P\}_{r_4} \{L\}_{r_3} \{d\}_{r_2} \{T\}_{r_1}] = R_1(\Omega[\{T\}_{r_5} \{P\}_{r_4} \{L\}_{r_3} \{d\}_{r_2} \{T\}_{r_1}]) \\
= R_1(R_2 + (\Omega[\{T\}_{r_5} \{P\}_{r_4} \{L\}_{r_3} \{d\}_{r_2} \{T\}_{r_1}])) \\
= R_1(R_2 + (R_3(\Omega[\{T\}_{r_5} \{P\}_{r_4} \{L\}_{r_3} \{d\}_{r_2} \{T\}_{r_1}]))) \\
= R_1(R_2 + (R_3(\Omega[\{T\}_{r_5} \{P\}_{r_4} \{L\}_{r_3} \{d\}_{r_2} \{T\}_{r_1}}))
\]
### Table I

Operators for Parsing Basic 1g-WFS $\sigma CG$, with (Non-empty) 0g-WFS $\sigma$, Component C, and Gene G, to Logic Function.

| Component $C$ | Operator $\Omega[\sigma C]$ |
|---------------|-------------------------------|
| $T$           | $0 \cdot \Omega[\sigma]$     |
| $P$           | $1 \cdot \Omega[\sigma]$     |
| $\{T\}_r$     | $R \cdot \Omega[\sigma]$     |
| $\{P\}_r$     | $\overline{R} \cdot \Omega[\sigma]$ |
| $d$           | $1 \cdot \Omega[\sigma]$     |
| $\{L\}_r$     | $R \cdot \Omega[\sigma]$     |
| $\{d\}_r$     | $R \cdot \Omega[\sigma]$     |

$= R_1(R_2 + (\overline{R_3}(\overline{R_4} + (R_3(\Omega[\sigma])))))$

$= R_1(R_2 + (\overline{R_3}(\overline{R_4} + (R_3(0))))))$

$= R_1(R_2 + \overline{R_3} R_4))$.  

2) Non-basic well-formed sequences: We extend the above derivation of Boolean function to non-basic 1g-wfs’s by having the operator $\Omega$ over a 0g-wfs $\{\sigma\}_r$ (which can be basic or non-basic) defined as

$$\Omega[[\sigma]]_r = R \cdot \Omega[\sigma] + R \cdot \Omega[\sigma]$$

where $\sigma$ is the inverted sequence of $\sigma$. To understand Eq. (11), consider a 1g-wfs $\sigma CG$ with a pair of non-basic targeting sites. Suppose $\sigma = \{\sigma_1\}_r$, where $\sigma_1$ is a basic 0g-wfs. Then $\sigma$ is equal to $\sigma_1$ when $R = 0$ and to $\sigma_1$, the inverted sequence of $\sigma_1$, when $R = 1$. For example, the logic function for $(\{T\}_r, \{P\}_r, \{\{L\}_r, \{d\}_r\}_r)_{r_1} G$ can be obtained by

$$\Omega[[\sigma]]_r = R \cdot \Omega[\sigma] + R \cdot \Omega[\sigma]$$

For a 1g-wfs with multiple (possibly nested) non-basic pairs of targeting sites, its logic function can also be directly obtained by the $\Omega$ operator. For example, the logic function for $(\{P\}_r, \{\{L\}_r, \{d\}_r\}_r)_{r_1} G$ can be obtained by

$$\Omega[[\sigma]]_r = R \cdot \Omega[\sigma] + R \cdot \Omega[\sigma]$$

... (remaining text continues with mathematical expressions and diagrams)
IV. CONSTRUCTION OF MULTI-LEVEL RECOMBINASE-BASED LOGIC CIRCUITS

With the recombinase-based logic gates built from 1g-wfs’s, we can cascade them to implement arbitrary complex multi-level circuits. For example, the logic function $Z = (A + B)(A \oplus B)$ can be implemented with the two-level circuit shown in Fig. 6(A), which is composed of an OR-gate, an XOR-gate, and an AND-gate. One possible DNA implementation of $Z$ with cascade can be derived by converting each gate to their 1g-wfs realizations as shown in Fig. 6(B). The 1g-wfs’s that encode the genes $R_1$ and $R_2$ encode the recombinases $r_1$ and $r_2$, respectively, which are the inputs to the downstream AND gate. The protein encoded by the gene $Z$ is the output of the circuit.

Fig. 6. (A) Logic circuit of Boolean function $Z = (A + B)(A \oplus B)$. (B) The corresponding DNA implementation of the circuit in (A) with gate cascade. $A$ and $B$ denote the recombinase inputs of the overall circuit. The genes $R_1$ and $R_2$ encode the recombinases $r_1$ and $r_2$, respectively, which are the inputs to the downstream AND gate. The protein encoded by the gene $Z$ is the output of the circuit.

Fig. 7. Logic synthesis flow for the implementation of recombinase-based logic circuit

Fig. 8. Circuit diagram of ISCAS benchmark c17. c17 circuit consists of six NAND gates with five inputs $\{A, B, C, D, E\}$ and two outputs $\{Y, Z\}$.

For area-driven synthesis of benchmark c17, there are 44 DNA gates defined by their 1g-wfs’s with up to three recombinase inputs. They are collected as the library as shown in Fig. 9. According to the experiment in [12], where the promoters and transcription terminators used are roughly of the same length, we treat the area cost of both promoter and transcription terminator as unity. Therefore, the area cost of a DNA gate is defined as the number of atomic terms, excluding the output gene, that appear in the 1g-wfs of the gate. For example, the gate c3_1 corresponding to a 3-input OR gate has three inverted promoters as shown in Fig. 2(D). Hence, the area cost of c3_1 is counted as 3 units. By providing the c17 netlist and the library to ABC, the tool can perform optimization and technology mapping to find an area-optimized circuit composed of DNA gates of the library.

Fig. 10 shows the result described in Verilog language of the synthesized c17 recombinase-based circuit using library gates listed in Fig. 9. The synthesized circuit comprises gates c2_4, c2_5, c3_14, and c3_25, and the total area cost is 10 units. Note that the naive DNA circuit implementation of c17
circuit by converting the digital logic gates in Fig. 8 to the corresponding DNA gates results in a total area cost of 12 units. Compared to the naive implementation, the area cost of the circuit synthesized by ABC technology mapping decreases. The logic functions of Y and Z in the synthesized circuit can be easily verified to be consistent with Eq. (12), implying the correctness of the synthesis result. The DNA circuit of module c17 in Fig. 10 is plotted in Fig. 12(A).

| NAME | AREA | FUNCTION |
|------|------|----------|
| c1_1 | 1    | O = a    |
| c1_2 | 1    | O = !a   |
| c2_1 | 2    | O = a+(b) |
| c2_2 | 2    | O = a+(b) |
| c2_3 | 2    | O = a+(b) |
| c2_4 | 2    | O = !a+(b) |
| c2_5 | 2    | O = a*(b) |
| c2_6 | 2    | O = a*(b) |
| c2_7 | 2    | O = !a*(b) |
| c2_8 | 2    | O = !a*(b) |
| c3_1 | 3    | O = a+(b+(c)) |
| c3_2 | 3    | O = a+(b+(c)) |
| c3_3 | 3    | O = a+(b+(c)) |
| c3_4 | 3    | O = a+(b+(c)) |
| zero | 0    | O = CONST0 |

Fig. 9. Library of DNA gates with specification of area cost. The library contains 44 different cells and each cell corresponds to a DNA logic gate defined by a 1g-wfs with up to three inputs. The variables a, b, and c in a function specification represents the recombinase inputs to a gate, and the variable O denotes the gate output.

Note that there can be more than one area-optimized circuit of a logic function. For comparison, in Fig. 11 we show another manually designed DNA implementation of c17 circuit whose area cost is 10 units as well. The corresponding DNA circuit is plotted in Fig. 12(B). Notice that the two circuits in Fig. 12 differ not only in their constituent logic gates, but also in their logic depths. The circuit of Fig. 12(A) is of two logic levels, whereas that of Fig. 12(B) is of three logic levels. There are six longest paths in the former circuit:

\[
\begin{align*}
A \rightarrow n7 \rightarrow Y; \\
B \rightarrow n7 \rightarrow Y; \\
B \rightarrow n8 \rightarrow Y; \\
B \rightarrow n8 \rightarrow Z; \\
C \rightarrow n8 \rightarrow Y; \\
C \rightarrow n8 \rightarrow Z.
\end{align*}
\]

They involve a cascade of two logic gates. On the other hand, there are two longest paths in the latter circuit:

\[
\begin{align*}
B \rightarrow n7 \rightarrow n8 \rightarrow Y; \\
C \rightarrow n7 \rightarrow n8 \rightarrow Y.
\end{align*}
\]

They involve a cascade of three logic gates. Although these two circuits have the same area cost, the circuit of Fig. 12(A) is preferred due to its better performance. In the experiments, we will synthesize circuits with area or performance optimized.

V. EXPERIMENTAL EVALUATION

To demonstrate the feasibility of the proposed synthesis flow, we experiment on other 67 ISCAS benchmark circuits using recombinase-based DNA gates. We expanded the library such that it includes all 684 DNA gates with decision list functions up to five inputs. In the library, the area cost of a gate is determined by the number of atomic terms, excluding the output gene, appearing in its corresponding 1g-wfs. We use a simple unit delay model for all the logic gates.

The experiment results of 54 (out of the 67) circuits are shown in Table III. The numbers of primary inputs/outputs, the number of inverters, and the number of logic gates (with the number of included buffers, if non-zero, reported...
in parentheses) are listed Columns 2, 3, and 4, respectively. The circuits were synthesized under two optimization settings: one for area optimization and the other for delay optimization. The results of area optimization are reported in Columns 5–7 and those of delay optimization are reported in Columns 8–10. For each synthesized circuit, its number of DNA gates, total area, and gate level are shown. In the naive implementations of benchmark circuits by simply converting the digital logic gates to the corresponding DNA gates, the total area of a DNA circuit can be roughly calculated as 

\[
\text{"#inverter" + 2 \times "#gate".}
\]

Compared to the naive implementation, the circuits synthesized by ABC have much less area cost. Taking circuit b18 for example, we observe that the total area of the naive implementation is about 202110 which is much larger compared to the area 101870 of the area-optimized implementation and 105328 of the delay-optimized implementation. On the other hand, comparing area and delay optimized b18 circuits, delay optimization reduces the number of gate levels from 137 to 51 at cost of increasing area by 3500 units.

VI. CONCLUSIONS

In this paper, we generalized the two-input recombinase-based DNA logic gates to multi-input cases. We formalized the syntax of recombinase-based logic gate construction, and obtained the Boolean function semantics of well-defined DNA sequences of recombinase-based logic gates. We also showed how to synthesize multi-level recombinase-based logic circuits using existing logic synthesis tools. Experimental results demonstrate the feasibility of our proposed methods. As recombinase-based logic circuits have been used in clinical biomarker detection, our results may automate complex recombinase-based circuit construction for advanced biomedical applications. With more and more evidence that DNA inversion can be mediated by genome editing tools such as the CRISPR/Cas9 system, we anticipate broad applications of recombinase-based logic gates in the future.

REFERENCES

[1] T. S. Gardner, C. R. Cantor, and J. J. Collins, “Construction of a genetic toggle switch in Escherichia coli,” Nature, vol. 403, no. 6767, pp. 339–342, 2000.
[2] M. B. Elowitz and S. Leibler, “A synthetic oscillatory network of transcriptional regulators,” Nature, vol. 403, no. 6767, pp. 335–338, 2000.
[3] J. Hasty, M. Dolnik, V. Rottschäfer, and J. J. Collins, “Synthetic gene network for entraining and amplifying cellular oscillations,” Physical Review Letters, vol. 88, no. 14, p. 148101, 2002.
[4] E. Fung, W. W. Wong, J. K. Suen, T. Butler, S.-g. Lee, and J. C. Liao, “A synthetic gene–metabolic oscillator,” Nature, vol. 435, no. 7038, pp. 118–122, 2005.
[5] J. Stricker, S. Cookson, M. R. Bennett, W. H. Mather, L. S. Tsimring, and J. Hasty, “A fast, robust and tunable synthetic gene oscillator,” Nature, vol. 456, no. 7221, pp. 516–519, 2008.
[6] A. E. Friedland, T. K. Lu, X. Wang, D. Shi, G. Church, and J. J. Collins, “Synthetic gene networks that count,” Science, vol. 324, no. 5931, pp. 1199–1202, 2009.
[7] A. Tamsir, J. J. Tabor, and C. A. Voigt, “Robust multicellular computing using genetically encoded NOR gates and chemical ‘wires’,” Nature, vol. 469, no. 7329, pp. 212–215, 2011.
[8] B. Wang, R. I. Kitney, N. Joly, and M. Buck, “Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology,” Nature Communications, vol. 2, p. 508, 2011.
[9] T. S. Moon, C. Lou, A. Tamsir, B. C. Stanton, and C. A. Voigt, “Genetic programs constructed from layered logic gates in single cells,” Nature, vol. 491, no. 7423, pp. 249–253, 2012.
[10] J. Bonnet, P. Yin, M. E. Ortiz, P. Subsoontorn, and D. Endy, “Amplifying genetic logic gates,” Science, vol. 340, no. 6132, pp. 599–603, 2013.
[11] M. B. Kopniczky, S. J. Moore, and P. S. Freemont, “Multilevel regulation and translational switches in synthetic biology,” IEEE Transactions on Biomedical Circuits and Systems, vol. 9, no. 4, pp. 485–496, 2015.
[12] P. Sisti, J. Yazbek, and T. K. Lu, “Synthetic circuits integrating logic and memory in living cells,” Nature Biotechnology, vol. 31, no. 5, pp. 448–452, 2013.
[13] A. Courbet, D. Endy, E. Renard, F. Molina, and J. Bonnet, “Detection of pathological biomarkers in human clinical samples via amplifying genetic switches and logic gates,” Science Translational Medicine, vol. 7, no. 289, p. 289ra83, 2015.
[14] H. J. Lee, J. Kweon, E. Kim, S. Kim, and J.-S. Kim, “Targeted chromosomal duplications and inversions in the human genome using zinc finger nucleases,” Genome Research, vol. 22, no. 3, pp. 539–548, 2012.
[15] A. Gupta, V. L. Hall, F. O. Kok, M. Shin, J. C. McNulty, N. D. Lawson, and S. A. Wolfe, “Targeted chromosomal deletions and inversions in zebrafish,” Genome Research, vol. 23, no. 6, pp. 1008–1017, 2013.
[16] D. F. Carlson, W. Tan, S. G. Lillico, D. Stverákova, C. Proudfoot, M. Christian, D. F. Voytas, C. R. Long, C. B. A. Whitelaw, and S. C. Fahrenkrug, “Efficient TALEN-mediated gene knockout in livestock,”
| circuit name | benchmark profile | area optimization | delay optimization |
|-------------|------------------|------------------|-------------------|
| b03         | 34/34            | 16               | 106               | 91 217 7 | 79 228 4 |
| b04         | 77/74            | 105              | 547               | 373 852 22 | 358 881 8 |
| b06         | 11/15            | 7                | 32                | 25 56 6 | 24 62 3 |
| b07         | 50/57            | 61               | 322               | 257 583 23 | 235 615 8 |
| b08         | 30/25            | 26               | 123               | 90 224 12 | 85 233 5 |
| b09         | 29/29            | 24               | 116               | 106 228 10 | 96 240 5 |
| b10         | 28/23            | 32               | 140               | 100 260 11 | 96 298 4 |
| b11         | 38/37            | 148              | 578               | 333 788 25 | 301 829 8 |
| b12         | 126/127          | 113              | 831               | 707 1648 15 | 673 1786 6 |
| b13         | 63/63            | 52               | 237               | 172 381 12 | 153 401 4 |
| b14         | 277/299          | 1531             | 8236              | 2851 6947 124 | 2791 7749 18 |
| b15         | 1452/1512        | 4474             | 26303             | 15344 37726 104 | 14802 39178 28 |
| b18         | 3357/3343        | 20372            | 90869             | 43018 101870 137 | 40277 105328 51 |
| b20         | 522/512          | 3068             | 16614             | 6119 14497 128 | 6111 16545 21 |
| b21         | 522/512          | 3089             | 16938             | 6173 14724 121 | 6147 16631 21 |
| b22         | 767/757          | 4491             | 24671             | 9302 22107 124 | 9286 24908 21 |
| c432        | 36/7             | 40               | 120               | 79 200 25 | 91 276 11 |
| c499        | 41/32            | 40               | 162               | 407 794 21 | 335 833 11 |
| c880        | 60/26            | 63               | 320 (26)          | 234 530 26 | 208 553 8 |
| c1355       | 41/32            | 40               | 506 (32)          | 394 781 19 | 328 878 10 |
| c1908       | 33/25            | 277              | 603 (162)         | 336 690 28 | 271 736 13 |
| c2670       | 233/140          | 321              | 872 (196)         | 409 956 19 | 400 1002 9 |
| c3540       | 50/22            | 490              | 1179 (223)        | 566 1473 36 | 553 1649 14 |
| c5315       | 178/123          | 581              | 1726 (313)        | 942 2202 25 | 908 2333 12 |
| c6288       | 32/32            | 32               | 2384              | 1825 3709 89 | 1502 3995 38 |
| c7552       | 207/108          | 876              | 2636 (534)        | 1149 2496 59 | 1084 2754 11 |
| s208        | 19/10            | 35               | 61                | 39 100 8 | 39 105 3 |
| s298        | 17/20            | 44               | 75                | 55 125 7 | 52 138 3 |
| s344        | 24/26            | 59               | 101               | 82 178 11 | 67 175 4 |
| s349        | 24/26            | 57               | 104               | 84 179 11 | 67 175 4 |
| s382        | 24/27            | 59               | 99                | 78 172 8 | 70 191 3 |
| s386        | 13/13            | 41               | 118               | 71 186 7 | 61 195 3 |
| s400        | 24/27            | 56               | 106               | 80 173 9 | 76 220 3 |
| s420        | 35/18            | 74               | 122               | 79 202 11 | 72 196 4 |
| s444        | 24/27            | 62               | 119               | 75 169 9 | 74 210 3 |
| s510        | 25/13            | 32               | 179               | 116 311 8 | 102 324 4 |
| s526        | 24/27            | 52               | 141               | 88 202 11 | 79 223 3 |
| s641        | 54/43            | 272              | 107               | 94 217 17 | 82 232 6 |
| s713        | 54/42            | 254              | 139               | 90 212 16 | 85 237 6 |
| s820        | 23/24            | 33               | 256               | 130 353 8 | 129 394 4 |
| s832        | 23/24            | 25               | 262               | 132 358 9 | 135 406 4 |
| s838        | 67/34            | 149              | 241               | 163 415 12 | 142 398 5 |
| s1196       | 32/32            | 141              | 388               | 243 647 17 | 236 734 6 |
| s1238       | 32/32            | 80               | 428               | 278 734 17 | 259 790 7 |
| s1423       | 91/79            | 167              | 490               | 341 775 50 | 313 815 13 |
| s1488       | 14/25            | 103              | 550               | 299 820 12 | 272 910 4 |
| s1494       | 14/25            | 89               | 558               | 303 829 11 | 279 920 4 |
| s5378       | 214/228          | 1775             | 1004              | 844 1843 14 | 780 1849 7 |
| s9234       | 247/250          | 3570             | 2027              | 1065 2379 20 | 986 2442 9 |
| s13207      | 700/790          | 5378             | 2573              | 2006 4075 26 | 1818 4153 9 |
| s13850      | 611/684          | 6324             | 3448              | 2224 4946 35 | 2131 5018 16 |
| s35932      | 1763/2048        | 3861             | 12204             | 6776 14953 9 | 5565 14718 5 |
| s38417      | 1664/1742        | 13470            | 8709              | 6147 14319 23 | 5858 14551 10 |
| s38584      | 1464/1730        | 7805             | 11448             | 7066 16905 37 | 6243 16433 11 |
inversions mediated by TALENs and CRISPR/Cas in zebrafish,” *Nucleic Acids Research*, vol. 41, no. 14, p. e141, 2013.

[18] R. B. Blasco, E. Karaca, C. Ambrogio, T.-C. Cheong, E. Karayol, V. G. Minero, C. Voena, and R. Chiarle, “Simple and rapid in vivo generation of chromosomal rearrangements using CRISPR/Cas9 technology,” *Cell Reports*, vol. 9, no. 4, pp. 1219–1227, 2014.

[19] P. S. Choi and M. Meyerson, “Targeted genomic rearrangements using CRISPR/Cas technology,” *Nature Communications*, vol. 5, p. 3728, 2014.

[20] J. Li, J. Shou, Y. Guo, Y. Tang, Y. Wu, Z. Jia, Y. Zhai, Z. Chen, Q. Xu, and Q. Wu, “Efficient inversions and duplications of mammalian regulatory DNA elements and gene clusters by CRISPR/Cas9,” *Journal of Molecular Cell Biology*, vol. 7, no. 4, pp. 284–298, 2015.

[21] T.-Y. Chiu, C.-H. Hsieh, and J.-H. R. Jiang, “Realization of large logic circuits with long-term memory using CRISPR/Cas9 systems,” in *Proc. International Workshop on Bio-Design Automation (IWBDA)*, 2016.

[22] R. Brayton and A. Mishchenko, “ABC: An academic industrial-strength verification tool,” in *Proc. International Conference on Computer Aided Verification (CAV)*. Springer, 2010, pp. 24–40.

[23] R. R. Rivest, “Learning decision lists,” *Machine Learning*, vol. 2, no. 3, pp. 229–246, 1987.