Binary Counting with Chemical Reactions

Aleksandra Kharam, Hua Jiang, Marc Riedel, and Keshab Parhi
Electrical and Computer Engineering, University of Minnesota
Minneapolis, MN 55455
`http://cctbio.ece.umn.edu`
E-mail: {`veden002, hua, mriedel, parhi`}`@umn.edu`

This paper describes a scheme for implementing a binary counter with chemical reactions. The value of the counter is encoded by logical values of "0" and "1" that correspond to the absence and presence of specific molecular types, respectively. It is incremented by one every time any quantity of a trigger type is injected. The system is "self-timed:" the rate at which the trigger type is injected does not matter. The system consumes the trigger type and then the binary value can be incremented again. Unlike all previous schemes for chemical computation, our scheme is dependent only on coarse rate categories for the reactions ("fast" and "slow"). Given such categories, the computation is exact and independent of the specific reaction rates. Synchronization of the phases of the computation is achieved through a three-phase protocol that transfers quantities between chemical types based on the absence of other types. We validate our designs through transient stochastic simulations of the chemical kinetics. Although conceptual for the time being, our methodology has potential applications in domains of synthetic biology such as biochemical sensing and drug delivery. We are exploring DNA-based computation via strand displacement as a possible experimental chassis.

1. Introduction

In the nascent field of synthetic biology, researchers are striving to create biological systems with functionality not seen in nature. Examples of such research include Salmonella that secretes spider silk proteins,[1] yeast that degrades biomass into ethanol,[2] and *E. coli* that produces antimalarial drugs.[3]

The field aims to apply engineering methods in a deliberate way.[4] It also suggests a new constructive approach to validating science. As the great Caltech physicist Richard Feynman stipulated, "If I can't create it, I don't understand it." Understanding is achieved by constructing and testing simplified systems from the bottom up, teasing out and nailing down fundamental principles in the process. As Drew Endy, an eloquent proponent of synthetic biology, describes it: natural biological systems are fiendishly complicated; instead of endlessly probing them with experiments, in some cases, we are better off rebuilding the functionality from the ground up. This provides engineered "surrogates" that are easier to understand and interact with.

We bring a particular mindset to the problem of synthesizing new biological functions. We tackle synthesis at a *conceptual* level, working with abstract chemical types. Working at this level, we implement *computational* constructs, that is to say, chemical reaction networks that compute specific outputs as a function of inputs. Here we aim for efficient and robust constructs that are rate independent. Then we map the design onto specific chemical substrates. We are exploring DNA-based computation via strand displacement as a possible experimental chassis for the ideas in this paper.[5]

The analogy for this approach is the design flow for digital electronics. There different designs are systematically explored at a *technology-independent* level, in terms of Boolean functions. Once the best design is found, it is mapped to specific technology libraries in silicon.[6]

In our prior and related work, we have described a variety of constructs for chemical reaction networks: logical operations such copying, comparing and incrementing/decrementing;[7] programming constructs such as "for" and "while" loops;[8] arithmetic operations such as multiplication, exponentiation and logarithms;[7,8] and signal processing operations such as filtering.[9] The framework that we have adopted is largely due to the group at Caltech.[10–12]

In this paper, we present a scheme for implementing a binary counter with chemical reactions. The value of the counter is encoded by logical values of "0" and "1" that correspond to the absence and presence of specific molecular types, respectively. It is incremented by one every time any quantity of a trigger type is injected. The system is "self-timed:" the rate at which the trigger type is injected does not matter. The system consumes the trigger type and then the binary value can be incremented again. Unlike all previous schemes for chemical computation, our scheme is dependent only on coarse rate categories for the reactions ("fast" and "slow"). Given such categories, the computation is exact and independent of the specific reaction rates. Synchronization of the phases of the computation is achieved through a three-phase protocol that transfers quantities between chemical types based on the absence of other types.

This paper is organized as follows. In Section 2, we summarize the main principles and the basic algorithm for our implementation of the binary counter. In Section 3, we introduce some specific concepts that we use, namely the concepts of "prereactants" and "absence indicators." We also introduce the essential synchronization mechanism that we use, namely the "red-green-blue" (RGB) scheme. Building on these concepts, we present the design of the molecular counter. In Section 4, we analyze our design with transient stochastic simulation. Finally, in Section 5, we suggest possible improvements to our design and we discuss our on-going work in implementing the scheme with DNA strand-displacement reactions.

1.1. *Chemical Model*

We adopt the model of discrete, stochastic chemical kinetics.[13] We assume that the system is well-stirred; accordingly, we only track the quantities of the different molecular types. These are whole numbers (i.e., non-negative integers). Reactions fire and alter these quantities by discrete, integer amounts. Consider the reaction

$$X_1 \xrightarrow{\text{fast}} X_2 + X_3. \tag{1}$$

When this reaction fires, one molecule of X_1 is consumed, one of X_2 is produced, and one of X_3 is produced. (Accordingly, X_1 is called a *reactant* and X_2 and X_3 the *products*.)

Given several reactions, the probability of each firing is proportional both to its rate and to the quantities of its reactants present. The challenge in setting up computation with chemical reactions is that they execute asynchronously and at variable rates, dependent on factors such as temperature. In spite of this, we aim to implement computation that does not depend on the rates. We will only speak of rates in qualitative terms, e.g., "fast" vs. "slow" (in our notation, such qualitative rates are listed above the arrows for reactions.)

The evolution of a chemical system over time can be characterized through stochastic simulation. First proposed by Gillespie, stochastic simulation has become the workhorse of computational biology – the equivalent, one might say, of SPICE for electrical engineering.[14] Such simulation tracks integer quantities of the molecular species, executing reactions at random based on propensity calculations. Repeated trials are performed and the probability distribution of different outcomes is estimated by averaging the results.[10,15]

2. Counting in Binary

We first review some of the algorithmic principles of counting in binary. Then we present an intuitive description of our approach to implementing a molecular binary counter.

Z	Y	X
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1
⋮	⋮	⋮

Fig. 1. Sequence of values in a three-bit binary counter.

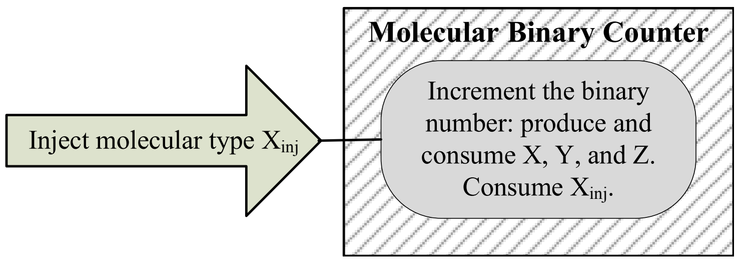
Inject molecular type X_{inj}
Molecular Binary Counter
Increment the binary number: produce and consume X, Y, and Z. Consume X_{inj}.

Fig. 2. Basic functionality of the molecular counter.

2.1. *General Principles*

Figure 1 lists the binary numbers that a 3-bit binary counter cycles through, starting at “000” and ending at “111.”

(1) *Every time the binary count incremented, the least significant (i.e., right-most) bit is flipped.* For instance, in the sequence $00\underline{0} \rightarrow 00\underline{1} \rightarrow 01\underline{0} \rightarrow 01\underline{1} \rightarrow 10\underline{0} \rightarrow 10\underline{1}$, note that least significant bit (underlined) alternates: $0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 1$.

(2) *Every time the binary count is incremented, exactly one bit changes from “0” to “1”. (However, several bits may change from “1” to “0.”)* For instance, in the sequence $00\underline{0} \rightarrow 0\underline{0}1 \rightarrow 01\underline{0} \rightarrow \underline{0}11 \rightarrow 10\underline{0} \rightarrow 1\underline{0}1$, the bits that change from “0” to “1” are underlined. Note that there is exactly one such bit each time. (As will be discussed in Section 3, this principle is important for synchronizing our molecular counter.)

(3) *When the binary count is incremented, a given bit changes from “0” to “1” only if all bits of lesser significance (i.e., all bits to the right of it) are “1.”* For instance, in the sequence $000 \rightarrow 00\underline{1} \rightarrow 01\underline{0}$, the second bit changes from “0” to “1” when the first bit was “1.” In the sequence $0\underline{11} \rightarrow 1\underline{00} \rightarrow 101$ the third bit changes from “0” to “1” when the first and second bits were “1.”

2.2. *Towards a Molecular Binary Counter*

Throughout this paper, the exposition will be in terms of a three-bit binary counter. However, the ideas can readily be generalized to an n-bit counter. We encode the binary values of “0” and

"1" by the presence or absence of specific molecular types, respectively. For the binary sequence in Figure 1, we use the types X, Y and Z. (We will call these "bit types.") For instance, if types X and Z are present, while type Y is absent, the corresponding binary number is "101".

Figure 2 shows the basic functionality of our molecular counter. Every time we want to increment it, we inject some amount of a "trigger" type X_{inj}. The system consumes X_{inj} and increments the binary value specified by the quantities of X, Y and Z. Once all the molecules of X_{inj} have been consumed, the counter can be incremented again.

Tables 2 and 3 specify the set of chemical reactions for our three-bit counter. In order to elucidate the final design, we will provide a succession of design refinements:

(1) We start with a simple intuitive set of reactions, ignoring issues such as synchronization (Section 2.3).
(2) We introduce two specific concepts that we use to implement the counter: the concept of "prereactants" and that of "absence indicators" (Section 3.1).
(3) We introduce our essential synchronization mechanism, the so-called "red-green-blue" (RGB) scheme (Section 3.2).
(4) Finally, we provide the full design of the counter, consisting of 24 chemical reactions (Sections 3.3).

2.3. *Intuitive Model*

A molecular counter cannot directly set bits to "0" or to "1"; rather the functionality much be achieved by reactions that produce and consume the molecular types corresponding to these bits. Call the three bits of the counter, the *high*, *middle* and *low* bits, encoded by the presence/absence of types X, Y and Z, respectively. The low bit is set to "1" by producing molecules of X whenever the type X_{inj} is injected into the system:

$$X_{\text{inj}} \to X. \tag{2}$$

The middle bit is set to "1" by producing molecules of Y whenever the type X is present:

$$X \to Y. \tag{3}$$

The high bit is set to "1" by producing molecules of Z whenever both types X and Y present:

$$X + Y \to Z. \tag{4}$$

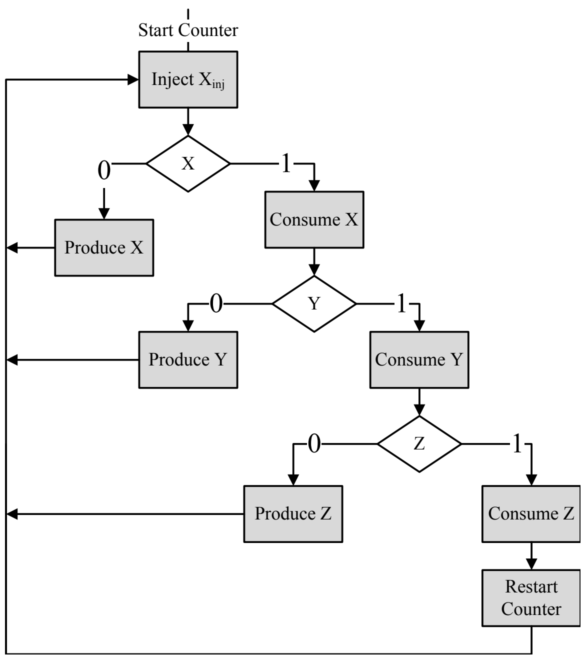

Fig. 3. Basic algorithm for the molecular counter.

Note that, in each of these reactions, the system consume molecules of X, Y and Z, reseting the corresponding bits to "0." When molecules of all three types X, Y and Z are present, the corresponding binary number is "111". The counter is reset:

$$X + Y + Z \to \varnothing. \tag{5}$$

(The symbol $\varnothing$ as a product indicates "nothing", meaning that the type degrades into products that are no longer tracked or used.)

A flowchart for the algorithm that we use is given in Figure 3. In the figure, decisions to produce and consume molecular types are made according to the presence and absence of types. (As we refine the design, we will have to implement these "decisions" through chemical reactions.) Let's assume the current binary number is set to "101". This number corresponds to the absence of Y and the presence of X and Z. Suppose that we inject the trigger type X_{inj}; we move to the first decision box. Since X is present, we do not produce more of it. We consume molecules of X and move to the next decision box. Here we check for the presence of type Y. Since Y is absent, we move to the left and produce molecules of Y. With the absence of X and the presence of Y and Z, the binary number has changed to "110". Next we return to "idle state," waiting for the next injection.

3. Synchronization

The challenge in setting up the molecular counter is that all the chemical reactions fire asynchronously. Each reaction starts producing its products as soon as its reactants are available. If these products participate as reactants in other reactions, then they immediately start getting consumed. Accordingly, with Reactions 2–5, we will not get a binary counter, encoded by the presence and absence of X, Y and Z. Rather, we will get a jumble of all of these. In particular, note that with Reaction 3, Y is produced from X. As soon as molecules of Y are available, Reaction 4 starts consuming molecules of X and Y to produce molecules of Z. This contradict the second principle described in Section 2.1: we should only change one bit from "0" to "1" in each increment operation. To mitigate against this issue, we introduce additional molecular types called "prereactants." We also introduce "absence indicator" types to coordinate the transfer between prereactants and reactants.

3.1. *Prereactants and Absence Indicators*

We use the following notation to describe these concepts. For each bit i of the counter,

(1) Q_i is a **bit type** corresponding to i^{th} bit. (For our three-bit molecular counter, we have $Q_1 = X$, $Q_2 = Y$ and $Q_3 = Z$.)
(2) a_{qi} is an **absence indicator** type for type Q_i. (For our three-bit molecular counter, we have $a_{q1} = a_x$, $a_{q2} = a_y$ and $a_{q3} = a_z$.)
(3) Q_{pi} is a **prereactant** type for Q_i. (For our three-bit molecular counter, we have $Q_{p1} = X_p$, $Q_{p2} = Y_p$ and $Q_{p3} = Z_p$.)
(4) We set $X_p = X_{\text{inj}}$: the trigger type is the first prereactant.

All the absence indicators a_{qi} are produced continuously:

$$\varnothing \to a_{qi}; \tag{6}$$

Here the symbol $\varnothing$ as a reactant indicates that the reaction does not alter the quantity of the reactant types, perhaps because the quantity of these is large or replenishable. If Q_i is present, then its absence indicator a_{qi} is destroyed:

$$a_{qi} + Q_i \to Q_i. \tag{7}$$

However, if Q_i is absent, then a_{qi} persists, so its presence indicates the absence of Q_i, as required. If both a prereactant Q_{pi} and the absence indicator a_{qi} for the i-th bit are present, we produce type Q_i:

$$a_{qi} + Q_{pi} \rightarrow Q_i. \tag{8}$$

Finally, the prereactant $Q_{p(i+1)}$ for the $(i+1)$-st bit is produced if both the prereactant Q_{pi} and the type Q_i for the i-th bit are present:

$$Q_i + Q_{pi} \rightarrow Q_{p(i+1)}. \tag{9}$$

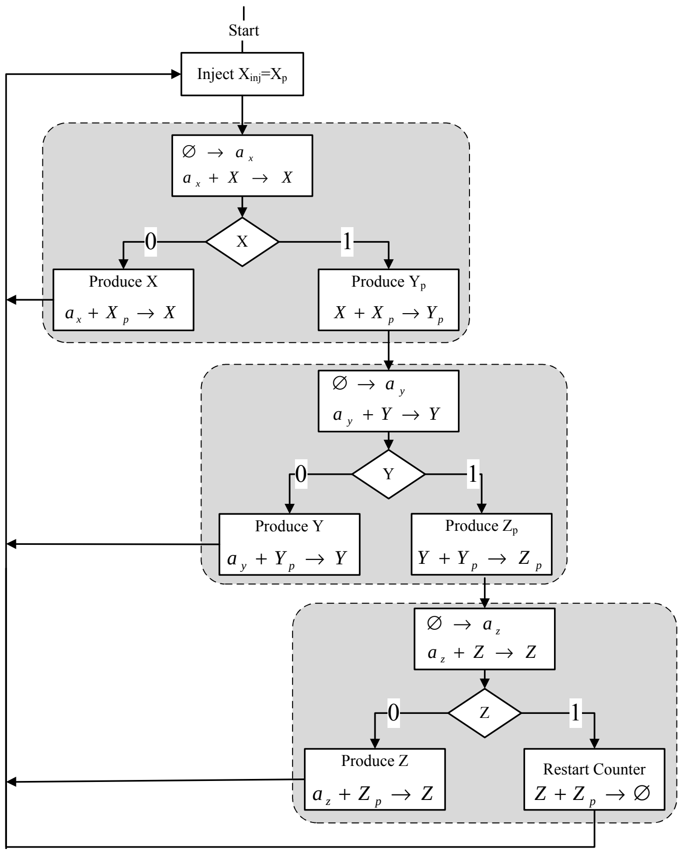
Start
Inject $X_{inj}=X_p$
$\varnothing \rightarrow a_x$
$a_x + X \rightarrow X$
X
0
1
Produce X
$a_x + X_p \rightarrow X$
Produce Y_p
$X + X_p \rightarrow Y_p$
$\varnothing \rightarrow a_y$
$a_y + Y \rightarrow Y$
Y
0
1
Produce Y
$a_y + Y_p \rightarrow Y$
Produce Z_p
$Y + Y_p \rightarrow Z_p$
$\varnothing \rightarrow a_z$
$a_z + Z \rightarrow Z$
Z
0
1
Produce Z
$a_z + Z_p \rightarrow Z$
Restart Counter
$Z + Z_p \rightarrow \varnothing$

Fig. 4. Modified algorithm for the molecular counter, with prereactants and absence indicators.

Table 1 lists the corresponding reactions for our three-bit counter in terms of the bit types X, Y and Z instead of generic Q_i's. Figure 4 shows a modified version of the flowchart in Figure 3, this time with prereactants and absence indicators.

3.2. *Three-Phase Synchronization*

Including absence indicators and prereactants establishes an order for the transfers of molecular quantities in the counter, but we need a mechanism to ensure that each transfer completes before the next one begins. As indicated on the left-hand side Figure 5, we must ensure that the accumulation or destruction of the absence indicator completes before the production of the bit type begins; in turn, we must ensure that the production of the bit type completes before the production of the next

Table 1. Reactions for the molecular counter, with prereactants and absence indicators.

#	Q_i	Z	Y	X
1	$\varnothing \rightarrow a_{qi}$ $a_{qi} + Q_i \rightarrow Q_i$	$\varnothing \rightarrow a_z$ $a_z + Z \rightarrow Z$	$\varnothing \rightarrow a_y$ $a_y + Y \rightarrow Y$	$\varnothing \rightarrow a_x$ $a_x + X \rightarrow X$
2	$a_{qi} + Q_{pi} \rightarrow Q_i$	$a_z + Z_p \rightarrow Z$	$a_y + Y_p \rightarrow Y$	$a_x + X_p \rightarrow X$
3	$Q_i + Q_{pi} \rightarrow Q_{p(i+1)}$	$Z + Z_p \rightarrow \varnothing$	$Y + Y_p \rightarrow Z_p$	$X + X_p \rightarrow Y_p$

prereactant begins; and so on. Similarly, as indicated on the right-hand side of Figure 5, we must ensure that the bit types X, Y and Z are not produced simultaneously.

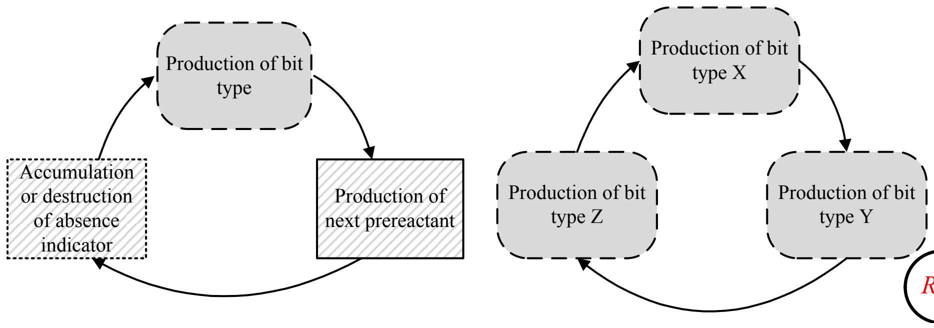

Fig. 5. Sequence of reactions for the molecular counter.

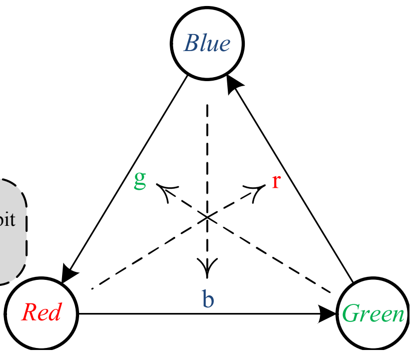

Fig. 6. The three-phase transfer scheme.

We must turn the two “dials” shown in Figure 5 simultaneously. To do so we use a three-phase synchronization protocol that we call “Red-Green-Blue” (RGB). This is illustrated in Figure 6. Reactions are “color coded” – that is to say assigned to one of the three categories. Quantities are transfered between color categories based the absence of types in the third category: red goes to green in the absence of blue; green goes to blue in the absence of red; and blue goes to red in the absence of green.

To elucidate the concept, here we introduce the reactions for RGB synchronization in isolation. In Section 3.3, we incorporate the principle into the design of the molecular counter. To implement the RGB synchronization, we introduce molecular types R, G and B. Computation cycles are implemented by transfering quantities among three types R, G and B, with following reactions:

$$\mathbf{b} + R \rightarrow G + \mathbf{b} \quad (10) \qquad \mathbf{r} + G \rightarrow B + \mathbf{r} \quad (11) \qquad \mathbf{g} + B \rightarrow R + \mathbf{g} \quad (12)$$

We generate “absence indicators” types **r**, **g** and **b** corresponding to R, G and B:

$$\begin{array}{c}\varnothing \xrightarrow{\text{slow}} \mathbf{r}\\ R + \mathbf{r} \xrightarrow{\text{fast}} R\end{array} \quad (13) \qquad \begin{array}{c}\varnothing \xrightarrow{\text{slow}} \mathbf{g}\\ G + \mathbf{g} \xrightarrow{\text{fast}} G\end{array} \quad (14) \qquad \begin{array}{c}\varnothing \xrightarrow{\text{slow}} \mathbf{b}\\ B + \mathbf{b} \xrightarrow{\text{fast}} B\end{array} \quad (15)$$

The absence indicators are continually generated. However, they only persist in the absence of the

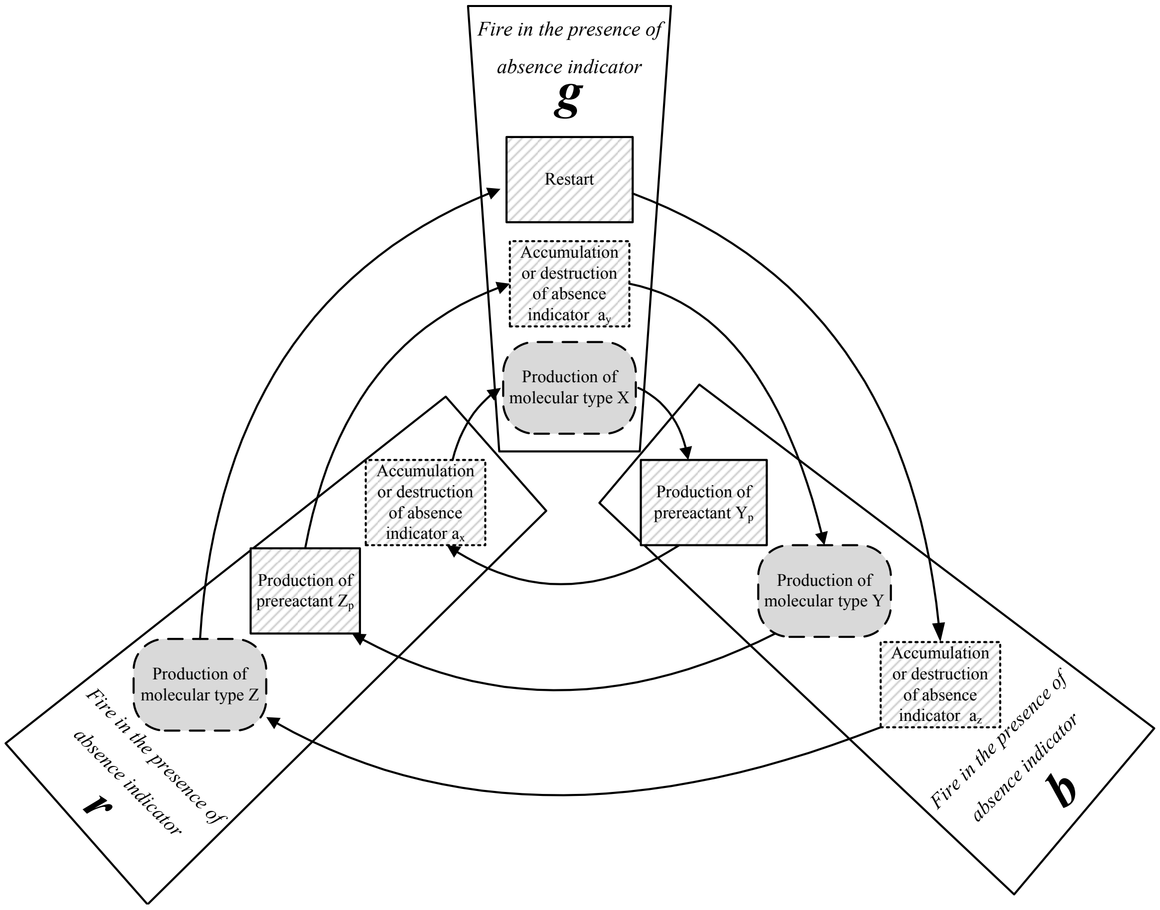
Fire in the presence of absence indicator g
Restart
Accumulation or destruction of absence indicator a_y
Production of molecular type X
Production of prereactant Y_p
Production of molecular type Y
Accumulation or destruction of absence indicator a_z
Fire in the presence of absence indicator b
Accumulation or destruction of absence indicator a_x
Production of prereactant Z_p
Production of molecular type Z
Fire in the presence of absence indicator r

Fig. 7. Combined diagrams for synchronization of reactions.

corresponding color-coded signals, since they are quickly consumed by signal molecules in their corresponding color categories. This feature assures that as long as any reaction in a given phase has not fired to completion, the succeeding phase cannot begin.

3.3. *The Molecular Binary Counter with RGB scheme*

Figure 7 shows the assignment operations to phases of the computation. Absence indicators **r**, **g** and **b** are used to initiate reactions in each phase. In lieu of the generic transfer Reactions 10– 11, we use transfer reactions that produce the absence indicators a_x, a_y and a_z for X, Y and Z, respectively:

$$\mathbf{r} + G \xrightarrow{\text{fast}} B + a_x + \mathbf{r} \quad (16) \qquad \mathbf{g} + B \xrightarrow{\text{fast}} R + a_y + \mathbf{g} \quad (17) \qquad \mathbf{b} + R \xrightarrow{\text{fast}} G + a_z + \mathbf{b} \quad (18)$$

This obviates the need for reactions of the form of Reaction 6 to generate a_x, a_y and a_z.

A set of reactions for the counter that incorporates the RGB transfer reactions is described in Figure 8. This is nearly the final design. However, we need a few more reactions to deal with accumulation of unused absence indicators, as well as a few to speed up the production of the absence indicators.

The Transfer reactions 16–18 supply absence indicators a_x, a_y and a_z in every RGB cycle. The scheme cycles continuously, irrespective of injections of X_{inj}. Accordingly, unused absence indicators a_x, a_y and a_z will accumulate. To mitigate against this, we include "**clean-up**" reactions initiated in the presence of corresponding absence indicator **r**, **g** and **b**:

$$\mathbf{b} + a_x \xrightarrow{\text{fast}} \mathbf{b} \quad (19) \qquad \mathbf{r} + a_y \xrightarrow{\text{fast}} \mathbf{r} \quad (20) \qquad \mathbf{g} + a_z \xrightarrow{\text{fast}} \mathbf{g} \quad (21)$$

As shown in Figure 8 the corresponding clean-up reactions always complete before the production of a_x, a_y and a_z begins. For instance, the absence indicator a_z is pushed into the system by Reaction 18 whenever the absence indicator **b** is present. Therefore, the clean-up Reaction 21 for a_z fires in the preceding RGB phase that was initiated in the presence of absence indicator **g**.

Note that the rates at which the reactions in each phase fire depend on the quantities of the corresponding absence indicators **r**, **g** and **b**. (Each phase is a separate column in Figure 8.) To ensure that the entire sequence of reactions in each phase completes, we add the following "**speed-up**" reactions. These reactions amplify the quantities of the absence indicators for each type when the other two types are present. In turn, this increases the rates of all the reactions in the phase.

$$R + G \xrightarrow{\text{slow}} R + G + \mathbf{b} \quad (22) \qquad G + B \xrightarrow{\text{slow}} G + B + \mathbf{r} \quad (23) \qquad B + R \xrightarrow{\text{slow}} B + R + \mathbf{g} \quad (24)$$

All reactions in Figure 8 fire at the "fast" rate, except the following reactions:[†]

$$\mathbf{r} + a_x + X \xrightarrow{\text{slow}} X + \mathbf{r}; \quad (25) \qquad \mathbf{g} + a_y + Y \xrightarrow{\text{slow}} Y + \mathbf{g}; \quad (26) \qquad \mathbf{b} + a_z + Z \xrightarrow{\text{slow}} Z + \mathbf{b}; \quad (27)$$

Table 2 shows the final set of RGB reactions and Table 3 shows the final set of reactions for X, Y and Z. Together, these comprise our complete design of the molecular counter.

Table 2. Final version of RGB reactions for the molecular counter.

Production of r, g, b	Destruction of r, g, b	Transfer reactions	Speed-up reactions	Clean-up reactions
$\varnothing \xrightarrow{\text{slow}} \mathbf{r}$	$R + \mathbf{r} \xrightarrow{\text{fast}} R$	$\mathbf{b} + R \xrightarrow{\text{fast}} G + a_z + \mathbf{b}$	$R + G \xrightarrow{\text{slow}} R + G + \mathbf{b}$	$\mathbf{b} + a_x \xrightarrow{\text{fast}} \mathbf{b}$
$\varnothing \xrightarrow{\text{slow}} \mathbf{g}$	$G + \mathbf{g} \xrightarrow{\text{fast}} G$	$\mathbf{r} + G \xrightarrow{\text{fast}} B + a_x + \mathbf{r}$	$G + B \xrightarrow{\text{slow}} G + B + \mathbf{r}$	$\mathbf{r} + a_y \xrightarrow{\text{fast}} \mathbf{b}$
$\varnothing \xrightarrow{\text{slow}} \mathbf{b}$	$B + \mathbf{b} \xrightarrow{\text{fast}} B$	$\mathbf{g} + B \xrightarrow{\text{fast}} R + a_y + \mathbf{g}$	$B + R \xrightarrow{\text{slow}} B + R + \mathbf{g}$	$\mathbf{g} + a_z \xrightarrow{\text{fast}} \mathbf{b}$

Table 3. Final version of reactions for molecular types X, Y and Z.

Accumulation or destruction of absence indicators	Production of molecules	Production of prereactant
$\mathbf{r} + a_x + X \xrightarrow{\text{slow}} X + \mathbf{r}$	$\mathbf{g} + a_x + X_p \xrightarrow{\text{fast}} X + \mathbf{g}$	$\mathbf{b} + X + X_p \xrightarrow{\text{fast}} Y_p + \mathbf{b}$
$\mathbf{g} + a_y + Y \xrightarrow{\text{slow}} Y + \mathbf{g}$	$\mathbf{b} + a_y + Y_p \xrightarrow{\text{fast}} Y + \mathbf{b}$	$\mathbf{r} + Y + Y_p \xrightarrow{\text{fast}} Z_p + \mathbf{r}$
$\mathbf{b} + a_z + Z \xrightarrow{\text{slow}} Z + \mathbf{b}$	$\mathbf{r} + a_z + Z_p \xrightarrow{\text{fast}} Z + \mathbf{r}$	$\mathbf{g} + Z + Z_p \xrightarrow{\text{fast}} \mathbf{g}$

[†]One might expect that these reactions should fire at the "fast" rate. However, there is always some leakage in the system; some amount of X, Y and Z is always present. These molecules of X, Y and Z consume molecules of the corresponding absence indicators, a_x, a_y and a_z. If Reactions 25–26 are "slow", a_x, a_y and a_z are still consumed, when this is necessary; but they are not completely consumed by trace amounts X, Y or Z that are present due to leakage, as they would be if the reactions were "fast".

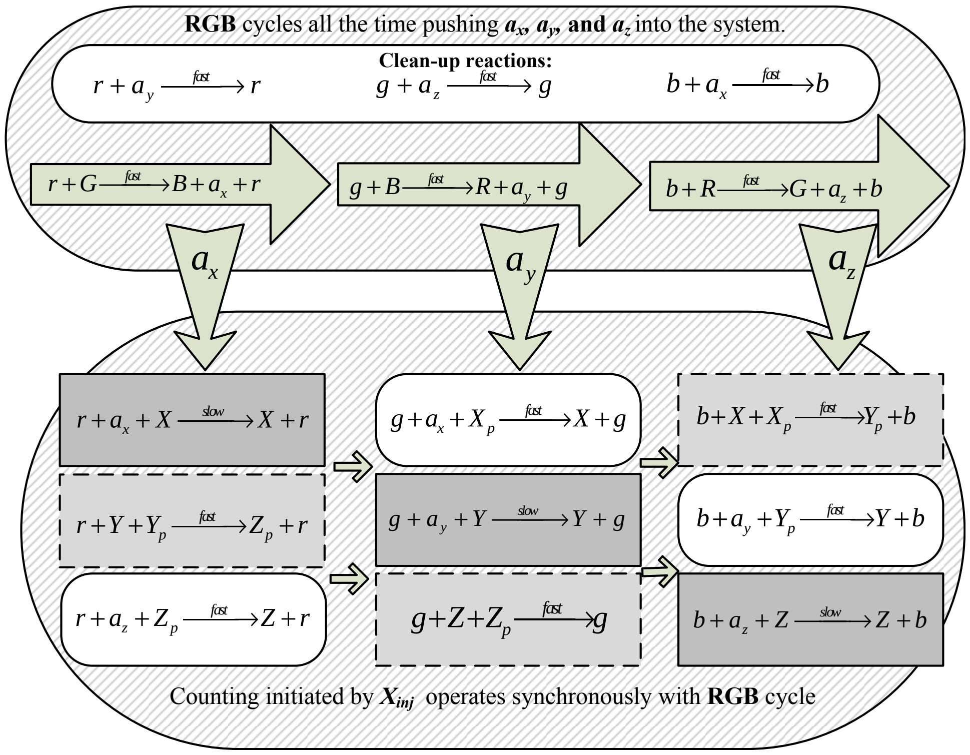
RGB cycles all the time pushing a_x, a_y, and a_z into the system.
Clean-up reactions:
$r + a_y \xrightarrow{fast} r$
$g + a_z \xrightarrow{fast} g$
$b + a_x \xrightarrow{fast} b$
$r + G \xrightarrow{fast} B + a_x + r$
$g + B \xrightarrow{fast} R + a_y + g$
$b + R \xrightarrow{fast} G + a_z + b$
a_x
a_y
a_z
$r + a_x + X \xrightarrow{slow} X + r$
$g + a_x + X_p \xrightarrow{fast} X + g$
$b + X + X_p \xrightarrow{fast} Y_p + b$
$r + Y + Y_p \xrightarrow{fast} Z_p + r$
$g + a_y + Y \xrightarrow{slow} Y + g$
$b + a_y + Y_p \xrightarrow{fast} Y + b$
$r + a_z + Z_p \xrightarrow{fast} Z + r$
$g + Z + Z_p \xrightarrow{fast} g$
$b + a_z + Z \xrightarrow{slow} Z + b$
Counting initiated by X_{inj} operates synchronously with **RGB** cycle

Fig. 8. Diagram for the molecular counter with RGB synchronization.

4. Simulation results

To evaluate our design, we performed stochastic simulation of the reactions in Table 2 and Table 3.[15] For the simulation, we assigned the molecular type G an initial amount that was greater than the amount of trigger type X_p. (Recall that we use X_p as the trigger type, so $X_{\text{inj}} = X_p$.) This is necessary: molecules of G produce molecules of a_x in Reaction 16; then molecules of a_x combine with molecules of X_p to produce X. So a sufficient quantity of G is needed to ensure that all of the injected X_p gets consumed during each increment operation.

Table 4 shows our parameters for the simulation. We perform 15 seperate injections of X_p. This corresponds to two full cycles through the 8 values of the binary counter. We expect to see the sequence of binary values:

$$000 \to 001 \to 010 \to \cdots \to 111 \to 000 \to 001 \to \cdots \to 110 \to 111.$$

Here the rates of all the "slow" reactions are unity. The rate separation specifies the rate of the "fast" reactions relative to that of "slow" reactions. We show results for a rate separation of four orders of magnitude.

Table 4. Parameters for the simulation.

Amount X_{inj}	Initial amount G	Time between injections	Number of injections	Rate separation	Trajectories
50	1000	100	15	10,000	750

The data from stochastic simulation is shown in Table 5. When analyzing it, we stipulate the following:

(1) If the amount of X, Y or Z is greater than 70% of the injected amount of X_p, then the corresponding bit is logically "1."
(2) If amount of X, Y or Z is less than 30% of the injected amount of X_p, then the corresponding bit is logically "0."

So, for injected amounts of X_p of 50 molecules, amounts of X, Y and Z greater than 35 molecules correspond to "1" and amounts less than 15 molecules correspond to "0." As can be seen, the mean values of X, Y and Z are well within the correct range; the counter is functioning as expected. For the first 8 increment operations, 97.17% of the bits are correct. For the total 15 increment oerations 94.02% are correct. We note that as the counter continues to cycle, the standard deviation increases. Eventually, significant errors will occur.

Table 5. Statistical data from the stochastic simulation of the molecular counter for 15 successive increment operations; 750 trajectories were generated.

# Injection	Binary number	Mean Z	Mean Y	Mean X	Stand. dev. Z	Stand. dev. Y	Stand. dev.X
Initially	000	0	0	0	0	0	0
1	001	0	0.0040	49.9920	0	0.0632	0.1263
2	010	0.1800	49.2800	0.6667	0.7757	3.9982	7.3336
3	011	0.3493	49.3653	49.7760	1.5641	2.6002	2.0832
4	100	46.0640	1.7080	0.7013	8.8421	9.0444	5.6212
5	101	46.0360	1.6893	49.4533	8.7042	8.7215	3.1116
6	110	45.4080	47.4453	1.6000	8.6243	7.2505	9.5936
7	111	44.9213	47.4920	49.3147	9.0950	6.2478	4.8469
8	000	4.2293	2.9013	1.6000	8.7295	8.6207	8.2601
9	001	4.0493	2.8387	49.0213	8.6718	9.1556	6.1392
10	010	4.1573	46.5040	2.4053	8.2922	8.9769	11.3972
11	011	4.1827	45.9413	49.0240	8.5032	9.7176	8.9244
12	100	40.4827	4.7307	3.4880	12.4410	12.3965	15.8317
13	101	39.9747	4.6480	49.1680	12.9882	13.9143	11.2223
14	110	38.3138	45.7290	4.0507	13.5609	12.9068	16.3682
15	111	37.6128	45.1789	48.9292	13.8860	14.5067	12.5843

5. Discussion

We have demonstrated the design of a molecular counter that is relatively robust and rate independent. Given only rate categories of "slow" and "fast, the computation is exact and independent of the specific reaction rates. In particular, it doesn't matter how fast any "fast" reaction is relative to another, or how slow any "slow" reaction is relative to another – only that "fast" reactions are fast relative to "slow" reactions. Throughout the paper, the exposition was in terms of a three-bit counter. In future work, we will generalize the construction to n bits. We will also strive for more robust counting over long time periods. (In preliminary work, we have shown that further layers of "clean-up" operations can achieve this.)

We comment that our design of a molecular counter could be applied for the task of counting cell divisions. This task is important for the analysis of aging and perhaps for the detection of cancer, where cell divisions runs rampant. Also, our design might find applications in biochemical sensing and drug delivery.

In this work, our contribution was to tackle the problem of synthesizing computation at an abstract level – working not with actual molecular types but rather with arbitrary types (a, b, c, etc.). We are exploring the mechanism of DNA-strand displacement advocated by Erik Winfree's group at Caltech as an experimental chassis for our method.[5] They have shown that the kinetics of arbitrary chemical reactions can be implemented through DNA strand-displacement reactions. They provide an assembler that accepts a set of biochemical reactions with nearly any rate structure and delivers the corresponding DNA sequences for the displacement reactions. Reaction rates are controlled by designing sequences with different binding strengths; the binding strengths are controlled by the length and sequence composition of toeholds. Our contribution can be positioned as the "front end" of the design flow; the DNA assembler and experimental chassis described by these authors constitute the "back-end."

References

1. D. Widmaier, D. Tullman-Ercek, E. Mirsky, R. Hill, S. Govindarajan, J. Minshull, and C. Voigt, "Engineering the Salmonella type III secretion system to export spider silk monomers," *Molecular Systems Biology*, vol. 5, no. 309, pp. 1–9, 2009.
2. M. Sedlak and N. Ho, "Production of ethanol from cellulosic biomass hydrolysate using generically engineered yeast," *Applied Biochemistry and Biotechnology*, vol. 114, no. 1-3, pp. 403–416, 2004.
3. D. Ro, E. Paradise, M. Ouellet, K. Fisher, K. Newman, J. Ndungu, K. Ho, R. Eachus, T. Ham, M. Chang, S. Withers, Y. Shiba, R. Sarpong, , and J. Keasling, "Production of the antimalarial drug precursor artemisinic acid in engineered yeast," *Nature*, vol. 440, pp. 940–943, 2006.
4. D. Endy, "Foundations for engineering biology," *Nature*, vol. 438, pp. 449–453, 2005.
5. D. Soloveichik, G. Seelig, and E. Winfree, "DNA as a universal substrate for chemical kinetics," *Proceedings of the National Academy of Sciences*, vol. 107, no. 12, pp. 5393–5398, 2010.
6. L. Lavagno, G. Martin, and L. Scheffer, *Electronic Design Automation for Integrated Circuits Handbook*. CRC Press, 2006.
7. P. Senum and M. Riedel, "Rate-independent biochemical computational modules," in *Proceedings of the Pacific Symposium on Biocomputing (submitted)*, 2011.
8. A. Shea, B. Fett, M. Riedel, and K. Parhi, "Writing and compiling code into biochemistry," in *Proceedings of the Pacific Symposium on Biocomputing*, 2010, pp. 456–464.
9. H. Jiang, M. Riedel, and K. Parhi, "Digital signal processing with biomolecular reactions," in *IEEE Workshop on Signal Processing Systems*, 2010.
10. M. Gibson, "Computational methods for stochastic biological systems," Ph.D. dissertation, California Institute of Technology, 2000.
11. D. Soloveichik, M. Cook, E. Winfree, and J. Bruck, "Computation with finite stochastic chemical reaction networks," *Natural Computing*, vol. 7, no. 4, 2008.
12. M. Cook, D. Soloveichik, E. Winfree, and J. Bruck, "Programmability of chemical reaction networks," in *Algorithmic Bioprocesses*, A. Condon, D. Harel, J. Kok, A. Salomaa, and E. Winfree, Eds. Springer, 2009, pp. 543–584.
13. D. Gillespie, "Stochastic simulation of chemical kinetics," *Annual Review of Physical Chemistry*, vol. 58, pp. 35–55, 2006.
14. L. Nagel and D. Pederson, "Simulation program with integrated circuit emphasis," in *Midwest Symposium on Circuit Theory*, 1973.
15. D. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.