

# Species Minimization in Computation with Biochemical Reactions

Ruei-Yang Huang<sup>1</sup>, De-An Huang<sup>2</sup>, Hui-Ju Katherine Chiang<sup>1,3</sup>, Jie-Hong R. Jiang<sup>1</sup>,  
and François Fages<sup>3</sup>

<sup>1</sup>Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan

<sup>2</sup>Academia Sinica, Taiwan

<sup>3</sup>EPI Contraintes, INRIA Paris-Rocquencourt, France

{b97901166@ntu.edu.tw, b97901010@ntu.edu.tw, b97901184@ntu.edu.tw, jhjiang@cc.ee.ntu.edu.tw,  
Francois.Fages@inria.fr}

Engineering biochemical reactions for computational purposes is a common pursue in synthetic biology [1]. In such design tasks, molecular species have to be carefully engineered to ensure modularity and orthogonality [3, 7], and are scarce resources. Minimizing the number of involved molecular species is crucial to accomplish a complex computation within a confined biochemical environment. This report aims at species minimization by reusing modular and regular reactions in an asynchronous time-multiplexed fashion. Our method enhances not only species utility, but also re-programmability and robustness in realizing various logic circuits. A case study demonstrates the ease of design in realizing general logic computation, and simulation confirms the feasibility and robustness of the proposed method.

The quest for deciphering nature’s design principle of living organisms has been the primary subject in systems biology. This goal cannot be fully achieved unless human engineers can construct complex biological systems from scratch. A synthetic approach to biology aims to create complex systems bottom up from elementary components [1]. In such construction, a convenient way of modeling biological systems is from a computational aspect regarding system states in terms of molecular concentrations. Despite the intrinsic *hybrid* nature (involving both continuous and discrete state evolutions) of a biological system, the dominating *digital* design methodology of electronic circuits can be systematically carried to designing computation with biochemical reactions [2, 5].

Building upon our prior work [2], we address a fundamental limitation of biological circuit design. Unlike electronic circuit design, where a logic gate can be freely instantiated into several copies to compose a complex logic function, in a biological circuit every signal corresponds to a distinct molecular species, and every instantiation of a logic gate “consumes” molecular species. As molecular species have to be carefully engineered to ensure modularity and orthogonality for proper operation [3, 7], they are expensive resources. It raises an important question, how to minimize molecular species involved in achieving a target computational task.

## Methods

We focus on logic computation, which is well studied in circuit synthesis, and without loss of generality assume a given (combinational) logic circuit consists of purely two-input NAND gates. (As a NAND gate is functionally complete, it can be used universally to construct arbitrary Boolean functions.) Moreover, following [2], we consider computation with biochemical reactions under the classical chemical kinetic (CCK) model for simplicity. (Our construct also works under the stochastic model.)

To minimize molecular species in realizing a given logic

circuit, we intend to reuse a single NAND processor for all NAND operations using a time multiplexing strategy (a well-known technique in reconfigurable computing). The NAND operations are executed in a topological order according to the circuit structure. In order to fulfill the reuse of the NAND processor, we introduce a three-valued NAND operation with biochemical reactions for asynchronous handshaking without a synchronizing clock signal. The primary inputs and intermediately computed signals are properly placed and shifted in an asynchronous linear pipeline (consisting of asynchronous shift registers) for iterative NAND processing. For each NAND operation, its two inputs are taken from the first two positions of the pipeline. Once the two data tokens are consumed by the NAND processor, an output token is produced and/or placed in proper pipeline position(s) according to the state of a *program counter*, which tracks which NAND gate of the given logic circuit is under processing. The data tokens in the pipeline are asynchronously shifted forwards for the next NAND processing. When all the NAND gates of the given circuit are executed, a done signal signifies its environment to load to the pipeline next input data. The procedure repeats accordingly. The technicalities are detailed as follows.

**Three-valued logic abstraction:** In the conventional Boolean model of biochemical systems, high and low concentrations of a molecule represent logic values 1 and 0, respectively. We extend the binary domain to the three-valued abstraction  $\mathbf{T}_\delta$  with respect to two parameters  $\delta$  and  $\epsilon$  for  $\delta \gg \epsilon$  with

$$\mathbf{T}_\delta(X) = \begin{cases} 1, & \text{if } |[X] - 2\delta| < \epsilon \\ 0, & \text{if } |[X] - \delta| < \epsilon \\ \text{NULL}, & \text{if } [X] < \epsilon \end{cases}$$

where  $[X]$  denotes the concentration of molecule  $X$ . In essence, this extension distinguishes NULL (denoting no data, with concentration  $\approx 0$ ), logic value 0 (with concentration  $\approx \delta$ ), and logic value 1 (with concentration  $\approx 2\delta$ ). It thus allows signal completion detection for asynchronous operation.

**Register transfer:** Let the linear pipeline consist of  $n$  register pairs in order  $\mathbf{r}^0, \mathbf{r}^1, \dots, \mathbf{r}^n$ , each corresponds to two inputs to be processed by the NAND processor. A register is essentially a molecule, whose concentration corresponds to the stored data. The NAND processor takes its inputs from  $\mathbf{r}^0$ . The data content of a register pair  $\mathbf{r}^{i+1}$  is shifted asynchronously to  $\mathbf{r}^i$  as long as  $\mathbf{r}^i$  contains no data token.

The asynchronous register transfer is accomplished in the following steps to ensure correctness. In the following code (see [2] for detailed program constructs), let  $R_i$  represent the species of  $\mathbf{r}^i$ , let  $C_0$  be a flag such that  $[C_0] > \delta$  signifies it is safe to start register transfer from each  $R_{i+1}$  to its temporary buffer  $R_{i+1_{\text{buf}}}$ . Once such transfers are done for all  $i$ , it is safe to start transfer from  $R_{i+1_{\text{buf}}}$  to  $R_i$ . The transfer activation condition of  $C_0$  is controlled by the iterative NAND operation to be detailed. (Assume the code is under sequential execu-

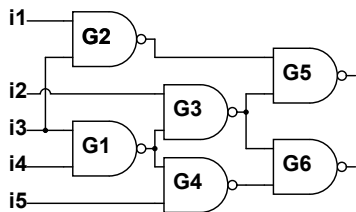


Figure 1: C17 circuit

tion, and reaction rates and *preconditions* [2] are omitted for brevity.)

```

begin
01  if ([C0] > ε)
03    while (∃ i, [Ri+1] > ε)
02      Ri+1 → Ri+1buf      i = 0, 1, 2, ..., n
04      C0 → C1buf
05      C1buf → C1
07    while (∃ i, [Ri+1buf] > ε)
06      Ri+1buf → Ri      i = 0, 1, 2, ..., n
08      C1 → C0buf
end

```

**Iterative NAND operation:** An iteration of NAND operation,  $Z = \text{NAND}(X, Y)$ , can be described by the following reaction code. (Assume the code is under sequential execution, and reaction rates and preconditions are omitted.) The reactions can be divided into two phases. In the first phase, value of  $Z$  is computed; in the second phase, the value of  $Z$  is exported to proper locations of the pipeline or to the output, and register transfer is activated.

```

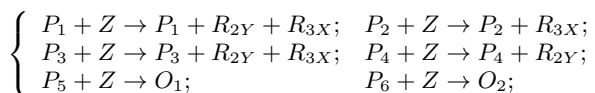
initial [S] = θ (3δ < θ < 4δ), [B] = δ, [A] = 0, [C0] = 2δ
begin
01  while [A] < ε
02    S + X → Stmp
03    S + Y → Stmp
04    if ([X] + [Y] > θ)
05      B → Z + A
06    else
07      B → 2Z + A
08    X → ∅
09    Y → ∅
10    C0 → C1buf
11    C1buf → C1
12    Export(Z)
13    Stmp → S
14    C1 → C0buf
15    C0buf → C0
16    A → B
end

```

**Pipeline writing:** When the output of the  $i^{\text{th}}$  NAND operation is an input of the later  $j^{\text{th}}$  NAND operation, it should be stored at register  $\mathbf{r}^{j-i}$  because the input of the  $j^{\text{th}}$  NAND operation should arrive at  $\mathbf{r}^0$  after further  $j - i$  shiftings. Note that the writing positions may vary depending on the state of the program counter.

### Case Study

We elaborate our method through the benchmark circuit C17, which consists of six NAND gates as shown in Figure 1. The gates  $G_1, G_2, \dots, G_6$  are in a topological order. Let species  $P_1, P_2, \dots, P_6$  serve as the program counter such that  $[P_i] < \epsilon$  if and only if  $G_i$  is not under current processing. The following reactions are added to control pipeline writing.



where  $Z$  is the output of the NAND processor,  $R_{iX}$  ( $R_{iY}$ ) is the  $X$ -end ( $Y$ -end) register of  $\mathbf{r}^i$ , and  $O_1$  and  $O_2$  correspond to the two outputs of C17. The ODE simulation result of SBW simulator [4] (under input assignments  $i_1 = 0, i_2 = 0, i_3 = 0, i_4 = 0, i_5 = 0, i_6 = 0$ ) is plotted in Figure 2.

### Analysis and Discussion

In our realization, a three-valued NAND gate operation mainly consists of 13 reactions and 11 species (excluding absence indicators and their generation reactions); similarly an asynchronous register transfer mainly consists of 12 reactions and 6 species. If a circuit consisting of  $n$  two-input NAND gates

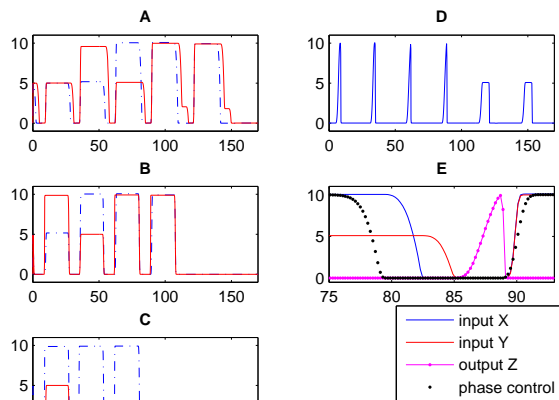


Figure 2: Simulation result: A. Waveforms for the two registers of  $\mathbf{r}^0$ ; B. Waveforms for  $\mathbf{r}^1$ ; C. Waveforms for  $\mathbf{r}^2$ ; D. Waveform for  $Z = \text{NAND}(X, Y)$ ; E. Detailed waveforms in the range of 75 ~ 93 time units.

are “hardwired” directly,  $13n$  reactions and  $11n$  species are required (excluding absence indicators and their generation reactions). On the other hand, our proposed approach requires  $12(n - 1) + 13$  reactions and  $6(n - 1) + 11$  species. Although our approach may require longer computation time than hardwired implementation, in comparison species limitation is often a more critical issue than delay in biochemical systems.

The advantages of our proposed architecture are three-fold: First, it allows NAND operation to be reused and reduces the number of required molecular species. Second, the NAND processor and the registers always take the same species as input. The reaction structure is very regular, and the corresponding reactions are easy to engineer. Third, expanding the array of shift registers is straightforward, and the computation procedure can be easily re-programmed to implement different circuits. Moreover, as the proposed operations are all asynchronous, our method is robust against delay variations.

### Conclusion

For universal logic computation, we proposed a three-valued NAND operation and a linear shift register architecture for asynchronous time multiplexing. Compared to hardwired logic netlists, our method reduces substantially the number of required species, provides insensitivity against delay variations, and admits re-programmability in realizing various logic circuits with minor modifications. For future chemical or biological realization, DNA strand displacement reactions [6] or intracellular genetic reactions [3] could be studied. Computation beyond the logic-gate formulation, e.g., by the dynamics of biochemical reactions, awaits future study.

### REFERENCES

- [1] E. Andrianantoandro, S. Basu, D. Karig, and R. Weiss. Synthetic biology: New engineering rules for an emerging discipline. *Molecular Systems Biology*, 2006.
- [2] D.-A. Huang, J.-H. R. Jiang, R.-Y. Huang, C.-Y. Cheng. Compiling program control flows into biochemical reactions. In *Proc. Int'l Conf. on Computer-Aided Design (ICCAD)*, pp. 361-368, 2012.
- [3] T. S. Moon, C. Lou, A. Tamsir, B. Stanton, and C. Voigt. Genetic programs constructed from layered logic gates in single cells. *Nature*, 491(11):249-253, 2012.
- [4] Systems Biology Workbench (SBW): <http://sbw.sourceforge.net/>
- [5] P. Senum and M. Riedel. Rate-independent constructs for chemical computation. *PLoS ONE*, 6(6), 2011.
- [6] D. Soloveichik, G. Seelig, and E. Winfree. DNA as a universal substrate for chemical kinetics. *Proc. of the National Academy of Sciences*, 107(12):5393-5398, 2010.
- [7] B. Wang, R. Kitney, N. Joly, and M. Buck. Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology. *Nature Communications*, 2011.