Lin *et al. Journal of Biological Engineering* (2018) 12:14
https://doi.org/10.1186/s13036-018-0109-4

Journal of Biological Engineering

RESEARCH

Open Access

Synthesis of control unit for future biocomputer

Chun-Liang Lin*, Ting-Yu Kuo and Wei-Xian Li

Abstract

Background: Synthesis of a variety of biological circuits for specific functional purposes has made a tremendous progress in recent years. The ultimate goal of combining molecular biology and engineering is to realize a functional biocomputer. To address this challenge, all previous efforts work toward building up the bio-computer as the ultimate goal. To this aim, there should be a key module, named control unit (CU), to direct a serious of logic or arithmetic operations within the processor.

Methods: This research task develops a bio-CU to work with a bio-ALU, which is realized from the combination of previously developed genetic logic gates to fulfill the kernel function of CPU as those done in the silicon computer.

Results: A possible framework of the bio-CPU has demonstrated how to connect a bio-CU with a bio-ALU to conduct a fetch-decode-execute cycle of a macro instruction. It presents not only capability of 4-bit full adder but coordination of related modules in biocomputer.

Conclusions: We have demonstrated computer simulation for applications of the genetic circuits in biocomputer construction. It's expected to inspire follow-up study to synthesize potential configurations of the future biocomputer.

Keywords: Synthetic biology, Genetic logic circuit, Arithmetic and logical unit, Control unit

Background

Synthesis of a variety of biological circuits for specific functional purposes has made a tremendous progress in recent years. The ultimate goal of combining molecular biology and engineering is to realize the biocomputer [1–3]. In particular, synthesis of fundamental Boolean logic gates [4] and design of genetic oscillator [5, 6] have been successfully or partly successfully reached since the earliest research effort focused on creating oscillating behavior of a genetic circuit in 2000 by Elowitz and Leibler [7]. Basically, a biological circuit is different from its counterpart in the digital logic circuit; the former uses chemical reaction of gene expression to simulate on and off states of the protein concentration [8] with DNA-binding proteins and DNA-binding factors expressing protein concentration for a specific logical function [9]. With the rapid development of synthetic biology, a class of combinational and sequential genetic logic circuits [10–12] have been developed toward specific applications. Fast grow of synthetic biology is because most of the biochemical reactions can be described in terms of mathematical models through the use of non-linear Hill differential equations [13] and computer simulation before conducting real-world experiments.

All of the efforts work toward building up a functional bio-computer as the ultimate goal [14, 15]. To this aim, there should be a key module, named control unit (CU), to direct a serious of logic or arithmetic operations within the processor. It guides the computer's memory, arithmetic/logic unit (ALU) and input/output devices to respond macro instructions by arranging timing and control signals.

Following our previous works [16–21], we continue to develop a bio-CU working with the bio-ALU, which is realized from the combination of previously developed genetic logic gates to fulfill the kernel function of CPU as in the silicon computer. A standard CPU consists of three cores: CU, ALU and memory. The ALU is triggered by the CU when it receives instructions. An instruction cycle including "fetch-decode-execute" is the standard operating procedure of the CPU. That is, fetching data from memory, decoding instruction into executable commands and executing the instruction by the ALU. The result is next

* Correspondence: chunlin@dragon.nchu.edu.tw
Department of Electrical Engineering, National Chung Hsing University, Taichung 402, Taiwan

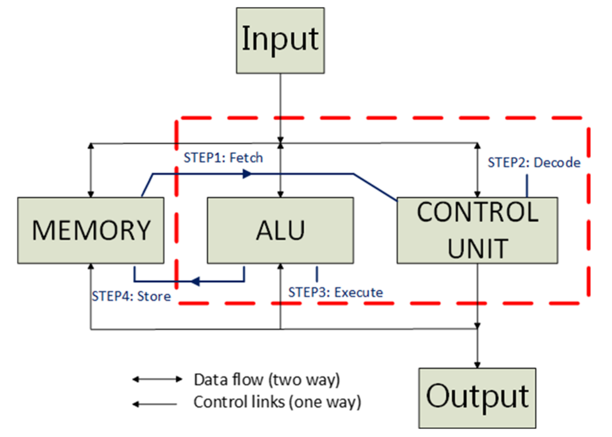


Fig. 1 Instruction flow inside the biocomputer. It shows the instruction flow that operates in the biocomputer, the process of the Decode and Execute are operated in the ALU and CU

stored in a temporary register for further processing. Once the process is completed, the CPU sets itself up to wait for the next instruction to come.

In this research task, we propose a possible framework of the bio-CPU to demonstrate how to connect a bio-CU with a bio-ALU to conduct a fetch-decode-execute cycle of a macro instruction (for the demonstration purpose, shown here is only a half function of the instruction cycle), see Fig. 1. The paper presents not only the capability of 4-bit full adder but the coordination of related modules in a biological computer.

Method

Design of bio-register

In the silicon CPU, the inherent registers are used to temporarily store data while the computational kernel conducting logic or arithmetic operation. In general, there are instruction register, program counter, temp register, and accumulator insider a CPU. Inside of a one-bit register, there is an edge-triggered D-type flip-flop, which constitutes the fundamental unit of the register. A D-type flip-flop is made up of a NOT gate, two AND gates and four NAND gates.

Fundamental genetic logic gate

In the traditional electronic circuits, one uses several Boolean logic gates to realize a CPU. Before constructing a biocomputer in the genetic system, we start here by constructing a class of genetic logic gates. The genetic expression consists of the transcription and translation processes [16]. When the enzyme RNA polymerase (RNAp) is restricted to the relevant promoter, the DNA transcript is converted to messenger RNA (mRNA), and the transcription rate is controlled by transcription factors (TF). Assembling a variety of standard biological sites, including reporters of the coding regions and transcription factors, RNA, promoters, can lead to functional realization of a fundamental genetic logic gate [9].

The mathematical model describing the biochemical response of the genetic system consists of a set of two differential equations is given by [7].

$$\begin{aligned} \dot{m}_i &= \alpha_i f_i(u) - \gamma_{m_i} m_i + \alpha_{i,0}, \\ \dot{p}_i &= \beta_i m_i - \gamma_{p_i} p_i, i = 1, \ldots, L \end{aligned} \tag{1}$$

where m_i and p_i represents, respectively, concentrations of mRNA and protein of the gene i, γ_{m_i} and γ_{p_i} represent, respectively, degradation rates of mRNA and protein, α_i is transcription rate of m_i, β_i is synthesis rate of p_i, $\alpha_{i,\ 0}$ is basal rate, $f_i(\cdot)$ denotes the promoter activity function used to describe the nonlinear transcriptional reactions, and u is concentration of TF from other inducers to control the gene expression.

A gene with an operator site can bind to a repressor or activator TF, and the promoter activity of the genetic logic NOT can be described as

$$f_{\mathrm{NOT}}(u) = \frac{1}{1 + \left(\frac{u}{K}\right)^n} \tag{2}$$

where f_{NOT} is a promoter activity function for logical NOT, u is concentration of the repressor or activator TF, n is Hill coefficient representing cooperatively effect between TF and the corresponding operator, and K is Hill constant. The binding site is inserted into the promoter region of the target gene. For the genetic expression of the logic NOT, the gene is activated by binding to the repressor TF in the corresponding operon, and its state is inactivated by binding the repressor TF. The framework is shown in Fig. 2a.

A gene with two manipulation sites can bind two repressors TFs or activator TFs, and the promoter activity of the genetic logic AND gate can be described as

$$f_{\mathrm{AND}}(u_1, u_2) = \frac{\left(\frac{u_1}{K_1}\right)^{n_1}\left(\frac{u_2}{K_2}\right)^{n_2}}{1 + \left(\frac{u_1}{K_1}\right)^{n_1} + \left(\frac{u_2}{K_2}\right)^{n_2} + \left(\frac{u_1}{K_1}\right)^{n_1}\left(\frac{u_2}{K_2}\right)^{n_2}} \tag{3}$$

where f_{AND} is a logic AND promoter activity function [22], u_1 and u_2 are concentrations of repressor or activator TFs, K_1 and K_2 are Hill constants of u_1 and u_2, respectively, $n_1$and n_2 are the corresponding Hill coefficients. For the logic AND gate, the protein is produced only in the presence of two TFs, see Fig. 2b. The NAND gate can be implemented by cascading a NOT gate and a AND gate, see Fig. 2c.

A combination of biological registers may include at least NOT, AND, NAND, and some other

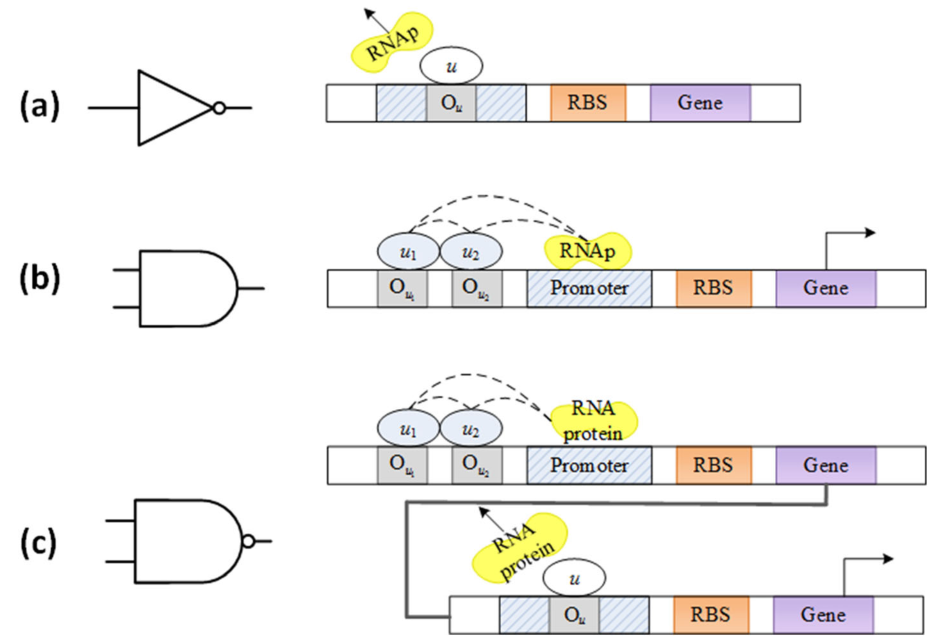


Fig. 2 Structure of the fundamental genetic logic gates. **a** NOT gate **b** AND gate **c** NAND gate. Figure **a**, **b**, and **c** represent the genetic sequences for expressing the logic functions, respectively

fundamental logic gates. One is referred to details of the creature from [13]. For practical realization, heterogeneous regulation can be used to synthesize an AND gate in *Escherichia coli*. The AND gate includes the coactivating genes hrpR and hrpS controlled by the promoter input, and the output depends on the σ54-dependent hrpL promoter. When the ribosome binding site (RBS) serves as is used as a linear amplifier, it is applied to regulate protein expression levels. It is next to cascade a NOT gate that is assembled with the cI/Plam repressor module containing the f lambda gene cI and PR promoters.

By the above description, a modular combinational NAND gate can be generated as illustrated in Fig. 3. To establish the simplified model, we focus only on the steady state behavior of the mRNA as

$$m_i = \frac{\alpha_i}{\gamma_{m_i}} f_i(u) + \frac{\alpha_{i,0}}{\gamma_{m_i}} \tag{4}$$

$$\dot{p}_i = \alpha_{P_i} f_i(u) - \gamma_{P_i} p_i + \alpha_{P_0,i} \tag{5}$$

The protein dominated response is given by the dynamic eq. (5) where the parameters $\alpha_{P_i} = {\alpha_i \beta_i}/{\gamma_{m_i}}$ and $\alpha_{P_0,i} = {\alpha_{i,0} \beta_i}/{\gamma_{m_i}}$. Established in the abbreviated dynamic equation and Fig. 3, the dynamic equation for

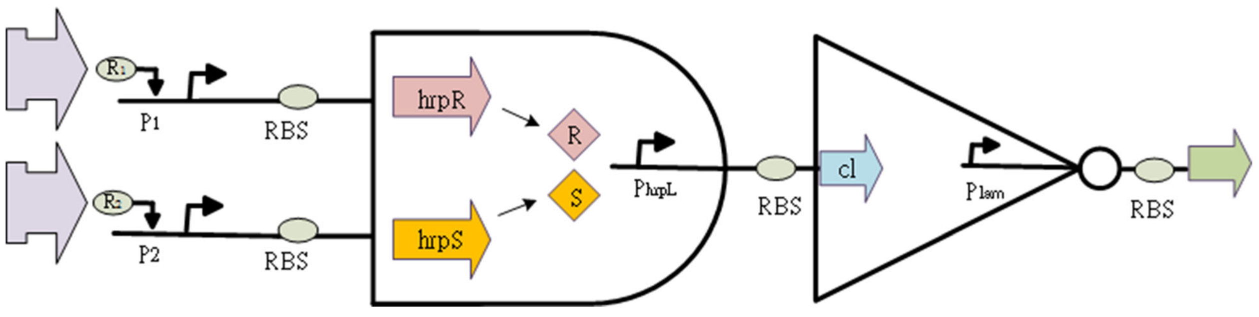


Fig. 3 Combinational genetic NAND gate. The structure illustrates the NAND gate in the biology

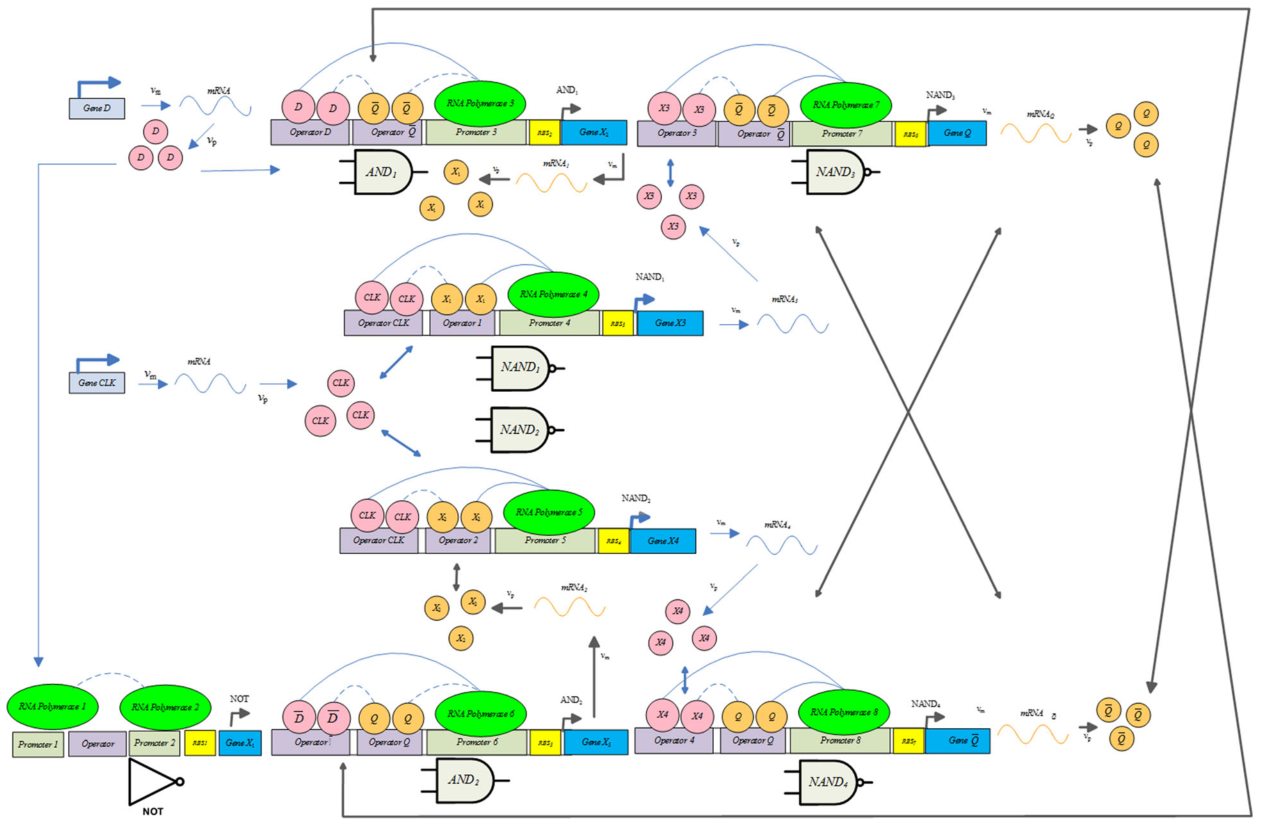
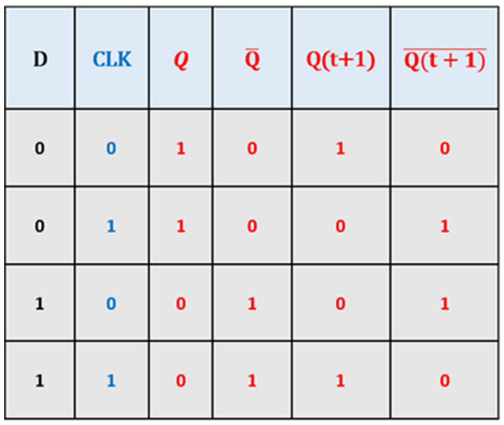
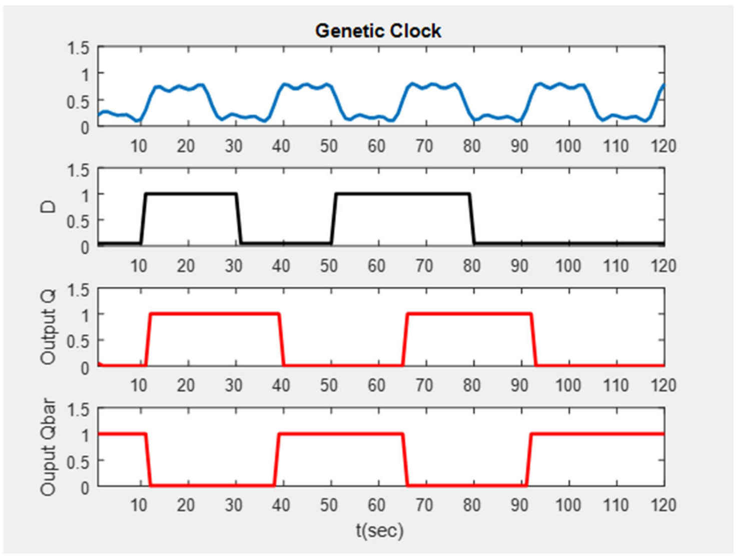


Fig. 4 Schematic diagram of the biological D-type FF. This figure shows the D-type FF structure that is synthesized by the corresponding genetic sequences



D	CLK	Q	$\bar{Q}$	Q(t+1)	$\overline{Q(t+1)}$
0	0	1	0	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	1	1	0

(a)



(b)

Fig. 5 a Truth table of D-FF. **b** Normalized response of the biological D-FF. **a** The truth table uses 1 and 0 to represent, respectively, high and low concentration levels. **b** The output is generated by the genetic circuit that uses protein concentration to represent the state response

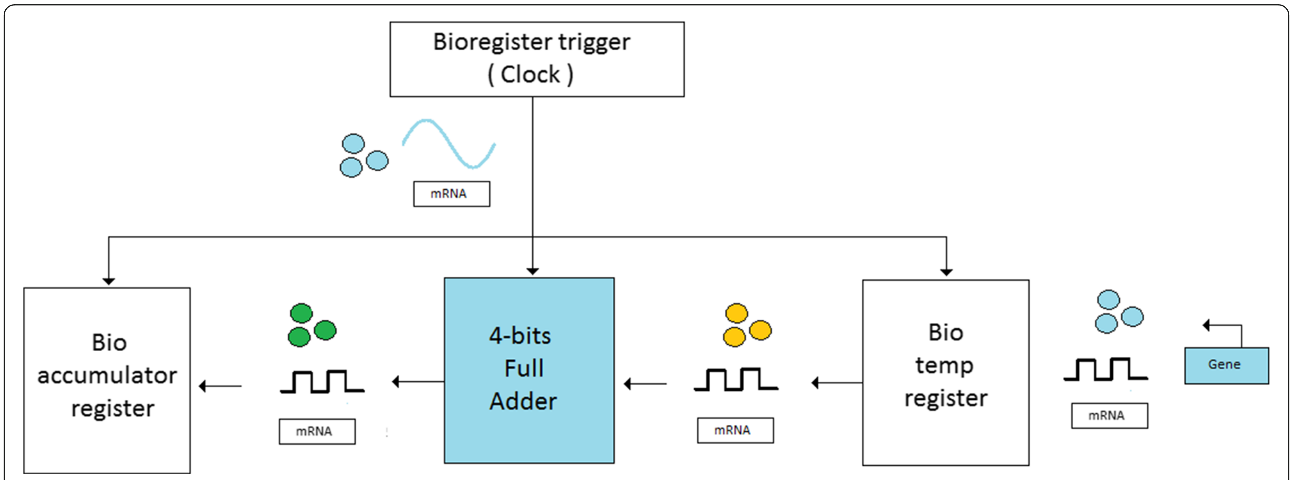


Fig. 6 Bio-ALU schematic diagram of the operating process. It illustrates the operating flow of the bio-ALU. First, the executed data is stored to the temp register. Second, the data is propagated to the full adder for execution. Finally, the processed data is propagated to the bio-accumulator and stored in the bio-accumulator. All of the operating steps is synchronized to the clock

characterizing the behavior of the genetic NAND is obtained as

$$\begin{aligned} \dot{p}_{AND} &= \alpha_P f_{AND}(u_1, u_2) - \gamma_P p_{NAND} + \alpha_{P_0,AND}, \\ \dot{p}_{NOT} &= \alpha_P f_{NOT}(p_{AND}) - \gamma_P p_{NOT} + \alpha_{P_0,NOT}, \\ p_{AND}(u_1, u_2) &= p_{NOT} \end{aligned} \tag{6}$$

The correct behaviour of $p_{AND}(u_1, u_2)$ confirming the truth table of a NAND gate can be easily observed by considering the steady-state response of $p_{AND}(u_1, u_2)$ described by (6).

Before proceeding with the system design, it should first be remarked that the system architecture adopted in this research is based on assembling several fundamental logical gates and circuits we published in the previous works [10, 17, 22] without making significant changes. All of the parameters used in the gates or circuits can be found from these references.

Synthesis of genetic D Flip flop (D-FF)

D-FF consists of several biological logic gates with a clock input as a state triggering command. Figure 4 illustrates implementation of the equivalent biological circuit. As shown in the figure, a similar conclusion in the digital circuit can be expected by adding a high or low protein concentration as the input.

$$\begin{aligned} \dot{p}_{AND1} &= \alpha_P f_{AND}(D, \overline{Q}) - \gamma_P p_{AND1} + \alpha_{P_0,AND1}, \\ \dot{p}_{AND2} &= \alpha_P f_{AND}(p_{NOT}(D), Q) - \gamma_P p_{AND2} + \alpha_{P_0,AND2}, \\ Q &= p_{NAND3}(p_{NAND1}(p_{AND1}, clock), \overline{Q}), \\ Q &= p_{NAND4}(p_{NAND2}(clock, p_{AND2}), Q) \end{aligned} \tag{7}$$

The edge-triggered D-FF model can be represented using the Hill differential eq. (7) where p_{NOT} is generated by $\dot{p}_{NOT} = \alpha_P f_{NOT}(\cdot) - \gamma_P p_{NOT} + \alpha_{P_0,NOT}$, D is the protein concentration input, the clock is a periodic biological signal. Simulation results show the results of the truth table of D-FF in Fig. 5a. The normalized output response of the biological D-FF is shown in Fig. 5b. When the clock signal goes from low to high, the leading edge triggers the subsequence response

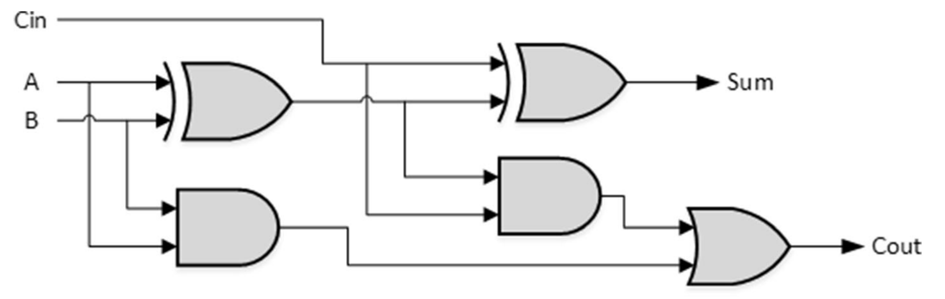


Fig. 7 Synthesis of a full adder based on two half-adders. The structure is shown by the logic gates in electronics which is constructed by two half-adders

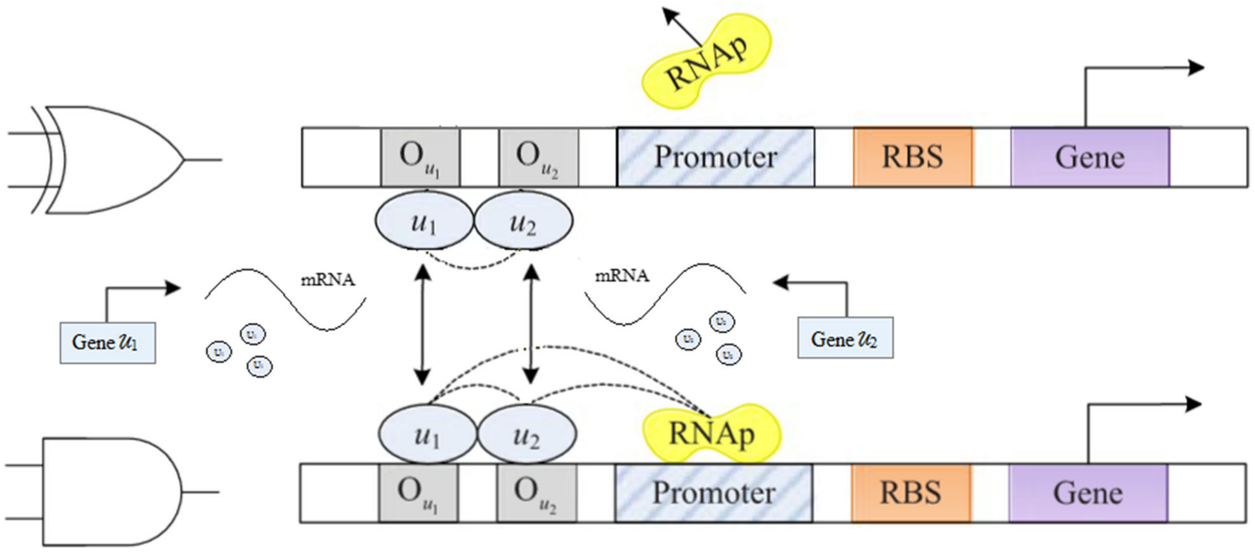


Fig. 8 Genetic half-adder structure. The structure is constructed by the corresponding gene sequences

where the clock signal can be generated by a genetic clock source [18, 21]. From this point of view, the stimulated concentration level changes at a time when intensity of the clock signal is enough to stimulate an accurate response. On the other hand, it is found that the signal exhibits a time delay compared to the ideal response ion the electronic circuit. This is acceptable in the biological systems because biochemical reactions are fairly slow.

Bio arithmetic logic unit

An ALU in the electronic circuit performs arithmetic and logical operations on the operands from the computer instruction word, such as addition or subtraction. A series of biological registers have been established in the previous work [16], and the biological arithmetic architecture has been proposed to develop basic biological computers. The simplified architecture is shown in Fig. 6. The three genetic

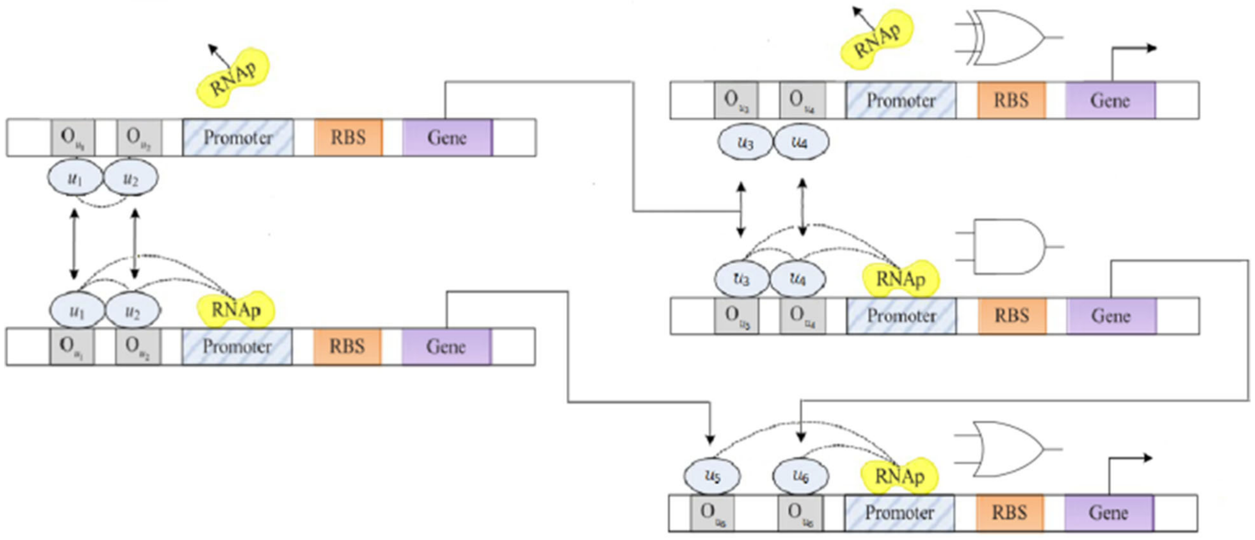
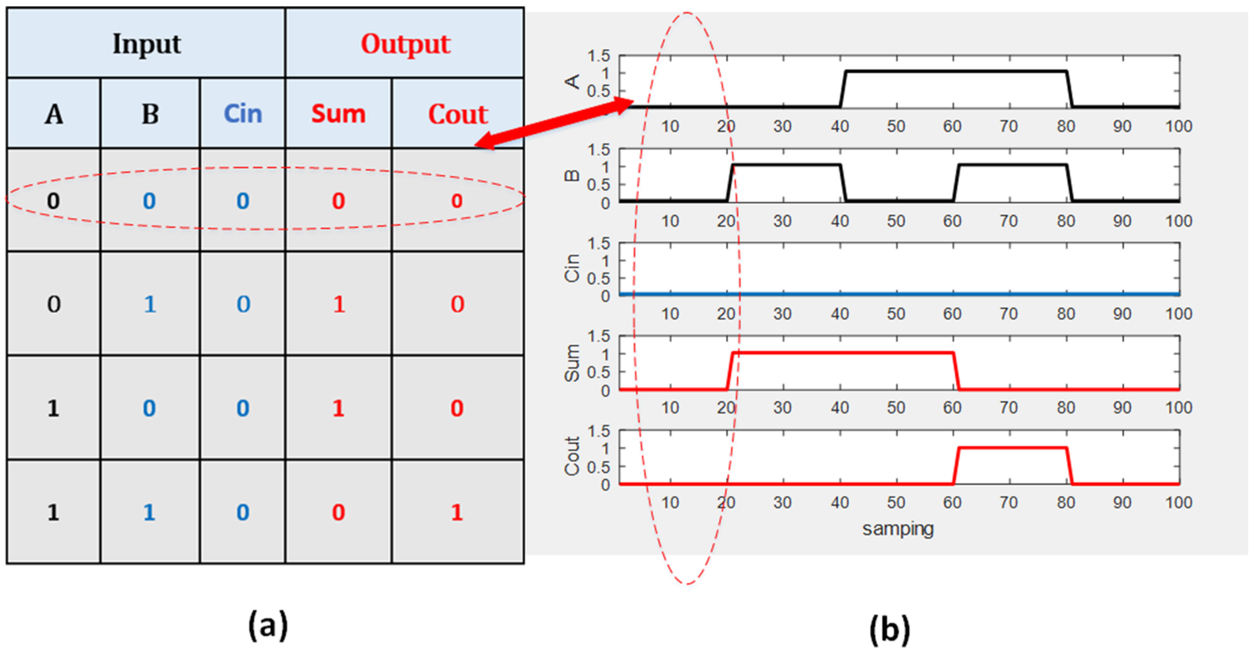


Fig. 9 Genetic 1-bit full adder structure. The structure is constructed by the corresponding gene sequences



Input			Output	
A	B	Cin	Sum	Cout
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1

Fig. 10 a Truth table of one bit full adder. **b** Genetic one bit full adder responses. **a** The table shows the result. **b** The result is produced by the genetic circuits with the dashed ellipsoid emphasizing the corresponding binary status

circuits and the corresponding clock pulses are connected together. Temporary registers and genetic accumulators are created by 4-bit parallel input and parallel output (PIPO) genetic registers. In the middle is a 4-bit genetic full adder for arithmetic operations. This model is used to implement fundamental arithmetic operations in the simplest bio-computer. Here, we only focus on implementing basic full-addition arithmetic operations without considering data acquisition and storage.

Genetic full adder

In the electronic circuit, mathematical addition is fulfilled by a full adder involves two half-adder in series [23]. The combination of two molecule inputs and a variety of genetic logic gates can be found from [15]. A genetic half-adder can be produced in a single mammalian cell, which produces a different reaction between erythromycin and phloretin. Through the use of natural or synthetic cell-cell communication [24], one can create a pattern formation based on functional modules. Following the previous development, we can use two gene-controlled binary adders to construct a gene binary full adder shown as in Fig. 7. As stated, a one bit genetic full-adder is fulfilled by cascading two genetic half-adders and an OR gate with the promoter activity functions for XOR gate within the half adder and the OR gate given by [25] where f_{OR} and f_{XOR} in the following equations are the promoter activity functions for logic OR and XOR, respectively. Definitions of all parameters follow those defined in (3).

$$f_{XOR}(u_1,u_2)=\frac{\left(\frac{u_1}{K_1}\right)^{n_1}+\left(\frac{u_2}{K_2}\right)^{n_2}}{1+\left(\frac{u_1}{K_1}\right)^{n_1}+\left(\frac{u_2}{K_2}\right)^{n_2}+\left(\frac{u_1}{K_1}\right)^{n_1}\left(\frac{u_2}{K_2}\right)^{n_2}} \tag{8}$$

$$f_{OR}(u_1,u_2)=\frac{\left(\frac{u_1}{K_1}\right)^{n_1}+\left(\frac{u_2}{K_2}\right)^{n_2}+\left(\frac{u_1}{K_1}\right)^{n_1}\left(\frac{u_2}{K_2}\right)^{n_2}}{1+\left(\frac{u_1}{K_1}\right)^{n_1}+\left(\frac{u_2}{K_2}\right)^{n_2}+\left(\frac{u_1}{K_1}\right)^{n_1}\left(\frac{u_2}{K_2}\right)^{n_2}} \tag{9}$$

The output of the first half-adder will be the input to the second half-adder. Referring directly to the half-adder structure shown in Fig. 8, it can be seen that a transcription factor was used as input for transcription of the gene. A one-bit full adder is realized in Fig. 9. Based on the truth table of full adder we conduct simulation study with the results shown in Fig. 10a-b. A four-bit full adder is a direct extension of one-bit version which can be created by concatenating four one-bit genetic full adders together, see Fig. 11. Performing the simulation confirms the correct result of the system where the input augend, addend, and the output response are shown in Fig. 12a-c.

Bio control unit

CU is a kernel of CPU that is responsible for all operations being carried out. The memory, registers and ALU will wait until CU directs the system to execute instructions. The structure of CU system can be as illustrated

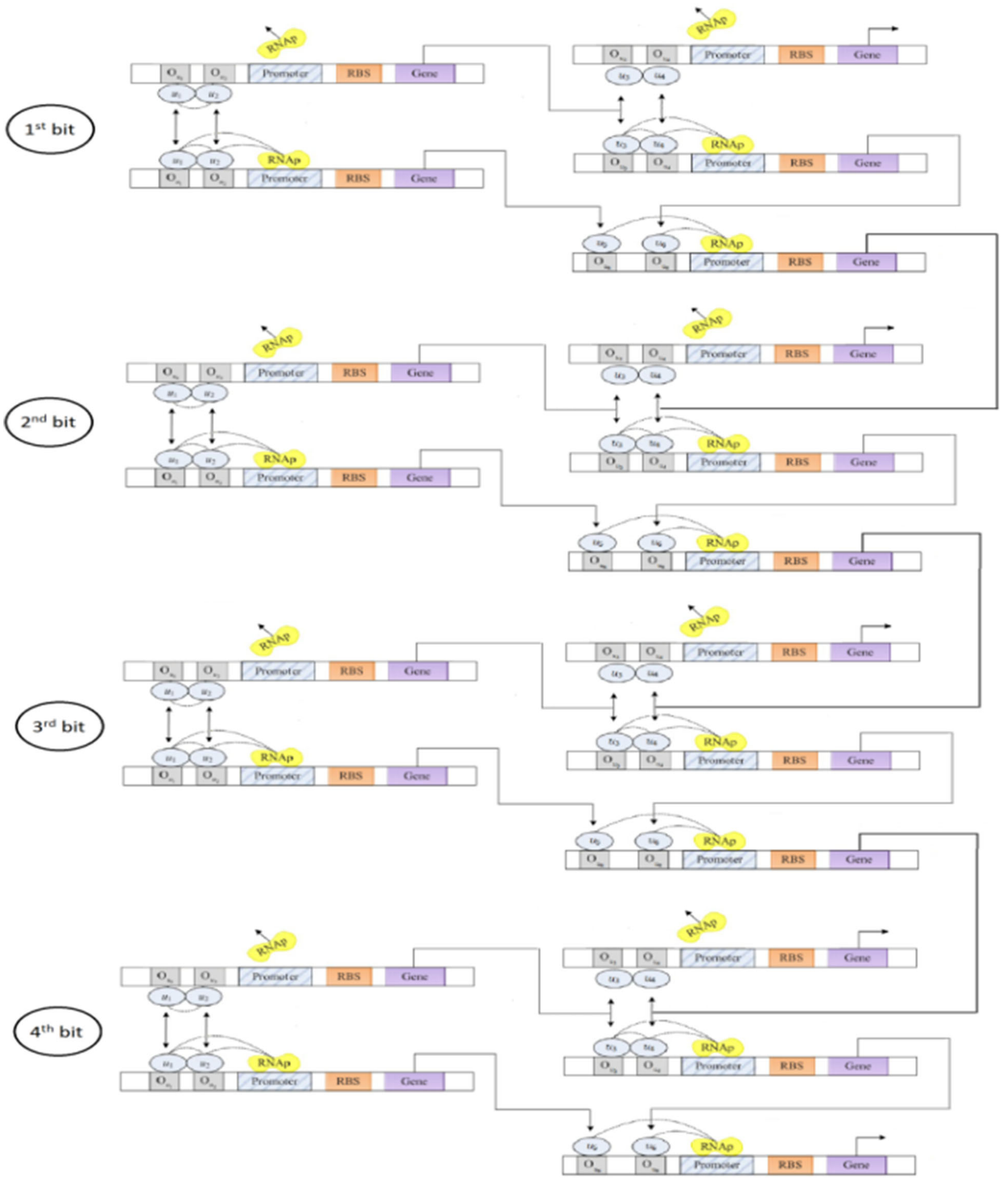


Fig. 11 Structure of a genetic four-bit full adder. The structure is assembled by four one-bit full adders with a cascaded connection

in Fig. 13, in which a ring counter is responsible for what instruction is the next in the operational sequence. The inputs of CU come from the instruction decoder which determine the instruction, such as MOV, ADD or SUB, to be executed. Once the instruction is received from the instruction decoder, the CU will allocate the corresponding outputs.

In the biological terms, accompanied by a genetic clock generator to generate clock signals, all of the data is originally stored in a bio-memory [26] until the

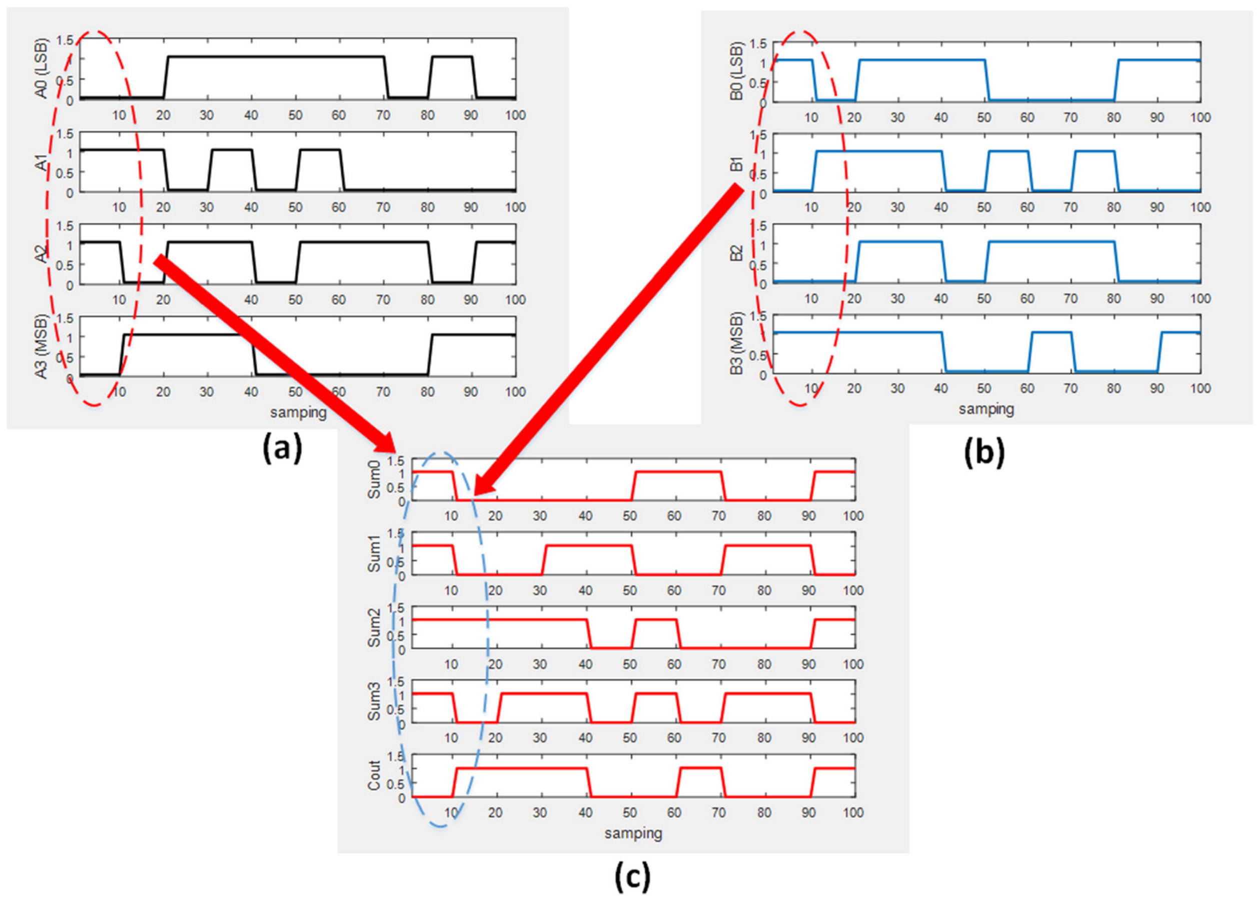


Fig. 12 **a** Protein concentration of the four-bit augend. **b** Protein concentration of the four-bit addend. **c** Output response of the four-bit genetic full adder. The dashed ellipsoid line explains the result of addition of the augend and the addends of the first clock cycle

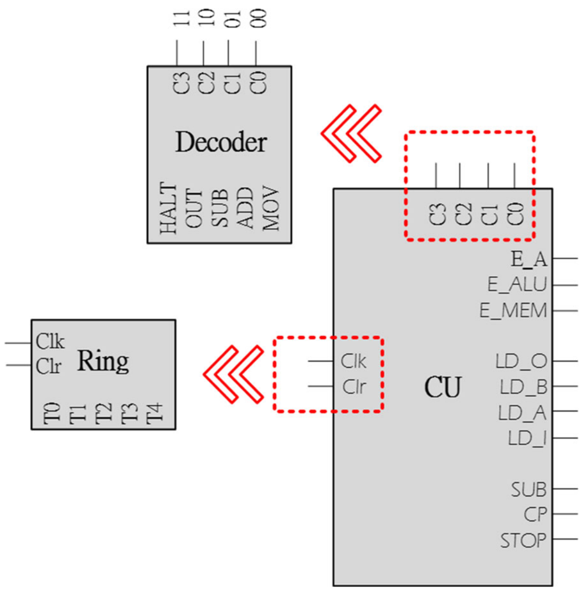


Fig. 13 Structure of the bio-control unit. The structure works with a ring counter and the decoder

bio-CU fetch and put it in a bio-temp register. Next, the data stored at the bio-temp register is moved to a genetic-full adder for execution. After the 4-bit genetic full adder accomplishes the task, the result is first moved to a genetic accumulator and next moved back to the bio-memory. The bio-CU plays the role of a commander among all functional modules. Details of all modules are described in the follows.

Bio ring counter

The bio-ring counter here is composed of five D-type FFs. It is an application of a shift register with the major difference being output of the last FF connected to input of the first FF, see Fig. 14. We use the ring counter as a counter to decompose the clock into four phases for instruction execution.

Let's consider the demonstration result of the ring counter shown in Fig. 15. The counter starts as the first clock comes, T0 is triggered by the first clock pulse, and T1 follows T0 while T0 goes from high to low. The process (T0 to T4) goes through 5 FFs until the last FF (T4) starts over. This five bits ring counter goes through

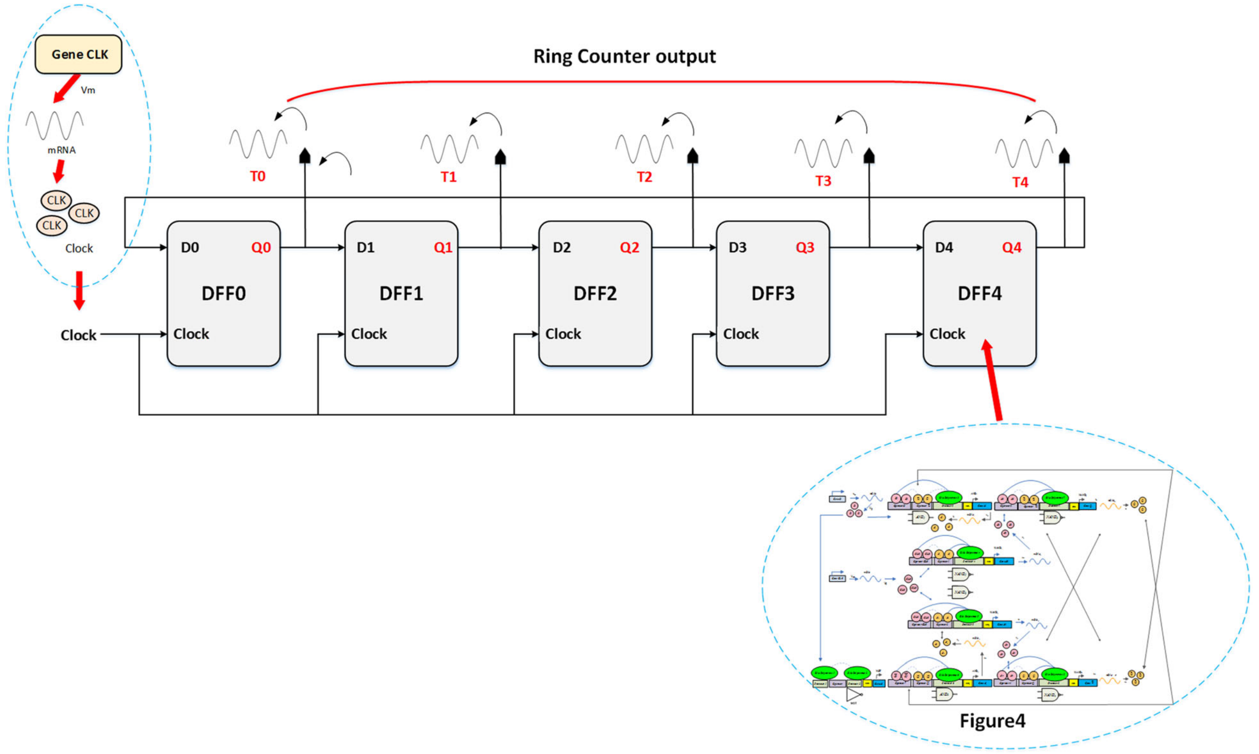


Fig. 14 Structure of the bio-ring counter. The five-bit ring counter is cascaded by connecting five D type FFs with feedback from the last FF output to the first FF input. All FFs are triggered simultaneously by the genetic clock. It works like a shift-register with feedback

5 sequences within the cycle: 00001, 00010, 00100, 01000, and 10,000.

Bio-temporary register and bio-accumulator

The function of temporary register is to save data fetched from memory. An accumulator is a register for temporarily saving data for or after calculation from the arithmetic and logical unit (ALU). Temporary register and accumulator are also called as the operand registers because they offer operands to ALU.

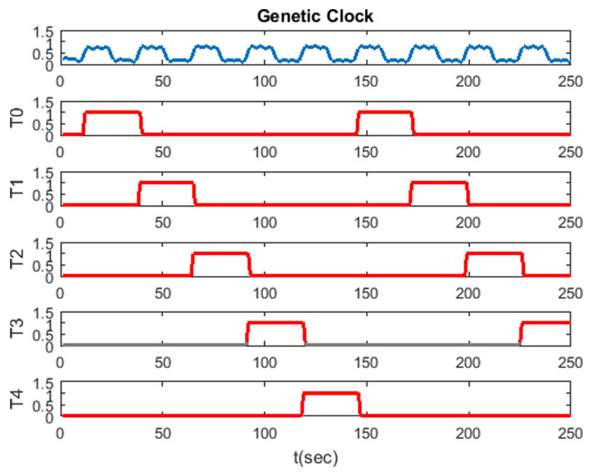


Fig. 15 Simulation result of the bio-ring counter. The result shows the changing concentration level for each D type FF in the bio-ring counter. Each triggering state of the bio-ring counter has its specific purpose depicted in Table 2

We refer to Intel 4004 for construction of the basic 4-bit computer architecture. In the computer architecture, data store and fetch are important steps, so for the 4-bit register, we synthesize it by combining several genetic D-FFs, which can be realized during the biochemical reaction. The action of data read appears at the rising edge of the clock.

The reaction of the sequential process should be synchronized with the clock signal. When the trigger's strength of the clock signal is sufficient to stimulate the exact response, the concentration level of the D-FF output follows.

For the PIPO bio-register, the 4-bit data enters in the parallel manner and is transferred in parallel to the corresponding outputs Qa to Qd in a clock pulse. Its schematic diagram is illustrated in Fig. 16.

The CU directs output of the adder to the accumulator, and data to be operated is placed in the temporary register where the 4-bit PIPO configuration is adopted to construct temporary register and accumulator which is triggered by a clock signal generator to synchronize state transmission.

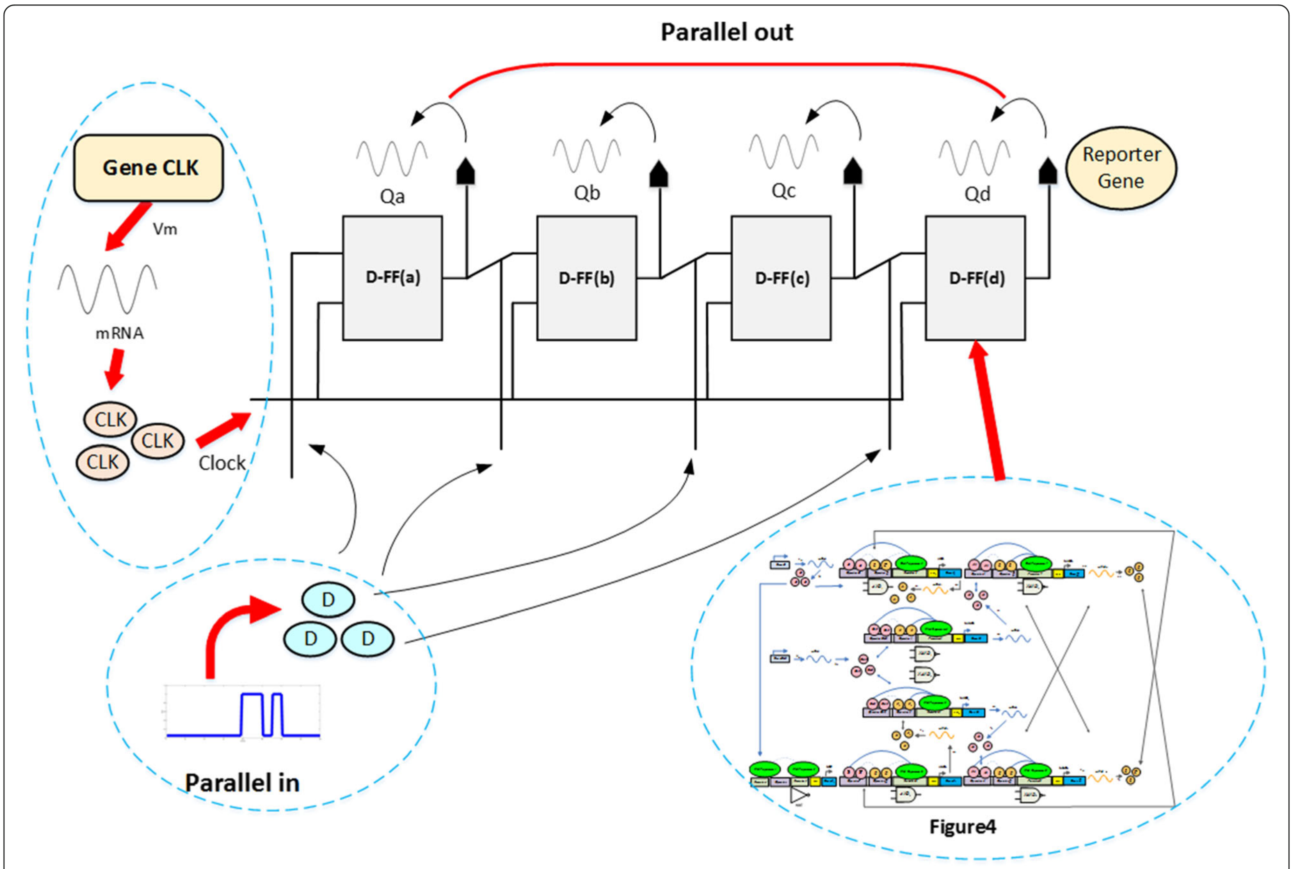


Fig. 16 Schematic diagram of the PIPO register. The four-bit PIPO register is cascaded by connecting four D type FFs with each FF accepting data input simultaneously

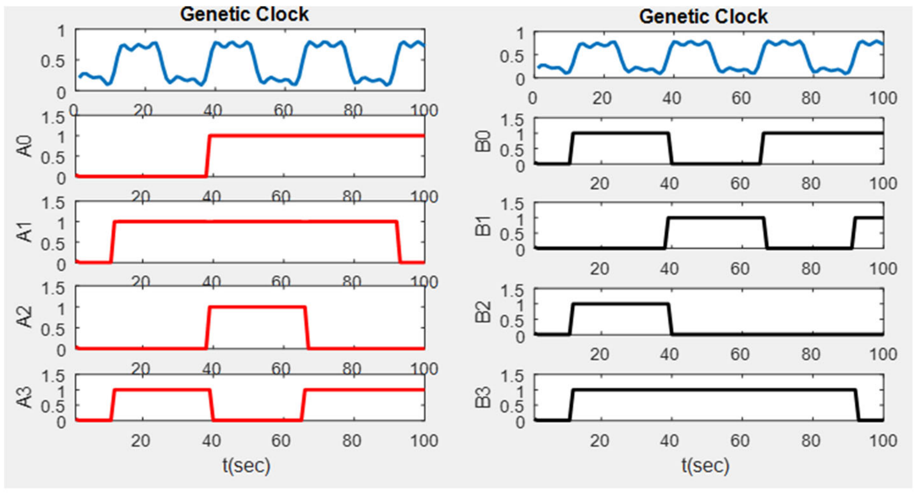
A full 4-bit addition requires five clocks to fulfill the entire work. When the instruction is triggered, all data will be placed in the register first. After the instruction being executed, the resulting data will be left in the accumulator. However, the data still needs to be stored back to the memory after execution. Figure 17a shows the bit states of the two temp registers. Figure 17b shows the resulting states of the 4-bit accumulator with a carry after performing addition.

Decoding of instruction

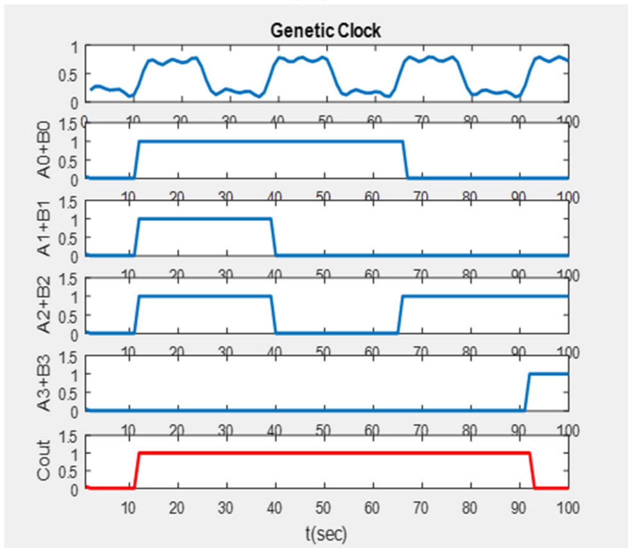
To fulfil a specific instruction, the CU needs to determine the steps required to complete the instruction generated by the instruction decoder which reads instruction from memory. The instruction decoder takes data stored in the instruction register and decodes it. The circuit of the instruction decoder for five fundamental logic/arithmetic functions is illustrated in Fig. 18.

The CU system in the digital computer is established by the output of instruction decoder collocated with a ring counter and several logic gates, see Fig. 19. An instruction enters the CU system while the instruction decoder decodes the instruction. Once the instruction decoder decodes the instruction and sends a corresponding output, the CU system carries out the required steps with the ring counter to enable the selected unit. The SUB is completed in five pulses by the ring counter. From T0 to T4, the CU enables the registers inside the CU system, see Table 1 for the working sequences to conduct the arithmetic subtraction (SUB).

Table 2 shows the execution steps of the CU. Using SUB as the example. First of all, at the time instant T0, the instruction to be executed is loaded from memory to the instruction register (IR), then the program counter keeps the address of the instruction being executed at T1, and the instruction loads data to the register A from memory at T2. At T3, the program counter holds the memory address of the next instruction-the subtraction operation from the register A and accumulator. Finally, at T4, the CU executes the instruction SUB and keeps data in the accumulator. The steps for accomplishing an arithmetic subtraction are accomplished under the command



(a)



(b)

Fig. 17 **a** Two genetic register output responses. **b** Bio-accumulator output responses. **a** Changes of concentration in the two genetic registers are data to be added together. **b** The bio-accumulator produces the result after performing binary addition

directed by the CU which decomposes instruction to several data movements between memory and ALU.

Simulation result

We demonstrate partial function of the instruction cycle which includes acquiring instruction from the instruction decoder and triggers ALU at the proper timing. Using the instruction SUB as an example, once the CU receives the instruction SUB from the instruction decoder, the individual steps to ensemble the instruction are activated sequentially. Fulfilment of a SUB instruction consists of five steps as illustrated in Fig. 20. The 5-stage ring counter repeats every five clock pulses and the ALU is triggered at T4, see Fig. 20. The ALU is triggered when T4 goes from low to high at $t =$ 120, see Fig. 21. The data is stored back to the ACC once ALU fulfils the operation.

In terms of biological circuits, the bio-CU and bio-ALU are created under synthetic biology. All of the circuits are composed of several genetic logic gates which use protein concentration as the signal input or output, the output reaction resembles individual behaviour of a variety of logic gates. The clock signal can be generated by a genetic clock generator which can be synthesized from an oscillator with a toggle switch. While responses of all functional modules presented here are not perfect as those of their counterparts in the silicon computer, the preliminary

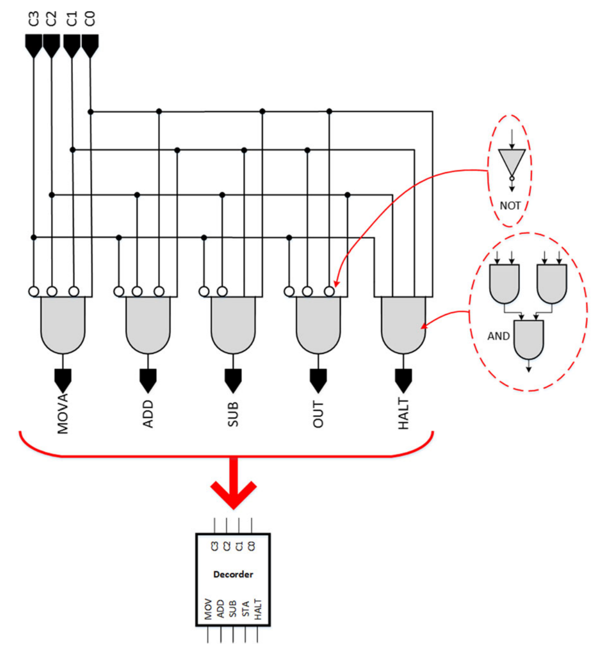


Fig. 18 Instruction decoder. The decoder determines the instruction sets to be executed by the bio-CPU

Table 1 Sequence for realizing an arithmetic subtraction (SUB)

SUB A,#n	CP	LD_ I	LD_ A	LD_ ACC	LD_ MEM	E_ A	E_ MEM	E_ ACC	SUB	ADD	Instruction
T0	0	1	0	0	0	0	1	0	0	0	IR ← MEM
T1	1	0	0	0	0	0	0	0	0	0	PC ← PC + 1
T2	0	0	1	0	0	0	1	0	0	0	A ← MEM
T3	1	0	0	0	0	0	0	0	0	0	PC ← PC + 1
T4	0	0	0	0	0	1	0	1	1	0	ACC ← A-ACC

simulation study reveals the possible reactions in terms of the genetic circuits.

Discussion

Synthetic biological circuit design is commonly inspired by the natural biological circuits using transcription factors. There are already biological circuits revealed in the literature using the transcription factor for construction [7, 27–29], some of the circuits use recombinases as the pathways [30, 31]. In this primitive research task, the biological circuits are realized from the viewpoint of mathematics and engineering. There were biological circuits with

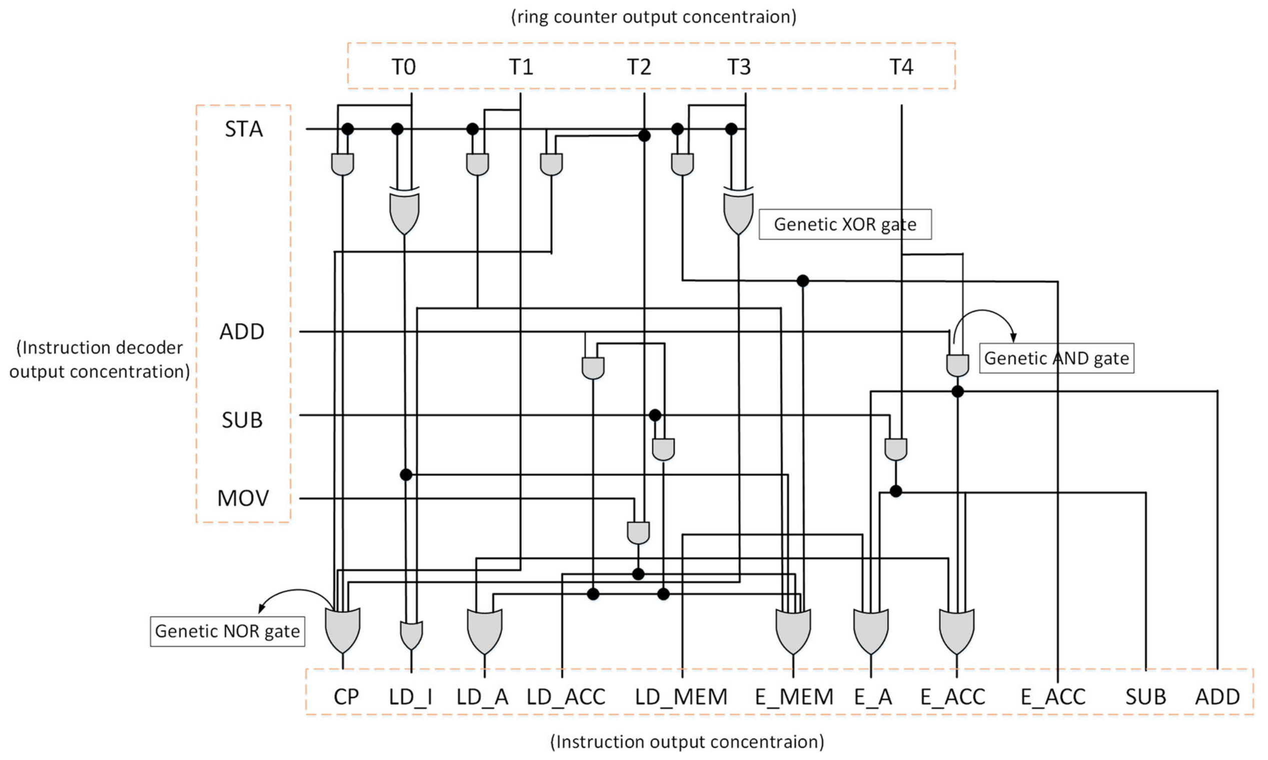


Fig. 19 Hardware of the bio-CU. This figures displays hardware structure of the bio-CU. The logic gates in this figure are constructed by a variety of fundamental genetic logic gates. The working sequence is controlled by the bio-ring counter accompanied with the opcode

Table 2 Operational sequence of the demonstrative instructions

Mnemonic	Effect	T0	T1	T2	T3	T4
MOVA	ACC ← Memory	IR ← Memory	PC ← PC + 1	ACC ← Memory	PC ← PC + 1	
ADD A	ACC ← ACC + A	IR ← Memory	PC ← PC + 1	A ← Memory	PC ← PC + 1	ACC ← (A + ACC)
SUB A	ACC ← A-ACC	IR ← Memory	PC ← PC + 1	A ← Memory	PC ← PC + 1	ACC ← (A-ACC)
OUT A	Memory← ACC	PC ← PC + 1	IR ← Memory	PC ← PC + 1	Memory← ACC	
HLT	STOP					

specific purposes developed and experimentally realized [1, 11, 12, 32–36]. This research task moves forward one step by building up a class of biological circuits which possess sophisticated functions including bio-sequential circuits and bio-combinational circuits. Virtually all circuits in practical digital devices are a mixture of combinational and sequential logic which forms as a basis for constituting a functional biocomputer.

However, the biological circuits being totally replaced by biological molecules still have lots of issues to be solved in nature. Fortunately, there are some impressive progresses unveiled recently. For example, setting up measurement devices to show fluorescence concentrations of a series of repressor or activating genes with different promoter-RBS components and TF by fluorescence measurement has been developed [37]. Synthetic biologists have also created software (such as the Web-based Cello [38]) that automates the design of DNA circuits for living cells. The research team led by Prof. Voigt has developed user interfaces for Cello that would allow biologists to develop a single program and be returned different

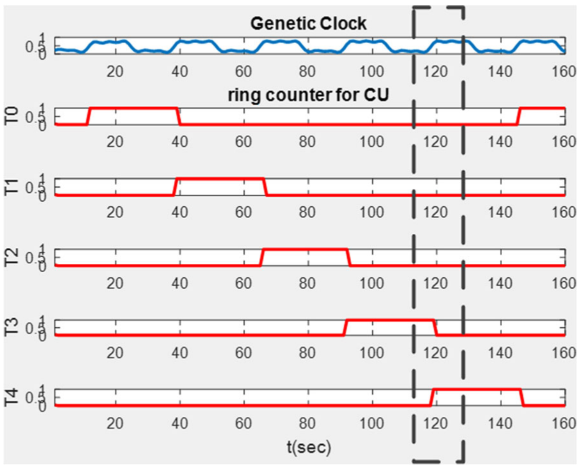


Fig. 20 Q4 is triggered at $t = 120$. The figure demonstrates subtraction executed by the bio-CPU where the bio-ALU is triggered when T4 goes from low to high at $t = 120$

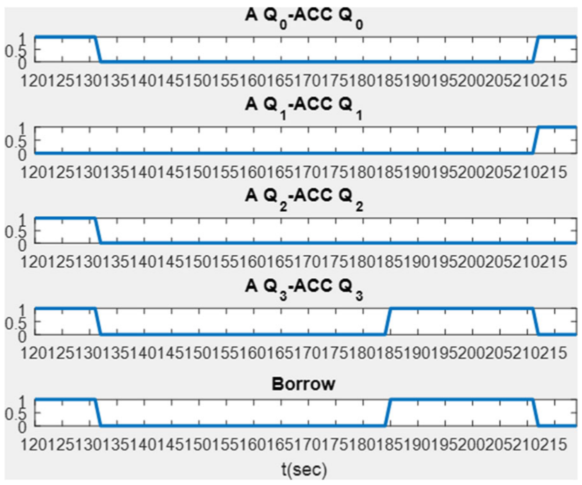


Fig. 21 Result of subtraction. The bio-ALU executes a 4-bit subtraction

DNA sequences for targeted organisms. The library encompasses rich information of system parameters related to the mathematical model describing for the behaviors of a class of genetic logic gates. Therefore, one would be easier to choose suitable promoter RBS components to realize various biological circuits and construct them based on the proposed topology.

Conclusions

Nowadays, science and technology evolve extremely fast than the past decades, the major reason should attribute the success to the invention of the computer. It can make many tedious computational works be processed by itself efficiently and manipulate a huge amount of data at one time. This paper presents an innovative idea that proposes a novel bio-CU structure based on the previously developed genetic circuits. The structure is basically referred to the fundamental silicon CPU. It is believed that if the bio-computer becomes mature in the near future, there will be increasing applications that may revolutionize breakthrough in other fields. As for applications, the DNA computer is reaching to a fair level that could be used for detection of genetic disease with gene therapy. For example, the DNA computer can measure the concentration of a specific antibody and it is possible to determine whether a patients is suffering from a particular disease. A DNA computer can determine whether a specific drug is needed based on the presence of one or more antibodies [39, 40]. Taking a biocomputer into the body from a pill is like to implant a subminiature doctor into patient’s body. Because of the function of bio-memory, it could store the information of diseases [41]. Through the transcription factors to regulate genetic sequences and

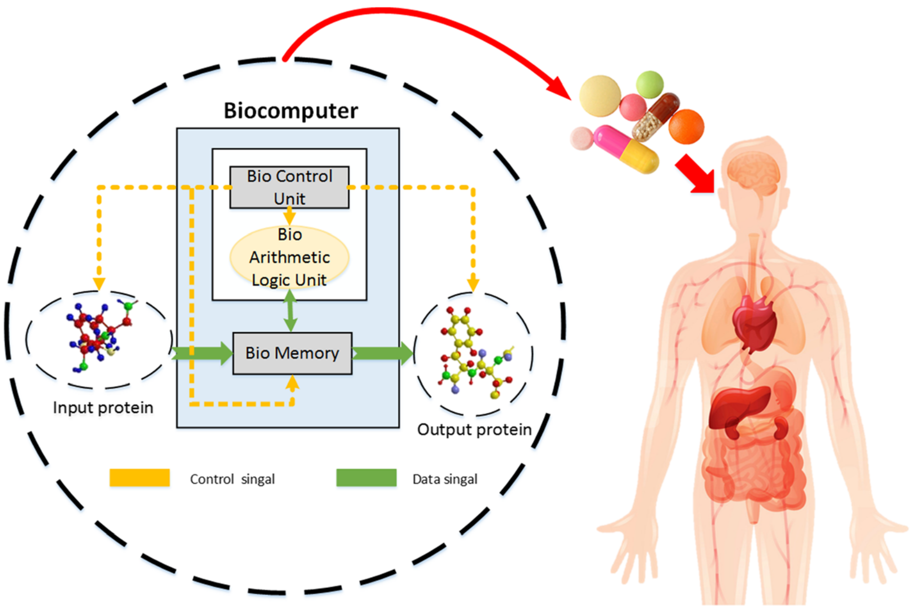


Fig. 22 The schematic diagram of the ideal biocomputer. It represents the potential application to be realized in the biocomputer for curing genetic disease with gene therapy

produce the specific protein is equivalent to the process that one inputs the instructions to dictate the computer to execute a specific function. Figure 22 demonstrates the schematic diagram of the use of a biocomputer for the diagnosis purpose.

The idea presented here is primitive which is not rigorous enough. Actually, the idea was originated from the silicon computer. However, there were significant modifications which have been conducted in our research such as the interface handling between two modules and signal shaping of the output of gene circuits as their output responses are not fast enough. This causes the signal's status change ambiguously leading to unavoidable delay during signal transition in the sequential circuit. Of course, there are many issues to be addressed before a full functioned biocomputer is ultimately realized.

Abbreviations
ACC: Accumulator; ADD: Addition; ALU: Arithmetic Logic Unit; Biocomputer: Biological computer; Bio-memory: Biological memory; Bio-register: Biological register; Bio-ring counter: Biological ring counter; Bio-temporary register: Biological temporary register; CPU: Central Processing Unit; CU: Control Unit; D-FF: D Flip-flop; E_A: Enable Register A; E_ACC: Enable Accumulator; E_MEM: Enable Memory; HLT: Halt; IR: Instruction Register; LD_A: Load Register A; LD_ACC: Load Accumulator; LD_I: Load Instruction Register; MOV: Move; mRNA: messenger RNA; PC: Program Counter; PIPO: Parallel Input and Parallel Output; RNAp: RNA polymerase; SUB: Subtraction; TF: Transcription Factor

Acknowledgements
This research was sponsored in part by National Science Council, Taiwan under the Grants NSC-102-2218-E-005-012 and NSC-101-2221-E-005-015-MY3.

Availability of data and materials
The datasets during and/or analysed during the current study available from the corresponding author on reasonable request.

Authors' contributions
C-LL is responsible for developing the overall system configuration and preparing the manuscript. T-YK is responsible for developing biologica control unit and conducting simulation. W-XL is responsible for conducting the overall system simulation study. All authors read and approved the final manuscript.

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Received: 28 April 2018 Accepted: 6 August 2018
Published online: 14 August 2018

References
1. Wang B, et al. Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology. Nat Commun. 2011;2:508.
2. Miyamoto T, et al. Synthesizing biomolecule-based Boolean logic gates. ACS Synth Biol. 2012;2(2):72–82.
3. Xie Z, et al. Multi-input RNAi-based logic circuit for identification of specific cancer cells. Science. 2011;333(6047):1307–11.
4. Konur S, Gheorghe M. Design and analysis of genetically constructed logic gates. In: The University of Sheffield Engineering Symposium Conference Proceedings Vol. 1. Sheffield; 2014.
5. Chen B-S, Chen P-W. GA-based design algorithms for the robust synthetic genetic oscillators with prescribed amplitude, period and phase. Gene Regul Syst Bio. 2010;4:S4818.
6. Lin CL, Chang YC. Design of synthetic genetic logic circuits based on RSGA. Evol Bioinforma. 2013;9:137–50.
7. Elowitz MB, Leibler S. A synthetic oscillatory network of transcriptional regulators. Nature. 2000;403(6767):335.

8. Wong A, et al. Layering genetic circuits to build a single cell, bacterial half adder. BMC Biol. 2015;13(1):40.
9. Marchisio MA, Stelling J. Automatic design of digital synthetic gene circuits. PLoS Comput Biol. 2011;7(2):e1001083.
10. Chuang C-H, et al. Design of synthetic biological logic circuits based on evolutionary algorithm. IET Syst Biol. 2013;7(4):89–105.
11. Lauria M, et al. Building blocks of a biochemical CPU based on DNA transcription logic. In: 3rd Workshop on Non-Silicon Computation (NSC-3), Munich; 2004.
12. Buchler NE, Gerland U, Hwa T. On schemes of combinatorial transcription logic. Proc Natl Acad Sci. 2003;100(9):5136–41.
13. Cox RS, Surette MG, Elowitz MB. Programming gene expression with combinatorial promoters. Mol Syst Biol. 2007;3(1):145.
14. Win MN, Smolke CD. Higher-order cellular information processing with synthetic RNA devices. Science. 2008;322(5900):456–60.
15. Terzer M, Jovanovic M, Choutko A, et al. Design of a biological half adder. IET Synthetic Biol. 2007;1(1.2):53–8.
16. Lin C-L, Kuo T-Y, Chen Y-Y. Implementation of a genetic logic circuit: bio-register. Syst Synth Biol. 2015;9(1):43–8.
17. LIN C-L, et al. Synthesising gene clock with toggle switch and oscillator. IET Syst Biol. 2014;9(3):88–94.
18. Chen P-K, Lin C-L. Synthesis of genetic clock with combinational biologic circuits. IEEE/ACM Trans Comput Biol Bioinform. 2015;12(5):1206–12.
19. Chang Y-C, Lin C-L, Jennawasin T. Design of synthetic genetic oscillators using evolutionary optimization. Evol Bioinform. 2013;9:137–50.
20. Kuo T-Y, et al. Toward theoretical synthesis of biocomputer. IET Syst Biol. 2017;11(1):36–43.
21. Lin C-L, Chen P-K. Synthesising periodic triggering signals with genetic oscillators. IET Syst Biol. 2014;8(1):1–12.
22. Chuang C-H, et al. Synthesizing genetic sequential logic circuit with clock pulse generator. BMC Syst Biol. 2014;8:63–77.
23. Lamaniya A, Patel B. Design of full adder and full subtractor using DNA computing. Int J Latest Trends Eng Technol. 2014;3:12–6.
24. Tu, D, et al. Engineering gene circuits: foundations and applications. 2007.
25. Chuang C-H, Lin C-L. A novel synthesizing genetic logic circuit: frequency multiplier. IEEE/ACM Trans Comput Biol Bioinform. 2014;11(4):702–13.
26. Church GM, Gao Y, Kosuri S. Next-generation digital information storage in DNA. Science. 2012;337(6102):1628.
27. Gardner TS, Cantor CR, Collins JJ. Construction of a genetic toggle switch in *Escherichia coli*. Nature. 2000;403(6767):339.
28. Stricker J, et al. A fast, robust and tunable synthetic gene oscillator. Nature. 2008;456(7221):516.
29. Nielsen AAK, et al. Genetic circuit design automation. Science. 2016; 352(6281):aac7341.
30. Johnson RC. Bacterial site-specific DNA inversion systems. In: Mobile DNA II: American Society of Microbiology; 2002. p. 230–71.
31. Blomfield IC. The regulation of pap and type 1 fimbriation in Escherichia coli. Adv Microb Physiol. 2001;45:3–51.
32. SIlva-Rocha R, De Lorenzo V. Mining logic gates in prokaryotic transcriptional regulation networks. FEBS Lett. 2008;582(8):1237–44.
33. Buchler NE, Gerland U, Hwa T. Nonlinear protein degradation and the function of genetic circuits. Proc Natl Acad Sci. 2005;102(27):9559–64.
34. Anderson JC, Voigt CA, Arkin AP. Environmental signal integration by a modular AND gate. Mol Syst Biol. 2007;3(1):133.
35. Tamsir A, Tabor JJ, Voigt CA. Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. Nature. 2011; 469(7329):212.
36. Weiss R, Basu S. The device physics of cellular logic gates. In: NSC-1: The First Workshop of Non-Silicon Computing. Boston, Massachusetts; 2002.
37. Su WW, et al. Observer-based online compensation of inner filter effect in monitoring fluorescence of GFP-expressing plant cell cultures. Biotechnol Bioeng. 2005;91(2):213–26.
38. Web-based genetic circuit design automation. http://www.cellocad.org/.
39. Engelen W, et al. Antibody-controlled actuation of DNA-based molecular circuits. Nat Commun. 2017;8:14473.
40. Chao J, et al. Structural DNA nanotechnology for intelligent drug delivery. Small. 2014;10(22):4626–35.
41. Goldman N, et al. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. Nature. 2013;494(7435):77.