Design of Arithmetic and Control Cells for a DNA Binary Processor

Aby K George
Electrical and Computer Engineering
Wayne State University
Detroit, MI, USA
Email: aby.george@wayne.edu

Amjad Almatrood
Electrical and Computer Engineering
Wayne State University
Detroit, MI, USA
Email: amjad.almatrood@wayne.edu

Harpreet Singh
Electrical and Computer Engineering
Wayne State University
Detroit, MI, USA
Email: hsingh@eng.wayne.edu

***Abstract*—Recently a great deal of interest has been shown by researchers on developing a bio-molecule based computer. The basic building blocks of such a computer are arithmetic units and memory. These units can be designed using Boolean logic gates as in the case of electronic circuits. Instead of using silicon based technology, Boolean logic gates can be generated from biological systems. One such system can be generated by a DNA reaction mechanism based on a reversible strand displacement process. A generalized pipeline architecture employing DNA reaction chain mechanism for the arithmetic operations such as addition, subtraction, multiplication, and division is discussed in this paper. A single control line is used in the pipeline array to control the different modes of operations. The primary functional blocks in a pipelined array are arithmetic unit and control unit. These units are made up of basic Boolean logic gates. To implement these gates, a DNA strand displacement process is employed. A set of integrating and amplifying gates are used in cascade with different threshold values to build different digital logic operations. The main advantage of such a system is that the arithmetic operations can be overlapped in the pipeline and thus a high speed operation is possible. Such a general Boolean model will be a step towards the development of a bio-computer. The ultimate goal of such a method is to design an automated system using logic gates, which make decisions within living cells.**

***Index Terms*—DNA binary processor; Arithmetic cell; Control cell; DNA strand displacement; Visual DSD.**

I. Introduction

The future of medical technology is on building bio-molecule based computers and nano-structures inside the living organism so that deceases like cancer could be treated internally. Such a system is possible only if there exists a powerful internal processor similar to that are used externally. In a digital computing device or processor the input information is first converted into its equivalent binary form and then processed digitally. The basic building block for such a computer system is digital logic gates such as NOT, OR, and AND. If a living cell can perform the basic digital logic gate functions, and they could be networked together to design complex circuits, then any system that built externally could be built internally also. The first step in achieving such a development is the implementation of bio-molecular arithmetic unit which can perform the basic arithmetic operations. Long term goal of such a system will be the ability to design some decision-making cell networks within the living cells.

To make any bio-molecular based computers the basic building block is bio-molecular digital circuits [1]. Recently a lot of interest has been shown towards building digital logic gates and finite state machines for bio-molecule based systems [2]. There are different kinds of bio-molecule based logic gates available in the literature [3]–[7]. Some application specific bio-molecule based circuits such as counter, and timer are discussed in [8] and [9]. The existing bio-molecular logic gates are broadly classified into a cell based and cell free systems. To design a large, complex logic circuit, the basic building block should be suitable for scaling up. Even though there are many methods available, the scaling up of circuits is a major issue. Scaling up of bio-molecular circuits depends on the interconnection between different gates. In most of the cases, the inputs and outputs are not uniform as in the case of electronic circuits. The introduction of toehold-mediated DNA strand displacement method solved this issue. The theory and practice of DNA strand displacement circuits are explained by Lulu Qian and Erik Winfree [10]. A seesaw gate based scaled DNA circuit architecture based on the reversible strand displacement is also available in the literature [11]. A dual rail AND-OR logic [11] is used in this paper as a universal Boolean gate for designing the arithmetic and control cell.

In this paper, a DNA arithmetic and control cell is proposed. These cells are used in a pipelined architecture to implement various binary arithmetic operations such as addition, multiplication, subtraction and division. A generalized pipeline array using basic logic gates are proposed by A K Kamal et al. in [12] and an optical design and implementation of such an architecture is explained in [13]. The binary arithmetic using DNA is designed using two basic building blocks called arithmetic cell and control cell. These cells have different Boolean functions associated with them and they are arranged in such a manner to give different mathematical operations for different selection of inputs. In this proposed method Boolean functions used in arithmetic and control cells are implemented using DNA strand displacement method.

The organization of the paper is as follows. Section I briefly explain the theory of DNA strand displacement method and the Seesaw logic gates. In Section II, the arithmetic and control cell and their working in pipelined processor for different arithmetic operations are discussed in detail. The

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.125

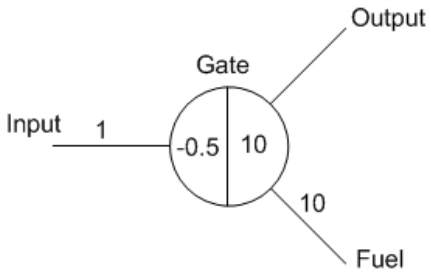

Fig. 1. Abstract Gate Diagram of a DNA motif or seesaw gate

simulation results of operations of arithmetic and control cells for different input combinations are explained in Section IV. Finally, some concluding remarks are given in section V.

II. DNA Strand Displacement

Even though different bio-molecule based Boolean circuits are available, scaling up of the circuit is difficult with most of the existing methods. These methods use non-uniform inputs and outputs. To scale up the circuit a new method called DNA strand displacement method is introduced [5], [14]. In this approach computations is performed based on the hybridization between complementary nucleotide sequences. This uses a simple seesaw gate, which is also called as DNA gate motif. Different DNA structures such as hairpins [15], [16], simple linear complexes [14] etc. can be used to perform a strand displacement operation. There are mainly two processes which drive a strand displacement system. First is the increase in entropy by releasing the strands and the second is enthalpy from creating additional base pairs [14].

A. DNA Gate Motif

A DNA gate motif or seesaw gate consists of inputs, fuel and output. Each gate is configured with a threshold level. When the input concentration exceeds this threshold level, the input catalytically converts the fuel to an output. A typical DNA motif is shown in Fig. 1. The signals (input, fuel and output) used in these gates are single stranded DNA molecules referred as signal strands while the gate is partially double stranded DNA molecule referred as threshold complex or gate complex. The numerical values given in the figure are initial concentrations. The numbers above wires give concentration of the signal strands, while those inside gate give gate:signal complex or threshold complex (negative) concentrations.

The domain level representation of gate complexes and signal molecules are shown in Fig.2. The signal strands consists of three parts: a left recognition domain, a central toe-hold, and a right recognition domain. The signal strands on either sides of toehold domain identify the two gates it is connecting. In the figure, input is S_1TS_2 , and the fuel is S_2TS_4. S_2 recognition domain is the active domain participating in the catalytic reaction. The recognition domains such as S_1, S_2, S_3 and S_4 are relatively larger than the toehold.

There may be different recognition domains used in a system, but the toehold used in all the molecules are the

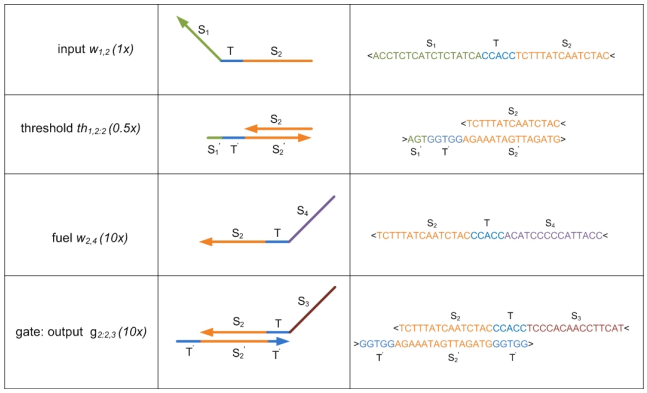

Fig. 2. Example sequences of a DNA motif for 'seesaw' gate

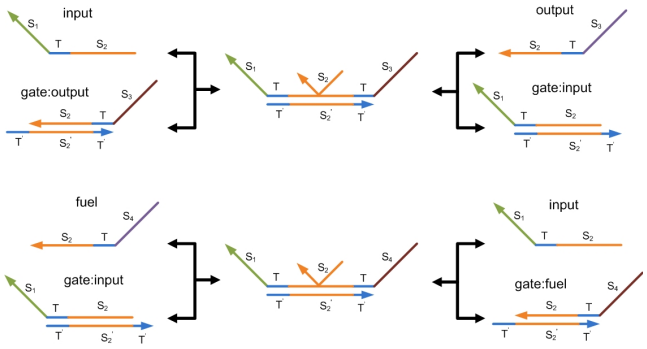

Fig. 3. The DNA gate motif reaction mechanism

same. The inputs, outputs and fuel are similar in mechanism and structure as seen from the example in Fig. 3. Similarly gate:signal complexes are also having uniform structure.

An abstract representation of a gate is shown in Fig. 1 as a two-sided node, with some wires connected on both sides. A signal strand can be either toehold exposed or bounded to a gate base strand with its toehold isolated and thus idle as shown in Fig. 3. In the given example, gate base strand $T^*S_2^*T^*$ bound to S_2TS_3 at the right to form gate:output. There are three principal operations associated with a gate.

1) Stoichiometric triggering: The signal strand such as input bound together with the exposed toehold of some gate:signal complex like gate:output to produce branch migration and creates a signal strand such as output and another gate complex like gate:input. In this gate complex the toehold will be exposed on the opposite side and thus the overall reaction is reversible [17].
2) Catalytic cycle: In a catalytic cycle, the gate complex (gate:input) produced by the stoichiometric triggering, binds to the fuel to produce more input and thus trigger more output [17]. Thus a small amount of input is enough to release a large amount of output.
3) Thresholding: This process occurs due to the extended toehold of the threshold complex. The reaction rate exponentially depends on the toehold length [17]. The

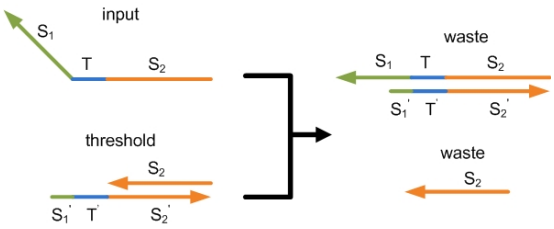

Fig. 4. Threshold motif reaction mechanism

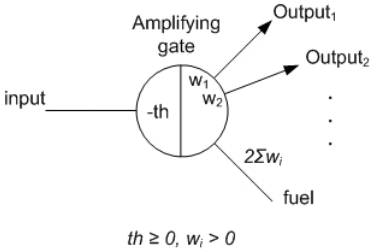

Fig. 5. Amplifying gate

reaction rate of input strands with threshold complexes are much higher than that of the gate complexes. The reaction between input strand and threshold complex produces waste complexes having no exposed toehold domains. Since there are no exposed toehold domains, they are inert. The stoichiometric triggering will occur only if the input concentration is greater than the threshold concentration.

For the example shown in Fig. 1, the concentrations of gate:ouput complex, fuel signal strand, input signal strand and threshold complex are 10x, 10x, 1x, and 0.5x respectively. Since the input concentration is greater than that of the threshold complex, the input strand reacts with the gate:output complex to produce the output strand. In such a situation the output will keep being released up to a maximum concentration. The final concentration of the output strand will be approximately 5x. Thus a seesaw gate amplifies the input signal when it exceeds the threshold complex concentration.

B. Boolean Gates using DNA motif

The seesaw gates in cascade can be used to design AND or OR logic gates. The cascaded combination of an integrating and amplifying gate are used to make the boolean gates. The amplifying gate can drive multiple outputs. This is similar to the gate:output complex with same gate base strand bound by signal strand with different right side recognition domains. The amount of free fuel should be greater than twice that of the sum of all bounded outputs [11]. The expressions for the outputs of the gates shown in the Fig. 5. are given by:

$$output_i = \begin{cases} w_i & \text{if input} > Th \\ 0 & \text{if input} \leq Th \end{cases} \quad (1)$$

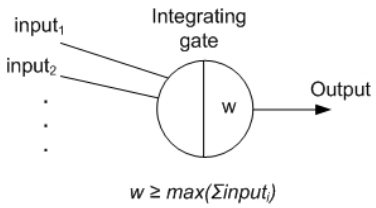

Fig. 6. Integrating gate

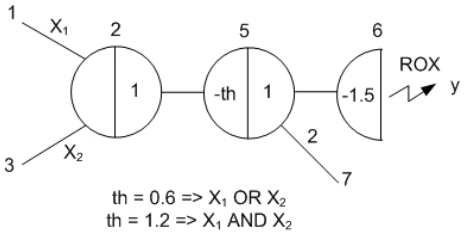

Fig. 7. AND or OR gate using integrating and amplifying gates

The integrating gate has no fuel or threshold associated with it. This gate corresponds to stoichiometric triggering. It can support multiple inputs by adding together inputs with same right side recognition domain and different left side recognition domain to connect to different gates. This is shown in Fig. 6. The output of integrating gate is given by:

$$output_i = input_1 + input_2 + ... \quad (2)$$

An OR or AND gate can be constructed by cascading the integrating gate and amplifying gate as shown in Fig. 7. When the threshold is 0.6, the output will be logical 1 when the sum of inputs for that particular gate (gate no. 5 in Fig. 7) is greater than 0.6 and logical zero otherwise. The logical zero is in the range of 0 to 0.2 and logical one is in the range of 0.8 to 1. An OR gate can be converted to an AND gate by changing the threshold from 0.6 to 1.2. Since multiple inputs are supported by integrating gate and multiple outputs are supported by the amplifying gate, they can efficiently support fan in and fan out.

C. Dual rail AND−OR logic

AND and OR gates are easy to implement using DNA strand logic as explained in the previous section, while NOT operation is difficult to implement. Hence AND logic or OR logic alone cannot be used to construct any logic circuit. To solve this issue each input is replaced by a pair of inputs and each gate is replaced by a pair of AND or OR gate [11]. Such a logic is called dual rail AND - OR logic. This dual rail AND - OR logic gate is considered as a universal gate and any complex logic circuitry can be implemented using this gate pairs alone.

III. Binary DNA Processor

A binary DNA processor is controlled arithmetic cells with adder subtractor operations. It has two basic cells associated

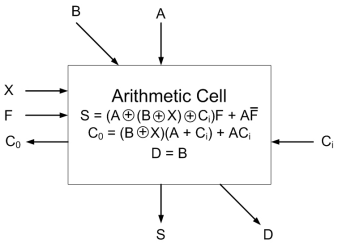

Fig. 8. Arithmetic cell

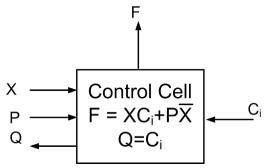

Fig. 9. Control cell

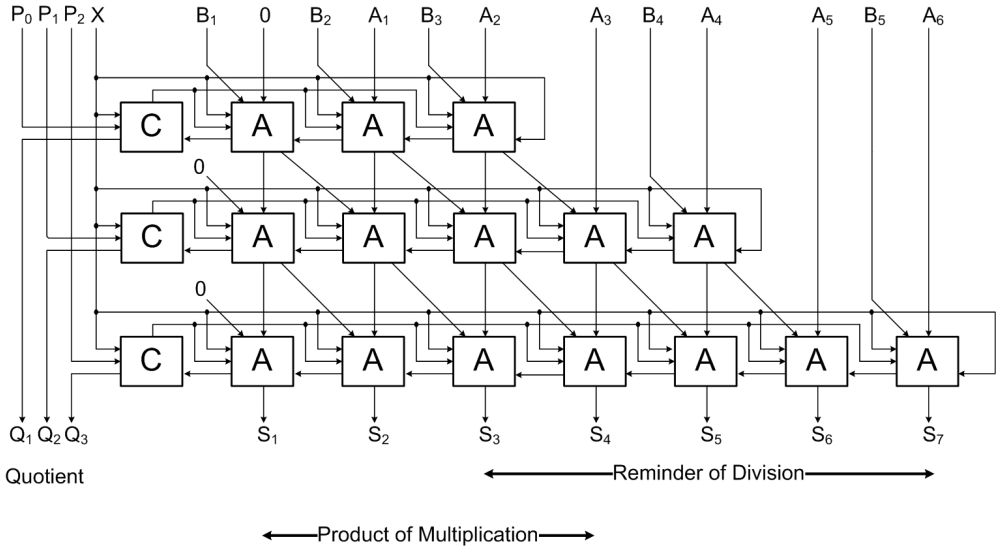

Fig. 10. Block diagram of pipeline array

with it: control cell and arithmetic cells. A purely binary arithmetic is used to facilitate all the desired operations. The implementation of the binary functions are carried out using AND OR dual rail logic.

The arithmetic cell comprises of three inputs ($A, B,$ and C_i), two control lines (X and F) and three outputs (S, C_0 and D) as shown in Fig.8. The input, output relations for arithmetic cell are given by:

$$S = [A \oplus (B \oplus X) \oplus C_i]F + A\overline{F} \tag{3}$$

$$C_0 = (B \oplus X)(A + C_i) + AC_i \tag{4}$$

$$D = B \tag{5}$$

The control cell takes three single bit inputs ($X, P_i,$ and C_i) and generate single output F, which is an input to the arithmetic cell. The block diagram of control cell is shown in Fig. 9 and the expression for F is given by:

$$F = XC_i + P\overline{X} \tag{6}$$

The overall diagram of a binary DNA processor is shown in Fig. 10. The C block corresponds to a control cell and A block is the arithmetic cell. Different operations of the DNA processor are explained in Table I.

A carry look ahead method is used for the addition operation [18]. The control cell input, X is assigned with a logical zero and $P_0P_1P_2$ is set to 010. The two binary numbers to be added are given to A and B. The carry in each operation is C_i and it passes from one arithmetic block to another as shown. The carry input for the first arithmetic block (right most block) is X itself. Here the most significant bit (MSB) is given with a subscript 1 (A_1, B_1). The result of addition is available from S_1 to S_5, where S_1 is the MSB.

For subtraction, a subtract borrow look ahead algorithm is employed [18]. For this operation, the X input is set to 1 such that 2's complement of B is added to A. The input X is referred as subtraction enable line. The $\overline{B}$ in each arithmetic block is achieved by $B \oplus 1$ operation and the 2's complement is achieved by adding X as C_i in the right most arithmetic cell.

The multiplication operation is selected by setting $X = 0$ and $A = 0$. The DNA processor is designed to perform the multiplication of a 3 bit number with a 2 - bit number. A right shift and add method is employed here [18]. The 3 bit input for multiplication is $B_1B_2B_3$, and 2-bit input is P_1P_2 with B_1 and P_1 as lease significant bits (LSB). P_0 is also set to 0 so that the multiplication will start from the second stage of the processor. The output is obtained from bits $S_1 - S_5$. Here S_1 will act as the LSB.

Another operation the processor designed to carry out is the division of a 4-bit number $(A_1 - A_4)$ by a 3-bit number $(B_1 - B_3)$. Here A_1 and B_1 are the MSB. A right shift and subtract method is used for this operation [18]. To set the arithmetic cells in the subtraction mode (2's complement addition), a logical high signal is given to X. Since the value of X is 1, the operation is independent of P value. B_1 in second and third stages are set to one for the left most arithmetic cells. The result is obtained as quotient (Q_1Q_2) and reminder $(S_3S_4S_5)$. Q_2 is the MSB for quotient and S_3 is the MSB for reminder.

IV. Results and Discussion

To simulate the design using the DNA strand displacement method, arithmetic cells and control cells are first written in AND-OR-NOT circuit level. This AND-OR-NOT circuit code is compiled and generated corresponding dual rail AND-OR circuit using Seesaw Compiler. The AND - OR dual rail design is then used to create the seesaw circuit. The seesaw

TABLE I
DNA BINARY PROCESSOR PIN SETTINGS FOR DIFFERENT OPERATIONS

Operation	**X**	$P_0P_1P_2$	**A**	**B**	**Output**
Addition	0	010	Input 1	Input 2	$S_1 - S_5$
Subtraction	1	X	Input 1	Input 2	$S_1 - S_5$
Multiplication	0	$P_0 = 0$, Input(P_2P_1)	0	Input $(B_3B_2B_1)$	$S_5 - S_1$
Division	1	x	Dividend$(A_1 - A_4)$	Divisor$(B_1 - B_3)$	Quotient(Q_2Q_1) Reminder$(S_3S_4S_5)$

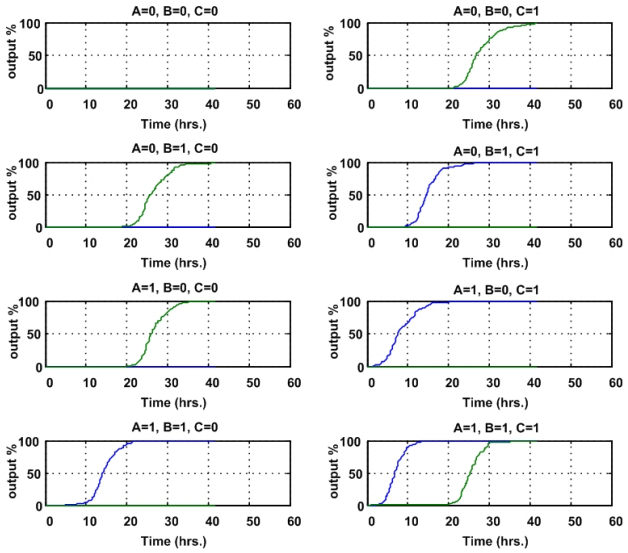

Fig. 11. Simulation of Arithmetic Cell as Adder

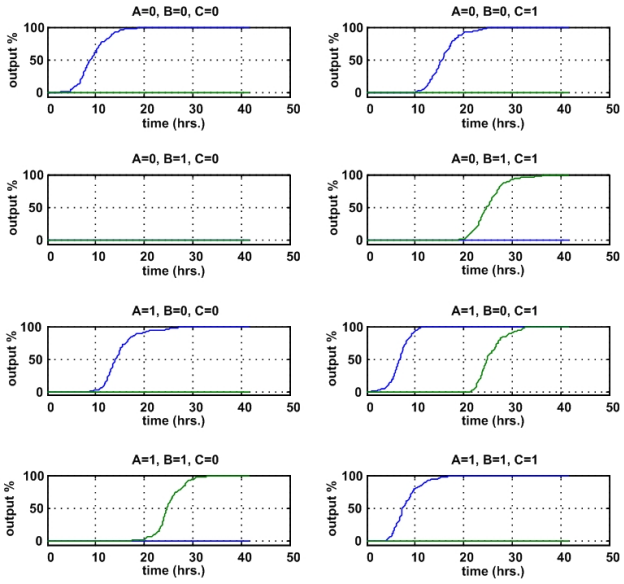

Fig. 12. Simulation of Arithmetic Cell as Subtractor

circuit thus created is simulated in Visual DSD (DNA Strand Displacement) software [19]. The visual DSD tool is a web based graphical interface to analyse the circuits designed using DNA strand displacement.

The arithmetic cell can be set up either as an adder or a subtractor. The mode of operation is decided by checking input X. When $X = 0$ and $F = 1$, the arithmetic cell will work as a full adder and the output S is given by:

$$S = A \oplus B \oplus C_i \tag{7}$$

The F value is decided by the control cell. The F value is given by eq. (6). If $F = 1$, the arithmetic cell work as adder otherwise it will simply pass the input A. This functionality is employed in multiplication operation. The simulation results for arithmetic cell configured as adder is shown in Fig. 11. The green lines are given for Sum (S) and the blue lines represents Carry (C_0).

When $X = 1$, the arithmetic cell expression is given by:

$$S = A \oplus \overline{B} \oplus C_i \tag{8}$$

$$C_0 = \overline{B}(A + C_i) + AC_i \tag{9}$$

This yields a 2's complement based subtraction operation. The simulation results obtained from the arithmetic cell for subtraction is shown Fig.12. In this case also when $F = 0$,

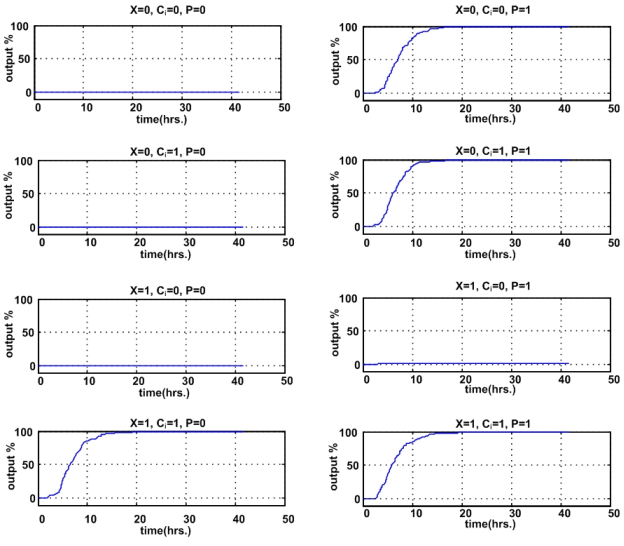

Fig. 13. Simulation of Control Cell

the S output passes the input A. The green lines are given for Sum (S) and the blue lines represents Carry (C_0).

The control cell is used for signal selection. The logical

operation of the control cell is given in eq. 6. When $X = 0$, $F = P$ and when $X = 1$, $F = C_i$. The simulation results of control cell are shown in Fig. 13.

V. Conclusion

In this paper a DNA strand displacement technique is used for the design of arithmetic and control cells. These cells can be considered as basic building block for a bio-molecule based computer. The circuit is first designed using basic digital logic gates. This digital logic circuit is converted to a dual rail AND-OR circuit and then to a seesaw circuit. The seesaw circuit is then implemented in Visual DSD and simulated with different input combinations. A binary processor made up of these arithmetic and control cells are also discussed in this paper. This binary processor is capable of performing basic arithmetic operations such as addition, multiplication, subtraction and division. However the key issue of such a design using DNA strand displacement method is that the gates used here are used only once type. This prevents the implementation of sequential circuits using filip-flops, clocks etc.

References

[1] Y. Benenson, T. Paz-Elizur, R. Adar, E. Keinan, Z. Livneh, and E. Shapiro, "Programmable and autonomous computing machine made of biomolecules," *Nature*, vol. 414, no. 6862, pp. 430–434, 2001.

[2] K. Oishi and E. Klavins, "Framework for engineering finite state machines in gene regulatory networks," *ACS synthetic biology*, vol. 3, no. 9, pp. 652–665, 2014.

[3] T. Miyamoto, S. Razavi, R. DeRose, and T. Inoue, "Synthesizing biomolecule-based boolean logic gates," *ACS synthetic biology*, vol. 2, no. 2, pp. 72–82, 2012.

[4] B. Wang, R. I. Kitney, N. Joly, and M. Buck, "Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology," *Nature communications*, vol. 2, p. 508, 2011.

[5] G. Seelig, D. Soloveichik, D. Y. Zhang, and E. Winfree, "Enzyme-free nucleic acid logic circuits," *science*, vol. 314, no. 5805, pp. 1585–1588, 2006.

[6] E. Agliari, M. Altavilla, A. Barra, L. D. Schiavo, and E. Katz, "Notes on stochastic (bio)-logic gates: computing with allosteric cooperativity," *Scientific reports*, vol. 5, 2015.

[7] F. Dannenberg, M. Kwiatkowska, C. Thachuk, and A. J. Turberfield, "Dna walker circuits: computational potential, design, and verification," *Natural Computing*, pp. 1–17, 2014.

[8] S. Basu, Y. Gerchman, C. H. Collins, F. H. Arnold, and R. Weiss, "A synthetic multicellular system for programmed pattern formation," *Nature*, vol. 434, no. 7037, pp. 1130–1134, 2005.

[9] T. Ellis, X. Wang, and J. J. Collins, "Diversity-based, model-guided construction of synthetic gene networks with predicted functions," *Nature biotechnology*, vol. 27, no. 5, pp. 465–471, 2009.

[10] L. Qian and E. Winfree, "A simple dna gate motif for synthesizing large-scale circuits," in *DNA computing*. Springer, 2009, pp. 70–89.

[11] ——, "Scaling up digital circuit computation with dna strand displacement cascades," *Science*, vol. 332, no. 6034, pp. 1196–1201, 2011.

[12] A. Kamal, H. Singh, and D. Agrawal, "A generalized pipeline array," *Computers, IEEE Transactions on*, vol. 100, no. 5, pp. 533–536, 1974.

[13] P. P. Banerjee and A. Ghafoor, "Design of a pipelined optical binary processor," *Applied optics*, vol. 27, no. 22, pp. 4766–4770, 1988.

[14] D. Y. Zhang, A. J. Turberfield, B. Yurke, and E. Winfree, "Engineering entropy-driven reactions and networks catalyzed by dna," *Science*, vol. 318, no. 5853, pp. 1121–1125, 2007.

[15] K. Sakamoto, H. Gouzu, K. Komiya, D. Kiga, S. Yokoyama, T. Yokomori, and M. Hagiya, "Molecular computation by dna hairpin formation," *Science*, vol. 288, no. 5469, pp. 1223–1226, 2000.

[16] P. Yin, H. M. Choi, C. R. Calvert, and N. A. Pierce, "Programming biomolecular self-assembly pathways," *Nature*, vol. 451, no. 7176, pp. 318–322, 2008.

[17] D. Y. Zhang and E. Winfree, "Control of dna strand displacement kinetics using toehold exchange," *Journal of the American Chemical Society*, vol. 131, no. 47, pp. 17 303–17 314, 2009.

[18] M. M. Mano, "Computer system architecture, 1993," *Prentice Hall*, vol. 3, p. 299.

[19] M. R. Lakin, S. Youssef, F. Polo, S. Emmott, and A. Phillips, "Visual dsd: a design and analysis tool for dna strand displacement systems," *Bioinformatics*, vol. 27, no. 22, pp. 3211–3213, 2011.