Scheduling and Optimization of Genetic Logic Circuits on Flow-Based Microfluidic Biochips

Yu-Jhih Chen[†], Sumit Sharma[‡], Sudip Roy[‡] and Tsung-Yi Ho[†]
[†]Dept. of Computer Science, National Tsing Hua University, Taiwan
[‡]CoDA Laboratory, Dept. of Computer Science & Engineering, Indian Institute of Technology Roorkee, India
Email: {m5566989, sumitsharma1825}@gmail.com, sudiproy.fcs@iitr.ac.in, tyho@cs.nthu.edu.tw

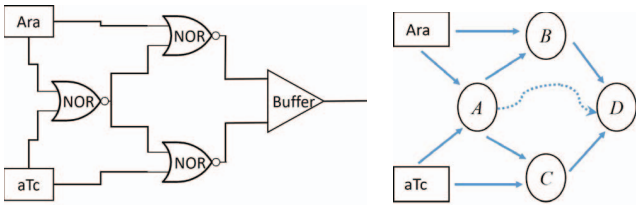
***Abstract*—Synthetic biologists design genetic logic circuit using living cells. A challenge in this task is the difficulty in constructing bigger logic circuits with several living cells due to the crosstalk effect among the biological cells. In order to remove the crosstalk effect, current practice is to use separate chambers on a flow-based microfluidic biochip to isolate each reaction zone. The state-of-the-art technique assumes different reaction times for each gates in a genetic logic circuit. This assumption is pessimistic as each gate has different reaction rate from others. Hence, it will cause unnecessary waiting time for faster gates and this may in turn increase the total experiment completion time significantly. In this paper, we propose a genetic logic circuit synthesis technique for flow-based microfluidic biochip considering different reaction time of each logic gate. Simulation results show that the proposed scheme reduces the total experiment completion time. We further minimize the number of control valves and optimize the routing of flow and control layers in the chip layout, which in turn reduces the design cost.**

I. Introduction

Recently, synthetic biology has emerged as a promising discipline, where engineering principles are used to design and assemble biological components [1]. Genetic logic circuits are designed using living cells to perform logical operations in a similar manner as in electronics circuits. In order to design a programming language by Deoxyribonucleic acid (DNA), synthetic biologists generate proteins using DNA fragments reacting with enzymes, where each fragment represents individual instruction [2]. A sequence of these fragments can perform certain computations, as a computer program does. This program can indicate cells to implement a series of tasks, which are performed by logic gates.

Synthetic biology attempts to create novel biological systems using engineering principles. In literature, many research groups have reported different synthetic biology systems [3–9]. Brophy and Voigt used different enzymes with different substances to implement logic gates [3]. Moon *et al.* combined genetic logic gates to implement a 4-input AND gate [4].

In traditional method, genetic logic circuits are implemented on petri dish, where controlling the signal flow is extremely difficult. Moreover, unexpected signal flow of one genetic logic gate may interrupt the functionalities of other genetic logic gates. This phenomenon is known as crosstalk effect [8]. It increases the wastage of sample and reactant fluids and reduces the accuracy of the biological experiments. For instance, Fig. 1(a) shows the construction of a XOR gate using genetic logic gates [10], where gate signals are spread in all directions.



(a) (b)

Fig. 1. (a) XOR gate realization using 3 NOR gates and one buffer. (b) Interference of signal flows from gate *A* to gate *D*.

Hence, the intermediate gate *A* may wrongly diffuse its signal to gate *D* causing a faulty reaction in gate *D* (Fig. 1(b)).

Moreover, as the traditional way of implementing synthetic biology on microfluidic biochip is manual, it is error-prone and expensive. An important challenge is to develop synthetic biology systems on a large scale. The advanced technology of microfluidics helps the synthetic biologists to design small and highly accurate genetic logic circuits with low latency. A flow-based microfluidic biochip can provide high precision control using microscale devices for the flow of biological substances. Hence, it can construct more reliable and scalable genetic logic systems for synthetic biology experiments. Flow-based microfluidic biochip provides a suitable platform for synthetic biology due to its compartmentalization of reactions in different chambers. Microchannels in flow-layer on a flow-based microfluidic biochip can separate the reaction in every genetic logic gate from others and can restrict the direction of flow. Thus, the effect of crosstalk in such platform for implementation of synthetic biology system reduces significantly. Recently, the biochemical reactions have been carried out on flow-based microfluidic biochip to achieve the functionality of genetic logic gates [8, 9]. However, there is no significant work in this direction except *Fluigi* by Huang *et al.* [8, 9].

In this paper, we propose a genetic logic circuit synthesis technique for flow-based microfluidic biochips considering different reaction times for each gate. This reduces the total experiment completion time. Moreover, we further minimize the number of control lines and optimize the routing of flow and control layers in the chip layout, which in turn reduces the design cost. We refer our proposed technique for scheduling, placement, flow and control layers routing as genetic logic circuit synthesis (called as *GLCS*). Simulation results show that *GLCS* can reduce the average number of control lines by 61% over the initial design with the same experiment completion

978-3-9815370-8-6/17/$31.00 ©2017 IEEE

time. *GLCS* can also reduce the experiment completion time by 13.4% compared with *Fluigi* [9] at the cost of 42% increase in the number of control lines (on average).

The remainder of the paper is organized as follows. Section II describes the basic background and prior work. Motivation and problem formulation are presented in Section III. The proposed technqiue is described in Section IV. Simulation results are presented in Section V and finally, conclusions are drawn in Section VI.

II. Background and Prior Work

Huang *et al.* [8] proposed a small, high accurate flow-based microfluidic biochip for genetic logic gates. They designed a genetic logic gate as a group of cells. Each cell of the genetic gate is placed in different microfluidic chambers and their fluid flow is controlled by the microvalves. These microfluidic chambers reduces the crosstalk effect. Fig. 2 depicts a model of a genetic logic gate having five control valves and three compartmental zones or chambers [8]. Two chambers C_1 and C_2 can store two input fluids (called "inducers" or enzymes) arabinose (Ara) and anhydrotetracycline (aTc), while the biochemical reaction is performed in chamber C_3. Five microvalves V_1, V_2, V_3, V_4 and V_5 are used to control the flow of fluids within the gate. Majority of the experiment time is spent in C_3, as it is the most time-consuming reaction. Two input fluids are send into C_1 and C_2 by opening V_1 and V_2. The purpose of C_1 and C_2 is to ensure that the experiment can obtain sufficient amount of input fluids. After input of fluids, two valves V_3 and V_4 are opened while closing two other valves V_1 and V_2 to make two fluids to pass into C_3 for executing the biochemical reaction.

Huang *et al.* [8] also designed a framework called *Fluigi* that uses the existing EDA tools for microfluidics. They proposed a design-flow for chip-level behavioral simulation by implementing following steps: logic synthesis and physical design of the chip. As the complexity of the experiment increases, many genetic logic gates are required to be placed on the biochip. Hence, it is not possible to control every valve using the independent control line, as it will increase the manufacturing cost significantly. In order to reduce the number of valves, they proposed the control line sharing with pipeline stage method. They put the genetic logic gates from output to input into the breadth-first search tree. The gates placed in the same depth are classified into the same pipeline stage and the corresponding control lines are shared. However, the gates need to be synchronized to either open or close. Therefore, all gates are assumed to have the same reaction time in every pipeline stage. This method may increase the total experiment completion time.

III. Motivation and Problem Formulation

A. Motivation

Huang *et al.* developed a computer-aided-design (CAD) framework called *Fluigi* to synthesis a genetic logic circuit using microfluidic devices [8, 9] and it generates the placement of that circuit into a flow-based microfluidic biochip. In *Fluigi*, every genetic logic gate is assumed to have an identical or equal reaction time. However, in practical experiment, different gates of a genetic logic circuit may take different times to complete the reactions. Hence, assumption of the identical reaction time for all the genetic logic gates may result in a longer completion time of the experiment (T_e). *Fluigi* can reduce the number of control lines (C) by 65%, 68%, 67% and 83% for four benchmark logic circuits namely AND4 [4], XOR3 [9], HALF ADDER [11] and 8-3ENC [9], respectively, compared to an unoptimized or initial design. However, it increases the experiment completion time (T_e) by 18%, 11%, 11% and 23% for those four benchmark logic circuits, respectively, compared to the initial design.

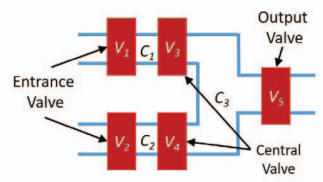


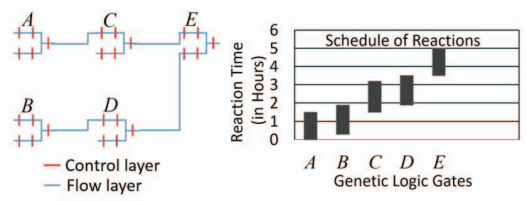
Fig. 2. Model of a genetic logic gate with 3 chambers and 5 microvalves.

Gonick and Wheelis proved that every enzyme has different reaction rates [12]. In this paper, different genetic logic gates are considered to have different reaction times and the circuit is scheduled to optimize the experiment completion time. This motivates us to propose a timing-driven optimization technique, in which each logic gate delivers its result to the successor gate immediately after completion of the reaction. Hence, there is no waiting time for any genetic logic gate and the overall experiment completion time can be reduced significantly.

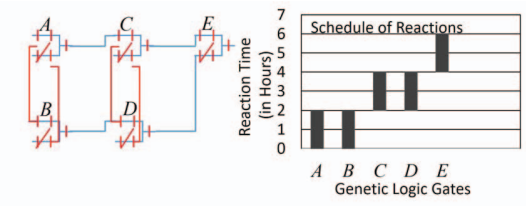
Three different layouts and corresponding schedules of an example genetic logic circuit with five gates are shown in Fig. 3. The traditional (initial) design is presented in Fig. 3(a), in which each gate is independent and hence the experiment completion time is minimum. In this design, each valve is controlled by individual control line and hence more number of control lines are required. Fig. 3(b) depicts the design obtained by *Fluigi* [9] that can reduce the number of control lines (from 25 to 12) with a trade-off of longer experiment completion time (from 5 *hrs* to 6 *hrs*) compared to the initial design. Whereas, the design (Fig. 3(c)) obtained by the proposed approach can have the similar control line usage as in *Fluigi* (Fig. 3(b)) and the same experiment completion time as in initial design (Fig. 3(a)).

B. Problem formulation

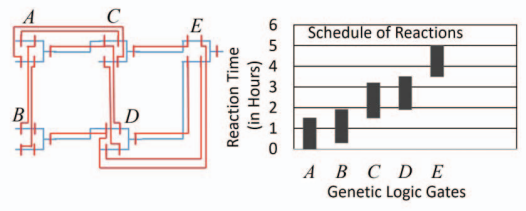
The netlist of a genetic logic circuit with n gates is represented as a directed acyclic graph (DAG) $G(V,E)$, where V is the vertex set $\{\{v_1,t_r(v_1)\},\{v_2,t_r(v_1)\},\ldots,\{v_n,t_r(v_1)\}\}$, $\{v_i,t_r(v_i)\}$-pair represents the vertex corresponding to a gate with its reaction time ($t_r(v_i)$), E is the set of edges in which each edge corresponds to the wire connection between two gates. In order to synthesis the circuit we need to schedule the DAG and perform placement and routing of the flow and control layers on a flow-based microfluidic biochip. The scheduled DAG is represented by a graph $G'(V,E,T)$ obtained



(a) Experiment completion time: ~ 5 *hrs*, #Control lines: 25.



(b) Experiment completion time: 6 *hrs*, #Control lines: 12.



(c) Experiment completion time: ~ 5 *hrs*, #Control lines: 12.

Fig. 3. Layout and schedule of a genetic logic circuit: (a) Initial, (b) Obtained by *Fluigi* [9], and (c) Obtained by *GLCS*.

from G, where T is a set of 2-tuples $\{t_s(v), t_e(v)\}$ having $t_s(v)$ and $t_e(v)$ as the start and end time for vertex v, respectively. We formulate the problem as follows.

Input: DAG (G) for the given genetic logic circuit.

Output: Schedule of reactions and routing of flow and control layers in the layout.

Objectives: (i) To schedule the reactions of all genetic logic gates, (ii) to optimize the routing of flow and control layers in the layout, and (iii) to minimize the control line usage.

IV. Genetic Logic Circuit Synthesis

In this section, we present the step-by-step description of the proposed synthesis technique while considering different reaction times for each logic gate. This technique includes scheduling of reactions, placement and routing of flow and control layers in a flow-based microfluidic biochip. We refer this synthesis technique as genetic logic circuit synthesis (*GLCS*), the flowchart of which is presented in Fig. 4. The scheduling step of *GLCS* tries to minimize the experiment completion time (T_e) and the number of control lines (C) in the chip layout. Moreover, *GLCS* tries to optimize the routing of flow and control layers in the layout using 45°-routing, which in turn reduces the design cost.

A. Scheduling of Reactions

Scheduling is an important step in high-level synthesis, which decides the execution sequence of all the operations (reactions) while satisfying all the timing constraints. The DAG corresponding to a genetic logic circuit is taken as the

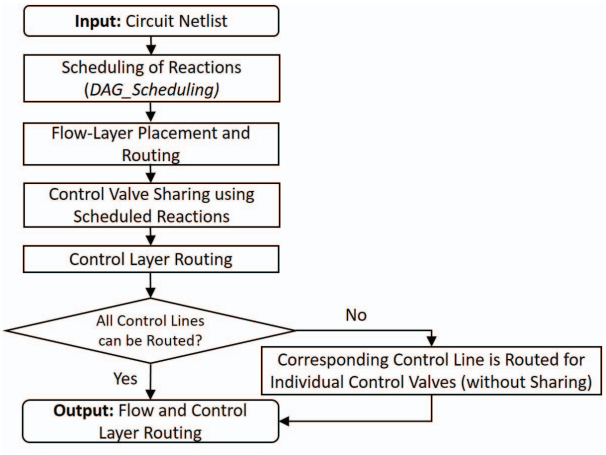


Fig. 4. Flowchart of the proposed synthesis technique *GLCS*.

input for the scheduling step *DAG_Scheduling*, the pseudocode of which is written by **Algorithm 1**. First, the critical path delay (T_e) of the DAG is computed using *ASAP* scheduling algorithm [13]. Next, the start time (t_s) and end time (t_e) for each gate are determined (line number $2-12$ of **Algorithm 1**).

For example, we consider the netlist of a genetic logic circuit with seven gates having reaction times in the range of 1.5 to 2 *hrs*. The circuit netlist and its corresponding DAG representation are shown in Fig. 5(a) and (b), respectively. Each vertex in the DAG (Fig. 5(b)) is associated with a pair $(v, t_r(v))$. First, the critical path (denoted by $B-E-F-G$) delay (7 *hrs*) is computed using *ASAP* on the DAG. After scheduling the reaction of each gate by *DAG_Scheduling*, the start and end times of each vertex are obtained and shown in Fig. 5(b) using a 2-tuple (t_s, t_e). Fig. 5(c) and (d) show the schedules of reactions obtained by *Fluigi* and *GLCS*, respectively. *GLCS* finds the schedule with $T_e = 7$ *hrs* as depicted in Fig. 5(d), whereas *Fluigi* results in a schedule with higher T_e (8 *hrs*) as shown in Fig. 5(c).

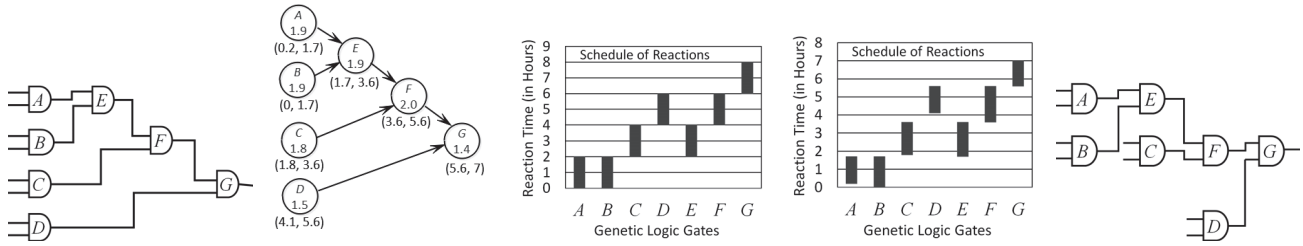
The scheduled DAG is used as the input to the flow-layer placement step, which then decides the correct location for each gate on the flow-based microfluidic biochip layout.

B. Flow-Layer Placement and Routing

Fluigi [9] consider the schedule of reactions of individual

Algorithm 1: *DAG_Scheduling*

Input: DAG $G(V,E)$ of a genetic logic circuit netlist.
Output: Scheduled DAG G' obtained from G.
begin
1 Compute the critical path delay T_e in G using *ASAP*;
2 ℓ = leaf vertex of G;
3 $Q \leftarrow$ Enqueue(ℓ); /* Q be a queue */
4 **while** Q *is NOT empty* **do**
5 $v \leftarrow$ Dequeue(Q);
6 **if** $v == \ell$ **then**
7 $t_e(v) = T_e$;
8 **else**
9 $t_e(v) =$ Start time of successor vertex of v;
10 $t_s(v) = t_e(v) - t_r(v)$;
11 **for** *each predecessor* p *of* v **do**
12 $Q \leftarrow$ Enqueue(p);



(a) (b) (c) (d) (e)

Fig. 5. (a) A netlist and its initial placement of 7 genetic logic gates. (b) DAG with t_r (in *hrs*) written inside each vertex and (t_s, t_e)-pair (in *hrs*) written below each vertex. (c) Schedule (with $T_e = 8$ *hrs*) obtained by *Fluigi* [9]. (d) Schedule (with $T_e = 7$ *hrs*) and (e) corresponding placement obtained by *GLCS*.

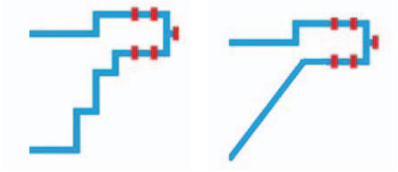
gates while getting their locations on the chip layout. It uses Manhattan distance measure as the microchannel length between any two logic gates that results in a placement where gates may be placed far from each other. Whereas, *GLCS* first places the output gate (leaf vertex of the DAG G') at the right-most location of the layout. In case of multiple output gates, they are equally distributed and placed at the right-most locations of the layout. Then it keeps on placing the gates with the same end times (in G') at the same column (i.e, at the left of the previous column) in the layout. This process repeats until all the input gates are placed. As a result, the layout will have higher routability in the flow-layer and more control valve sharing in the control-layer. Fig. 5(a) shows the initial placement of the given circuit, where gate D is placed far from its successor gate G. Whereas, *GLCS* places all the gates nearby their successors as shown in Fig. 5(e). After placement, A^* search algorithm can be used for flow-layer routing to connect the corresponding gates as in *Fluigi* [9]. Using Manhattan distance as the cost function for A^* search, it may result in many bends in the flow-layer routing. These bends can be reduced by assuming diagonal distance as the cost function of A^* search. In *GLCS*, 45°-routing is used to obtain better routing results.

C. 45°-Routing

Munson *et al.* reported that the presence of bends in the path of fluid flow decreases its flow pressure [14]. This pressure drop decreases the original flow rate of the fluid. In order to overcome this pressure drop the pumping power of the flow-layer needs to be increased. However, increasing the pumping power for the flow-layer reduces the lifetime of the microfluidic biochip. In *GLCS*, we use 45°-routing for both the flow and control layers to reduce the number of bends. After placement, 45° flow-layer routing is performed where *GLCS* finds the results using Manhattan and diagonal distance to obtain better routing results. For example, Fig. 6(a) and (b) presents an instance of flow-layer routing obtained by *Fluigi* and *GLCS*, respectively. It can be observed that the number of bends is reduced significantly by using 45°-routing in *GLCS* which will prolong the lifetime of the chip.

D. Control Valve Sharing

In current design practice the cost of control lines is high. Therefore, the number of control lines in a flow-based microfluidic biochip should be minimized. As shown in Fig. 2,



(a) (b)

Fig. 6. Flow-layer routing results obtained by (a) *Fluigi* [9]and (b) *GLCS*.

each gate is controlled by five control lines for five valves. In a larger circuit, using five control lines for individually controlling each gate increases the cost. Hence, after flow-layer placement and routing we try to find out sharing of control values based on scheduling of reactions. As shown in Fig. 3(c), the start time of genetic logic gate A is less than start time of gate B, i.e., $t_s(A) < t_s(B)$. Hence, the central valve of A can be shared with the entrance valve of B and we denote this control valve sharing as a sharing set $\{A, B\}$. Similarly, the control valve sharing $\{C, D\}$ between two gates C and D can be possible. Whereas, the end time of genetic logic gate A is less than start time of gate C, i.e., $t_s(C) = t_e(A)$. Hence, the entrance valve of A can be shared with the central valve of C (denoted by the sharing set $\{A, C\}$). Similarly, the control valve sharing $\{D, E\}$ between two gates D and E is also possible.

E. Control-Layer Routing

After control valve sharing for each sharing set a bounding box is considered. For example in Fig. 3(c), the bounding boxes are created for the sharing sets $\{A, B\}$, $\{A, C\}$, $\{C, D\}$, and $\{D, E\}$. The routing paths are estimated and kept in ascending order for further process. As the shorter control lines can be routed using smaller routing area, hence more space will be available for routing of larger control lines. Whereas, if the routing paths are processed in descending order, then there may be a possibility that shorter routes are blocked by the larger control lines. However, if all the control lines are not routed, then the corresponding control lines are routed individually (without sharing).

This method can also be used for different interconnection of genetic logic gates with the help of their schedule of reactions. For example, Fig. 7 shows the interconnection of three logic gates A, B and C with their schedule of reactions. Here gates are connected in series and only after the completion of previous reaction, the next reaction can start. Hence the output valves of previous gates can be shared with the entrance valve of next gate. As the schedule of reactions shows that

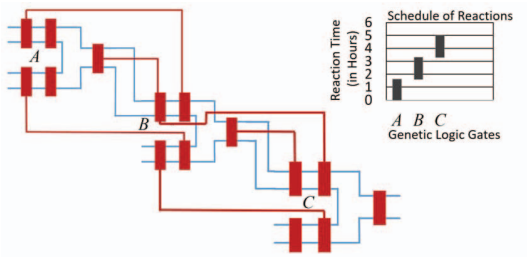


Fig. 7. Control valve sharing among 3 gates in a genetic logic circuit.

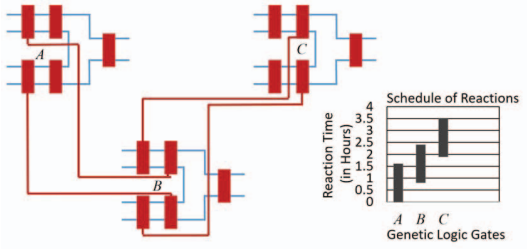


Fig. 8. Control valve sharing among 3 gates with overlapping reaction time.

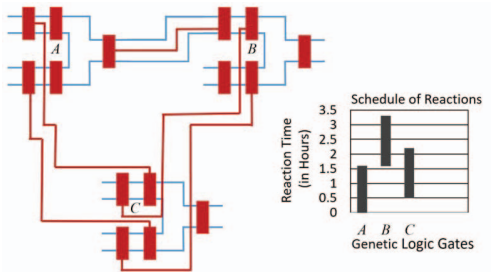


Fig. 9. Control valve sharing among 3 gates with some dependencies.

$t_s(B) = t_e(A)$ and $t_s(C) = t_e(B)$, the entrance valves of A can be shared with the central valves of B (denoted by sharing set $\{A,B\}$). Similarly, entrance valves of B can be shared with the central valves of gate C (i.e., $\{B,C\}$). The initial design of this circuit is controlled by a total of 15 control lines. However, *GLCS* can implement the circuit by only 9 control lines.

Fig. 8 shows the interconnection of three independent logic gates A, B and C with their overlapped schedule of reactions. Here each independent logic gate uses one control line for their output valve. As the schedule of reactions shows that $t_s(A) < t_s(B)$ and $t_s(B) < t_s(C)$, the entrance valves of A can be shared with the central valves of B denoting $\{A,B\}$. Similarly, entrance valves of B can be shared with the central valves of C denoting $\{B,C\}$. The initial design for this circuit is also controlled by a total of 15 control lines. Whereas, *GLCS* can control the circuit by only 11 control lines.

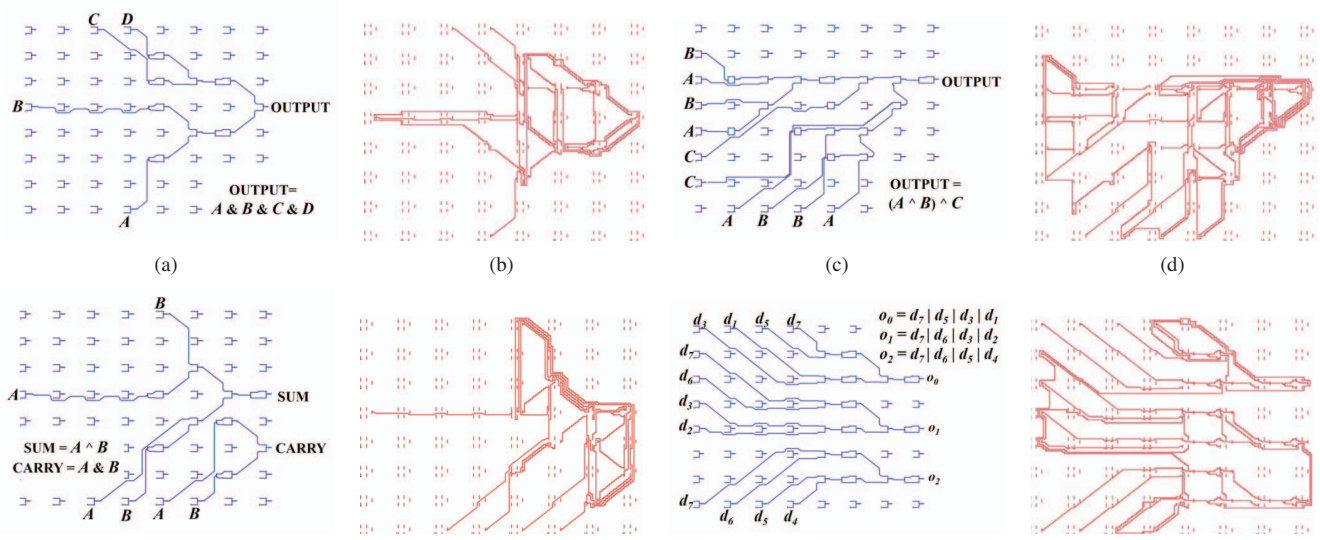
Fig. 9 shows the interconnection of three logic gates A, B and C with their schedule of reactions, where gate C is independent of A and B. However, the input of gate B is dependent on the output of gate A and the scheduling of reactions shows that $t_s(B) = t_e(A)$. Hence, the output valve of A can be shared with one of the entrance valves of B. As $t_s(A) < t_s(C)$ and $t_s(C) < t_s(B)$, the entrance valves of gate A and C can be shared with the central valves of the gate C and B denoting the sharing sets $\{A,C\}$ and $\{C,B\}$, respectively. Here, the total number of control lines used by *GLCS* is 10 that indicates a total reduction of 5 control lines compared to the initial design. The genetic logic circuits of Fig. 8 and 9 have some independent gate(s) in their netlists. This independency among logic gates indicates that these gates may be parts of different genetic logic circuits. Thus, we observed that *GLCS* is applicable for implementation of two or more genetic logic circuits on the same microfluidic biochip.

V. Simulation Results

We have implemented the proposed synthesis technique *GLCS* in Python 3.5.1 on a 3.6GHz Intel CPU with 8GB memory and simulated using four benchmark genetic logic circuits, whose detailed descriptions are listed in Table I. AND4 is a four input AND gate having single output [4], XOR3 is the three input XOR gate with one output [9], HALF ADDER is a simpler version of half adder with two outputs [11] and 8-3ENC is an 8-3 encoder having three outputs [9]. HALF ADDER and 8-3ENC are complex benchmarks compared to AND4 and XOR3 that contain more than one circuit in their layouts. In an 8×8 size microfluidic chip there are 42 gates, 22 input ports and 210 microvalves. Each gate is controlled using five valves while input and output are controlled by one valve. Three features have been considered for each benchmark circuit such as the number of gates (H) used out of 42 gates, the number of input (I) ports used out of 22 input ports and the number of valves (J) used out of 210 valves for implementation of the circuit on the chip layout.

Fig. 10 shows the layout results for both flow (in 'blue') and control layers (in 'red') for all four benchmark circuits (Table I). In Fig. 10(a), the layout of AND4 is shown that uses 9 gates and 5 input-output ports. Therefore, the whole circuit uses a total of 49 microvalves. Fig. 10(c) represents the layout of XOR3 with 16 gates, 10 input-output ports and a total of 90 valves. The control layer layout for HALF ADDER is presented in Fig 10(e) with 9 gates, 6 input-output ports and a total of 51 valves. Similarly, the layout of 8-3ENC using 18 gates, 12 input-output ports and a total of 102 gates is shown in Fig 10(g). For all four benchmark circuits the corresponding control layer routing results are shown in Fig. 10(b), (d), (f) and (h). The scheduling of reactions in *GLCS* helps in reducing the number of control lines by sharing between two different gates within the same/different circuit(s).

Table II presents comparative results of control lines (C) and experiment completion time (T_e) obtained by initial design, *Fluigi* and *GLCS*. Simulation results for *GLCS* show an average reduction of 61% in C with same T_e compared with initial designs. Whereas, *GLCS* reduces T_e by 13.4% (on average) at the cost of increasing C by 42% compared with *Fluigi* [9]. Table III shows the comparative values of the total length of flow lines (L_f), total length of control lines (L_c) and total length of microchannels ($L_m = Lf + L_c$) obtained *Fluigi* and *GLCS* for all four benchmark circuits. *GLCS* provides average reductions of 28% and 10.4% in L_m compared with *Fluigi* [9] without and *Fluigi* with 45°-routing, respectively. Whereas, *GLCS* shows average increments of 77.6% and 33.6% in L_c and L_m, respectively, compared with *Fluigi*.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 10. (a, b) Flow and control-layer routing results of AND4. (c, d) Flow and control-layer routing results of XOR3. (e, f) Flow and control-layer routing results of HALF ADDER. (g, h) Flow and control-layer routing results of 8-3ENC.

TABLE I
DESCRIPTIONS OF BENCHMARK CIRCUITS [9].

Circuit Name	#Gates Used (H)	#Input Ports (I)	#Valves ($J = 5H + I$)
AND4 [4]	13	4	49
XOR3 [9]	26	10	90
HALF ADDER [11]	15	6	51
8-3ENC [9]	30	12	102

TABLE II
TOTAL NUMBER OF CONTROL LINES (C) AND EXPERIMENT COMPLETION TIME (T_e IN *hrs*) OBTAINED INITIALLY [8], BY *Fluigi* [9] AND *GLCS*.

Circuit Name	C			T_e			T_c
	Initial	*Fluigi*	*GLCS*	Initial	*Fluigi*	*GLCS*	*GLCS*
AND4	49	17	19	6.8	8	6.8	18.79
XOR3	90	29	37	12.6	14	12.6	48.36
HALF ADDER	51	17	21	7.2	8	7.2	18.73
8-3ENC	102	17	35	6.5	8	6.5	72.58

TABLE III
COMPARISON OF TOTAL LENGTH OF FLOW LINES (L_f), TOTAL LENGTH OF CONTROL LINES (L_c) AND TOTAL MICROCHANNEL LENGTH ($L_m = L_f + L_c$) OBTAINED BY *Fluigi* [9] [8] AND *GLCS* FOR ALL THE BENCHMARKS.

Circuit Name	L_f (in Pixels)			L_c (in Pixels)		L_m (in Pixels)	
	Fluigi	*Fluigi with 45°*	*GLCS*	*Fluigi*	*GLCS*	*Fluigi*	*GLCS*
AND4	445	376	299	610	1199	1055	1498
XOR3	890	687	666	1009	1699	1899	2365
HALF ADDER	623	536	413	714	1720	1337	2133
8-3ENC	770	594	585	1477	2152	2247	2737

VI. CONCLUSIONS

We proposed a synthesis technique *GLCS* that performs scheduling, placement, routing and control line minimization for implementation of a genetic logic circuit on microfluidic biochip. This technique reduces the total experiment completion time compared to *Fluigi* by considering the different reaction time for each gate. Simulation results show that *GLCS* reduces the experiment completion time with a trade-off of increased number of control lines compared with *Fluigi*. Whereas, *GLCS* is reduces the total number of control lines from the initial design while having the same experiment completion time. Hence, *GLCS* is preferable to *Fluigi*, if the experiment completion time is more important to be reduced compared to control line minimization.

VII. ACKNOWLEDGEMENT

The authors wish to thank Dr. Doug Densmore, Department of Electrical and Computer Engineering, Boston University, USA for useful discussions.

REFERENCES

[1] R. Wellhausen and K. A. Oye, "Intellectual property and the commons in synthetic biology: Strategies to facilitate an emerging technology," in *Atlanta Conference on Science, Technology and Innovation Policy*, 2007, pp. 1–2.
[2] K. Clancy and C. A. Voigt, "Programming cells: Towards an automated 'genetic compiler'," *Current Opinion in Biotechnology*, vol. 21, no. 4, pp. 572–581, 2010.
[3] J. A. Brophy and C. A. Voigt, "Principles of genetic circuit design," *Nature methods*, vol. 11, pp. 508–520, 2014.
[4] T. S. Moon, C. Lou, A. Tamsir, B. C. Stanton, and C. A. Voigt, "Genetic programs constructed from layered logic gates in single cells," *Nature*, vol. 491, pp. 249–253, 2012.
[5] A. Prindle, P. Samayoa, I. Razinkov, T. Danino, L. S. Tsimring, and J. Hasty, "A sensing array of radically coupled genetic biopixels," *Nature*, vol. 481, pp. 39–44, 2012.
[6] L. Nissim and R. H. Bar-Ziv, "A tunable dual-promoter integrator for targeting of cancer cells," *Molecular systems biology*, vol. 6, p. 444, 2010.
[7] S. K. Lee, H. Chou, T. S. Ham, T. S. Lee, and J. D. Keasling, "Metabolic engineering of microorganisms for biofuels production: from bugs to synthetic biology to fuels," *Current opinion in biotechnology*, vol. 19, no. 6, pp. 556–563, 2008.
[8] H. Huang and D. Densmore, "Fluigi: Microfluidic device synthesis for synthetic biology," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 11, no. 3, pp. 26:1–26:19, 2014.
[9] H. Huang, "Fluigi: An end-to-end software workflow for microfluidic design," Ph.D. dissertation, Boston University, 2015.
[10] A. Tamsir, J. J. Tabor, and C. A. Voigt, "Robust multicellular computing using genetically encoded nor gates and chemical wires," *Nature*, vol. 469, pp. 212–215, 2011.
[11] J. Beal, T. Lu, and R. Weiss, "Automatic compilation from high-level biologically-oriented programming language to genetic regulatory networks," *PLoS ONE*, vol. 6, no. 8, 2011.
[12] L. Gonick and M. Wheelis, *Cartoon guide to genetics*, 1991.
[13] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*, 1994.
[14] B. R. Munson, D. F. Young, T. H. Okiishi, and W. W. Huebsch, *Fundamentals of Fluid Mechanics*, 1999.