Scientia Iranica D (2023) 30(4), 1279–1295

Sharif University of Technology
Scientia Iranica
Transactions D: Computer Science & Engineering and Electrical Engineering
http://scientiairanica.sharif.edu

RTL2DNA: An automatic flow of large-scale DNA-based logic circuit design

Z. Beiki[a] and A. Jahanian[b,*]

a. *Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.*
b. *Faculty of Computer Science and Engineering, Shahid Beheshti University, G.C., Velenjak, Tehran 19839-63113, Iran.*

Received 8 December 2019; accepted 14 March 2022

KEYWORDS
DNA circuit;
DNA computer;
Micro-architecture;
DNA design flow;
Automatic design flow.

Abstract. DNA computing is a new kind of computation for solving complex problems with significant parallelism. Research findings indicate that DNA-based logic systems can be useful in many biomedical applications such as early cancer detection. DNA logic systems have been applied successfully to detect the risky patterns of nucleotide-based cancer biomarkers (microRNAs). Detection of real diseases requires large-scale DNA-based logical systems. Therefore, the issue of large-scale DNA-based logic circuits is a crucial research topic. In this paper, an automatic design flow is proposed to facilitate the design, verification, and physical implementation of multi-stage and large-scale DNA logic circuits. Digital Microfluidic Biochips (DMFB) have been used recently as a promising platform for efficient implementation of DNA-based computing systems and circuits. We used this technology as the physical platform for implementation of DNA-based circuits. Our experiments and implementations show the feasibility, accuracy, efficiency, and simplicity of the proposed design flow. Final DNA reactions that are synthesized by the proposed design flow are verified and simulated using stochastic DNA-reaction simulators to prove the correctness of the proposed design flow. This design flow can open a new horizon for researchers and scientists to design, implement, and evaluate the DNA-based logic systems.

1. Introduction

In the past, DNA molecules were primarily known for their role in producing proteins and transmitting genetic traits to future generations. However, in recent decades, it has been discovered that DNA molecules can also be utilized for a novel form of computing [1]. Adleman appropriated DNA molecules to solve the Hamiltonian path problem [2]. Adleman's experiments indicate that NP-complement problems can be solved using DNA molecules while this could not be done with silicon-based computers, thanks to the significantly huge degree of parallelism. DNA computation is a bridge between biological science and computer engineering. Some applications of DNA computing include Gen analysis, medical therapeutics, pharmacy, and solving NP-complete/NP-hard problems [3,4].

Bio-computing methods, especially DNA computing and DNA nanostructures design methods, have been developed over the last two decades to realize and control matter on the nano scale [5]. The following provides a review of the previous work on DNA-based logic gate design.

The concept of localized DNA strand displace-

*. *Corresponding author. Tel.: +98 21 29904188*
E-mail addresses: z.beiki@eng.ui.ac.ir (Z. Beiki); jahanian@sbu.ac.ir (A. Jahanian)

doi: 10.24200/sci.2022.54993.4017

ment was initially proposed by Sakamoto et al. [6], this proposed method is a mechanism for implementing the chemical reaction network on the surface of a DNA nanostructure. In [7], the authors proposed a method of AND/OR functions implementation based on DNA strands. Their proposed structure consists of micro-reactors along with attached heating elements utilized for controlling the DNA annealing process. This structure can be used to solve the satisfiability problem in a linear space within a quadratic time [7]. DNA localized circuits, proposed by Qian and Winfree in 2014, accelerate the kinetics by increasing the relative concentration of strand reaction [8]. Fan et al. [9] proposed a three-input label-free/enzyme-free majority gate through DNA hybridization without DNA replacement and enzyme catalysis; further, the system is capable of implementing various basic/cascade logic gates.

The Seesaw logic model was a breakthrough for circuit design based on DNA strands. This model was proposed by Qian et al. in 2008 [10] and further improved in 2011 [11]. Their methodology increased scalability and reliability of DNA circuits compared with earlier methods; however, this method increased the number of strands.

Recently, DNA logic systems have been utilized successfully to detect microRNAs. MicroRNAs are biomarkers based on nucleotides whose concentration changes in different types of cancer [12] and [13]. Hemphill and Deiters applied the DNA logic gates to create a molecular system with microRNAs inputs for cancer detection [12]. DNA circuits recognize microRNA cancer biomarkers through strand hybridization.

Microfluidic biochips are controlled and automated platforms that are used for chemical reactions [14]. These chips are known as a promising platform for executing DNA operations in a controlled process. Van [15] discussed a flexible configuration platform for performing DNA computation on a microfluidic architecture in order to realize basic logic structures such as switches, memories, and logic gates. Their proposed design is capable of programming DNA strands into various Boolean problems. However, each technique comes with its own advantages and disadvantages.

More than 40 years of electronic design experience indicate that Computer-Aided Design (CAD) plays a fundamental role in progress of the VLSI. Really, the benefits and drawbacks of an emerging technology cannot be evaluated without a suitable CAD tool. The progress of DNA computing will be boosted when there is an efficient CAD environment that enables the researcher to implement and evaluate their ideas on this platform [16]. Some very limited design flows have been proposed in recent years for simulating DNA reactions. Dwyer [16] presented a scientific tool for DNA self-assembly design that enables the design of Small-Scale Integration (SSI) and Medium-Scale Integration (MSI) systems with DNA strands. Circuit design with DNA self-assembly associated with many of the challenges contains costs and yield [16]. DNA self-assembly circuits are not expandable and they often are used as a base to build other structures [17,18]. Another design tool was proposed by Selnihhin and Ebbe Sloth [19] for DNA origami structures. They provided a small tool for designing simple circuits such as a simple DNA origami biosensor device. A placement algorithm was proposed in [20] for localized DNA logic circuits. In [21], seesaw compiler toolbox was employed to convert the AND/OR circuits into dual-rail seesaw logic circuits. Output of seesaw compiler can be simulated and evaluated using the standard Visual-DSD simulator.

DENA architecture is a configurable DNA Architecture based on microfluidic biochips that are used for the implementation of DNA large-scale circuits. This proposed method is introduced in [22]. DENA can improve the cascade-ability and feasibility of DNA circuits; in addition, the basic concepts of configurable DNA architectures were described in this work. However, this paper does not address any automatic design flow for this technology.

As mentioned before, considerable contributions have been addressed on design and analysis of DNA-based logic gates and circuits for computing and medical applications. However, no scalable and easy-to-use design flow is introduced for the DNA-based logic systems. This problem makes the design and evaluation of DNA-based systems difficult and even infeasible for researchers and designers of this scope.

This paper proposed a scalable and automatic design flow for DNA circuits facilitating synthesis, simulation, and implementation of DNA logic systems automatically. This design flow is called RTL2DNA in this paper. RTL2DNA flow provides a feasible and straightforward design flow from RTL to physical implementation of combinational logic circuits based on DNA strands. Microfluidic Biochips (MFBCs) are used in this paper as the infrastructure of DNA logic gates. In other words, the proposed design flow implements the DNA logic gates on MFBCs. The main contributions of this paper are given as follows:

- Providing an RTL-to-DNA simulation/synthesis flow for large-scale DNA-based logic circuits that enable the design, implementation, and verification of large DNA circuits. This feature enables researchers to implement and evaluate their ideas in this area easily;
- Using the microfluidic biochips as the implementation platform provides a controlled framework for re-

alizing the DNA-based logic circuits. The proposed design flow synthesized an RTL logic circuit into the biochemical assay, which can be executed on Digital Microfluidic Biochips (DMFBs);

- We used the microarchitecture in [22] as the base technology. A highlight feature of the used microarchitecture is that the number of required strands in the proposed method is constant for all the circuits and does not increase for large circuits. This feature is a key contribution to the implementation of large Boolean systems based on DNA logic gates.

The rest of this paper is organized as follows. Section 2 provides a brief review of DNA-based logic design styles. Section 3 reviews the digital microfluidic technology and its application for DNA computing. Section 4 explains the used DNA micro-architecture and the improvements that are made on it. The proposed design flow and the experimental results of RTL2DNA flow are described in Sections 5 and 6, respectively. Finally, Section 7 concludes the paper.

2. DNA-based logic circuit design

The Watson-Crick complementary rules are the foundation of DNA-based computation. These rules define reactions between DNA strands. Each DNA strand is included in a finite set of nucleotide acids that are encoded by "Adenine" (A), "Guanine" (G), "Cytosine" (C), and "Thymine" (T). Authors in [23,24] proposed different methods for designing DNA-based logic gates and there are mainly categorized into enzyme-based and enzyme-free categories. The enzyme-free category is recommended because of the lower costs, the speed of operation, and the simplicity of implementation. The enzyme-free method is called Toehold-mediated because these reactions use a small and fast DNA domine in the name of Toehold. This domain starts the reactions in the DNA displacement process. In this section, the concept of Toehold-mediated design and Seesaw DNA logic (as the most used method) are described briefly.

2.1. Toehold-mediated logic design

Toeholds start the reaction in the strand displacement method. In this method, we do not have any enzyme compared with enzyme-based methods. Therefore, the strand displacement method is faster and cheaper. The most important disadvantage of the strand displacement method is the great number of orthogonal strands. By increasing the size of the circuit, the orthogonal strands increase sharply.

In an article in 2000, Yurke et al. stated that the probability of binding two complementary DNA strands was dependent on the reverse of their length or the number of bases in strands. In simple terms, the shorter complement strands have a higher merging probability [25]. In fact, small strands in [25] were the Toehold strands called Primer in biological science fields.

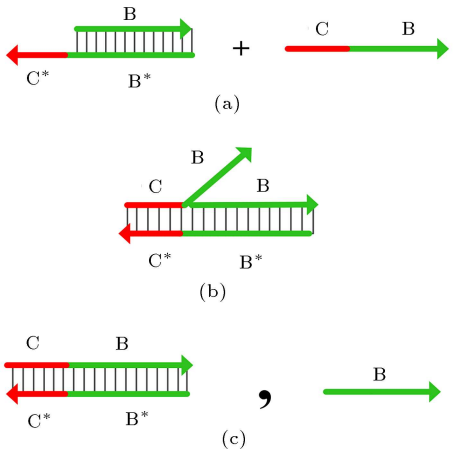


Figure 1. A simple view of Toehold-mediated strand displacement.

In 2000, Yurke et al. [25] used Toehold as part of their attempt to accelerate the merging of DNA strands. They found that applying the Toehold strand improved the controllability of the DNA displacement process. Figure 1 shows a simple example with strand displacement method. Toehold strands are illustrated in red color; they are reaction starter strands. Strand displacement reactions continue to be used after energy is released from Toehold reaction. Green strands represent a single DNA strand with its direction showing the merging direction (from 5 to 3). The labels of the complement strand are represented by the quote sign (*).

Figure 1 illustrates a strand displacement reaction which uses Toehold strands for starting. Figure 1(a) shows a complex strand consisting of Toehole C* and double strand B (BB*). This complex strand reacts with a single upper strand CB. As shown in Figure 1(b). Initially, Toehold C* of complex strand binds to its complementary strand (C in single strand CB). The energy released from this Toehold binding leads to continued reaction; therefore, strand B in CB is replaced with strand B in the initial complex strand (Figure 1(c)). Finally, reaction result, as shown in Figure 1(c), consists of double-stranded CB and single-strand B.

The kinetic of Toehold strands depends on the sequence of nucleotides and the length of strands. The speed of strand displacement reactions regulates with Toehold mediated [26]. Typically, lengths of Toehold strands range from 3 to 7 nucleotides.

2.2. Seesaw logic gate

Qian and Winfree proposed the Seesaw logic design technique in 2008, which is known as an efficient Toehold-mediated DNA logic design style [10]. A

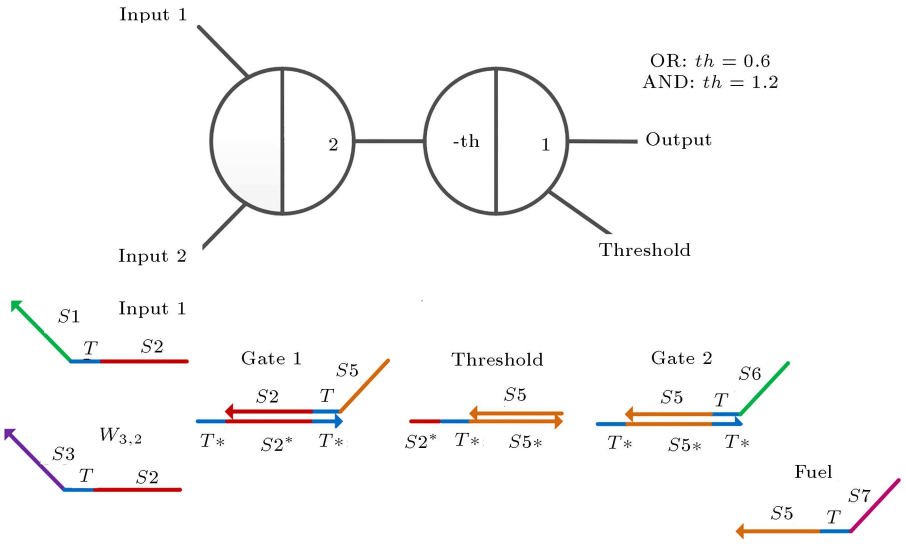


Figure 2. Internal strands and operations of the AND/OR seesaw logic gate [10].

Seesaw logic gate is comprised of five strand types: inputs, outputs, gate, threshold, and fuel. The output strand is generated as a result of reactions between input, gate, and threshold strands.

Seesaw gates apply two basic components based on Toehold-mediated method. The input strands react catalytically so that a single-input strand is capable of releasing multiple output strands from several Seesaw gates. On the other hand, the output strands act as the input strands for the next Seesaw gate. A Seesaw-based system consisting of 'n' inputs and 'm' outputs incorporates 'n' input strands, 3 internal strands (gate, threshold, and fuel strands), and m output strands.

Figure 2 shows the internal structure of an OR gate. This element can compute either (AND/OR) functions depending on the initial concentration of the Threshold DNA strand. Initial Threshold concentration is 600 nM for an OR gate and the concentration of strands should be increased to 1200 nM to generate an AND gate.

Input strands (Input 1 and Input 2) in Figure 2 react with Gate 1 to produce upper strand $S_2.T.S_5$. This strand combined with Threshold strand consists of two Toehold domains. Therefore, the concentration of $S_2.T.S_5$ can be controlled by Threshold concentration. The remaining $S_2.T.S_5$ reacts with strands Gate 2 and $S_2.T.S_6$ and produces the output strand. It is worth noting that fuel strand increases the chance of reactions to generate the output strand.

The major advantage of seesaw logic gates is the cascading capability of this design style since the output concentration of seesaw does not degrade considerably, such that it can be used as the input of the next stage of the system. However, the seesaw design methodology is subject to some limitations as follows:

- Lacking of the inverter gate (NOT gate) complicates designing gates in the seesaw logic greatly. Dual-rail logic is applied to seesaw to overcome this weakness at a cost of considerable overheads in the number of strands and system complexity;
- Unintended reactions (i.e., crosstalk) increase with the growing number of gates such that designing large circuits is infeasible in practice;
- Large number of orthogonal strands is required for medium and large-sized circuits such that it is quite difficult to design large circuits.

Seesaw design methodology is appropriate for SSI logic design with many DNA strands, but it is infeasible in case of larger circuit designs.

3. Digital microfluidic and DNA computing

A DMFB device is a platform for performing operations of biological assays in an automatic and controllable manner [27]. A DMFB can automatically manage droplets reaction on the two-dimensional silicon array. The attempt is to program the electrodes under surface. Figure 3 illustrates the structure of DMFBs.

The core technology realizing DMFB devices is Electro-Wetting on Dielectric (EWOD) [27]. This technology works based on an electrical field applied to the electrode beneath the droplet in order to actuate (move) the droplet [27]. Movement of droplets across the electrodes allows for fundamental microfluidic operations such as holding (storing), transporting (moving), merging, mixing, and splitting. Further, the chips can accommodate other operations such as detecting, heating, and cooling by means of external equipment integrated into the chip during the manufacturing time.

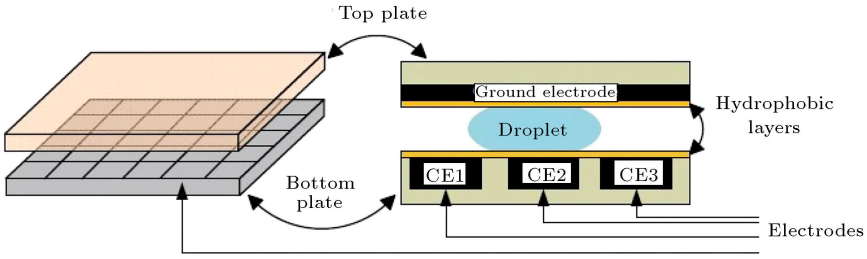


Figure 3. The structure of a digital microuidic biochip [27].

3.1. DMFBs as the platform for DNA computing

Chemical reactions between the DNA strands represent manual operations. DNA computing is quite challenging. Recently, with the development of DMFB, automatic control of DNA computing reactions is hopeful and almost realistic [28]. Furthermore, microfluidic biochips can be utilized for providing scalability and flexibility demanded by large scale DNA logic circuit designs [22].

The following items describe the proposed techniques for using the MFBCs in various steps of DNA computing:

- **Annealing**: Temperature ramp (annealing) is the most important operation in DNA computing. Malic et al. [29] indicated that temperature changing was possible in the DMFB. They proposed a practical system for a scalable and flexible mechanism to change the temperature on an MFBC for DNA strand displacement;
- **Strand separation**: Strand separation is required for separating the output strands of a computation from the environment to be used as the input strands of another DNA operation. Strand separation on the microfluidic biochip was conducted in [30] with a very good resolution;
- **Polymerase Chain Reaction (PCR)**: PCR is an effective technique for producing the input strands of a DNA computer and improving the concentration levels and strand reactions in DNA computing. Microfluidic platforms were utilized successfully in the referenced studies [28,31,32].

As mentioned before, DMFBs are a revolutionary solution for implementation of DNA-based computing system and circuits. We used DMFBs as the controlling platform for executing the DNA operations.

4. DNA-based micro-architecture

In [22], a micro-architecture (DENA) was proposed for large-scale circuits based on DNA computing. The proposed architecture design was inspired by FPGA and was configurable and useful for any automatic or semi-automatic design flow. Our design flow uses the revised version of DENA micro-architecture for automatic DNA-based circuit synthesis. DENA is a regular architecture that consists of the arbitrary number of DENA Clusters (DCs), which can be configured to implement a 4-input logic function. Figure 4 shows the general structure of DENA micro-architecture with 2×2 DCs. As shown in this figure, each DC is implemented on a 3×3 grid of a microfluidic biochip.

The detailed structure and operation of this micro-architecture was described in [22]. We used the DENA [22] as the logical micro-architecture. We made some improvements on DENA to increase its capabilities for automatic synthesis. The rest of this section describes these improvements.

4.1. Improvements applied to DLB design

DNA Logic Block (DLB) is the basic functional element of a DC. Original DLB in [22] is implemented with 7 DNA-based configurable OR/AND gates (COA), as shown in Figure 5(a). It implements a Sum-of-Product (SOP) or Product-of-Sum (SOP) with 8 inputs according to COA configuration. In this paper, DLB is upgraded from a 2-level AND/OR to a 4-input lookup table that brings better performance/overhead tradeoff (Figure 5(b) and (c)).

As can be seen in Figure 5(c), the modified DLB contains 16 5-input AND gates. Therefore, 80 orthogonal DNA strands are required only for input of AND gates that are implemented using the seesaw design method. On the other hand, DCs contain 4 inputs as MUX select pins. Each DLB input is connected to 8 different AND gates and hence, it should be converted into 8 orthogonal strands. We used conventional seesaw amplifying gate for generating 8 orthogonal strands. The schematic view of the amplifying gate is shown in Figure 6. More details about the amplifying gate can be found in [33].

We utilized 8 amplifying seesaw gates in this paper. Each amplifying gate received one input (A to D or $\bar{A}$ to $\bar{D}$) and produced 8 different strands. DLBs have 4 input strands, 1 output strand, and 16 AND gates. The inputs of AND gates are different from each other. The threshold of AND gate is between $n-1$ and n input concentration when inputs are ON. The

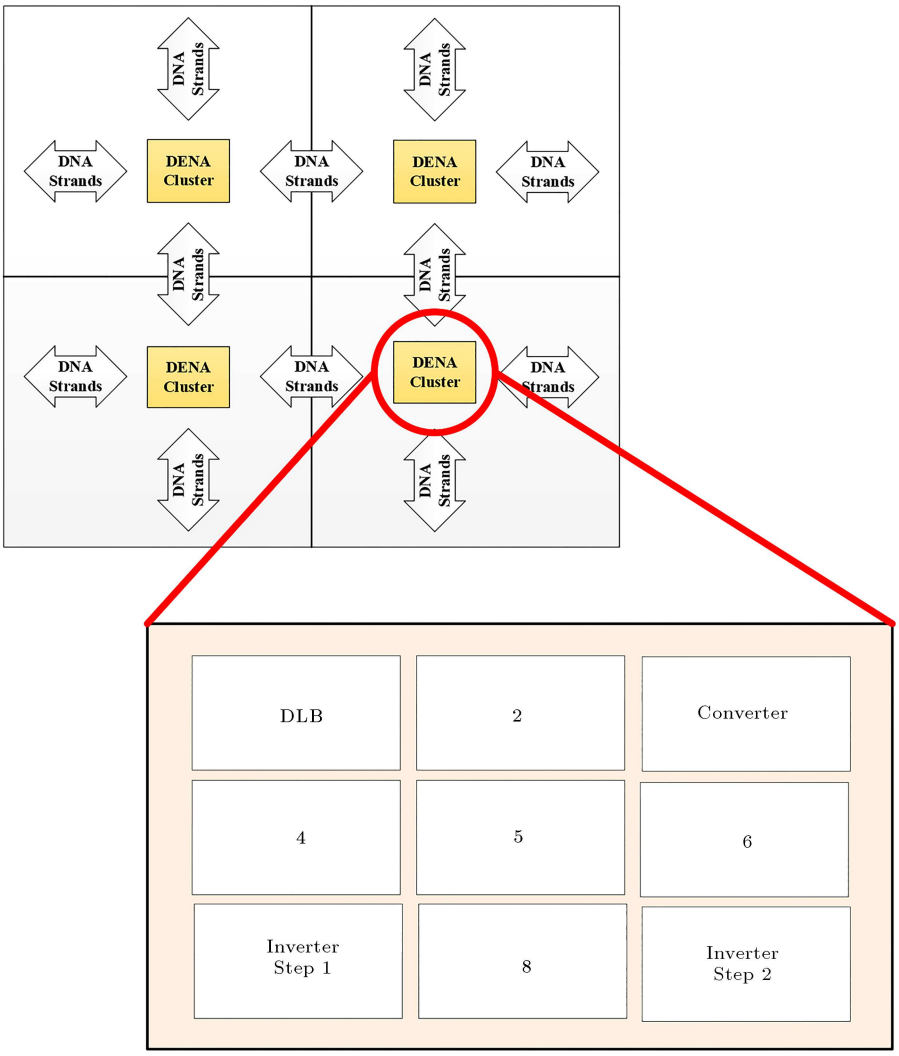


Figure 4. A simplified view of the DENA architecture proposed in [22].

threshold concentration for OR gate with n-inputs is between $n * C_{off}$ and $1 * C_{on}$ when C_{off} and C_{on} are the concentration for OFF and ON input modes (100 nM and 900 nM) [33]. AND/OR in DLB tile is similar to AND/OR seesaw gate, as shown in Figure 4, only by increasing the threshold concentration.

Each DLB is implemented with 16 AND gates, 5 OR gates (OR network), and one amplifying gate. Therefore, each DLB contains 320 different strands among which 80 strands are used for the amplifying gate while 4 strands are utilized for inputs.

4.2. Modifications applied to the inverter

As mentioned before, there is no straightforward solution for DNA-based Inverter design so that the design of an inverter gate has been a serious challenge in DNA logic gate design. In [22], a 2-stage inverter gate (NOT gate) was proposed. In this paper, we modified this NOT gate and used 4 numbers of NOT gates in each DC. Each NOT gate was used for one input (A to D) and the output of the inverter was sent to DLB. In this way, DLB had 8 inputs (A to D and $\bar{A}$ to $\bar{D}$).

Moreover, each inverter (Step 1 and Step 2) has 2 different strands and we use 4 inverter gates. Therefore, 8 different strands are used for inverter gates.

4.3. Improvements of the converter

The output strands of a DLB should be used as input strands of the succeeding DLBs. A converter gate presented in [22] reshapes the output of a DLB to the form of the first input of next DLB. We revised the converter enabling transformation of the DLB output to an arbitrary input of the next DLB. This modification improves the logic utilization and flexibility of the design. An amplifying gate with 4 outputs is utilized to design the converter stage. DLB output is sent to the converter stage (amplifying gate) and then, one of 4 converter outputs is sent to the next DC according to routing between DCs. The converter used 6 new strands. Fuel concentration for this amplifying gate is

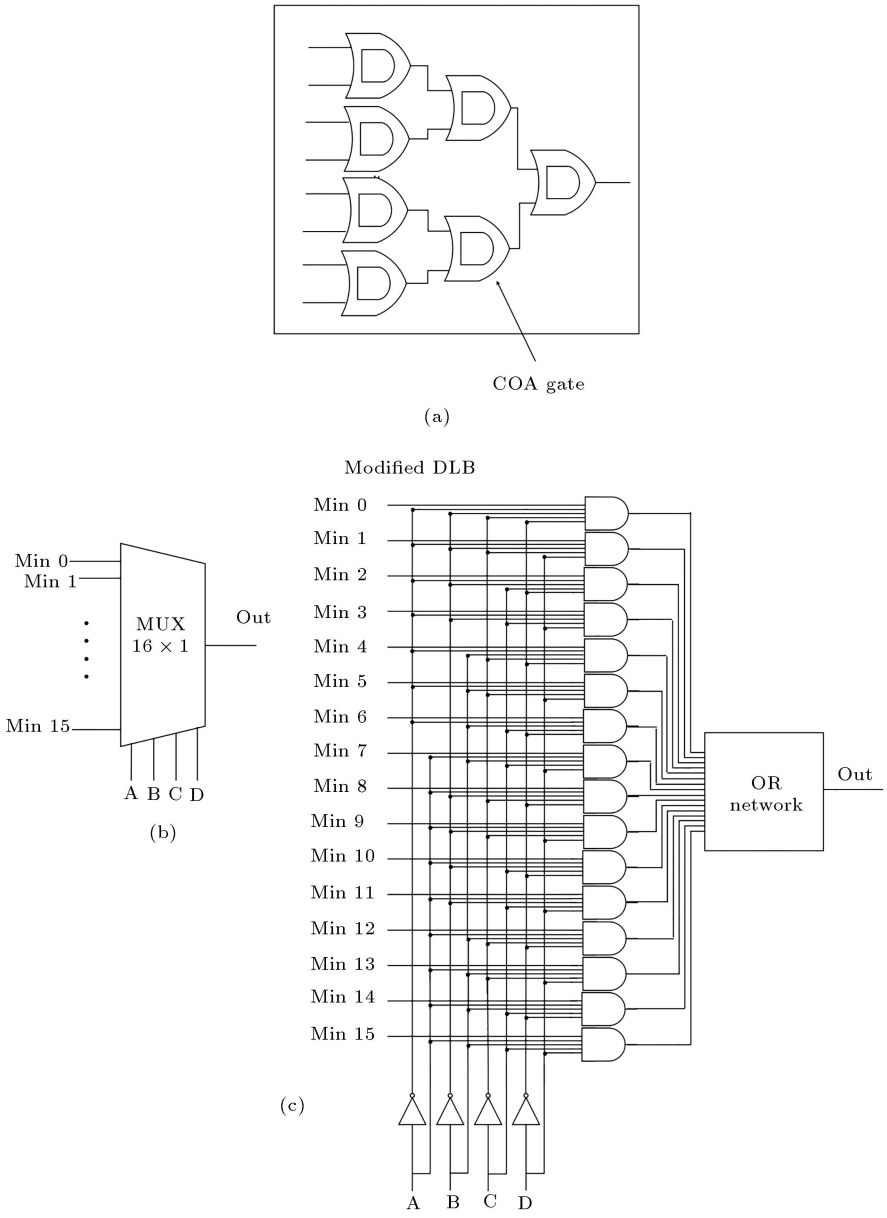


Figure 5. (a) Structure of two-level AND/OR DLB in [22]. (b) Improved DLB which implements a 4-input lookup table. (c) Gate-leve-schematic of DLB.

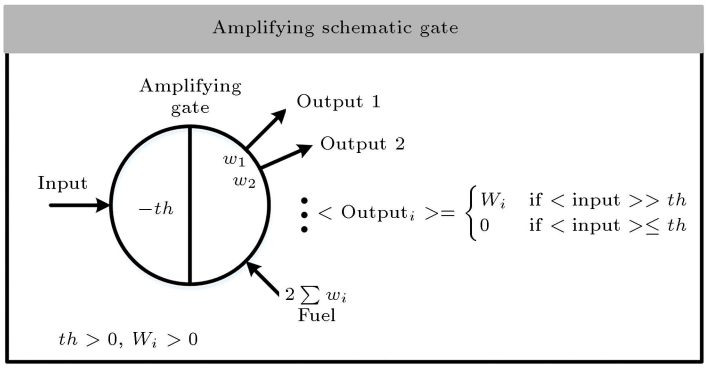
800 nM and another concentration is similar to that shown in Figure 6.

5. The proposed design flow

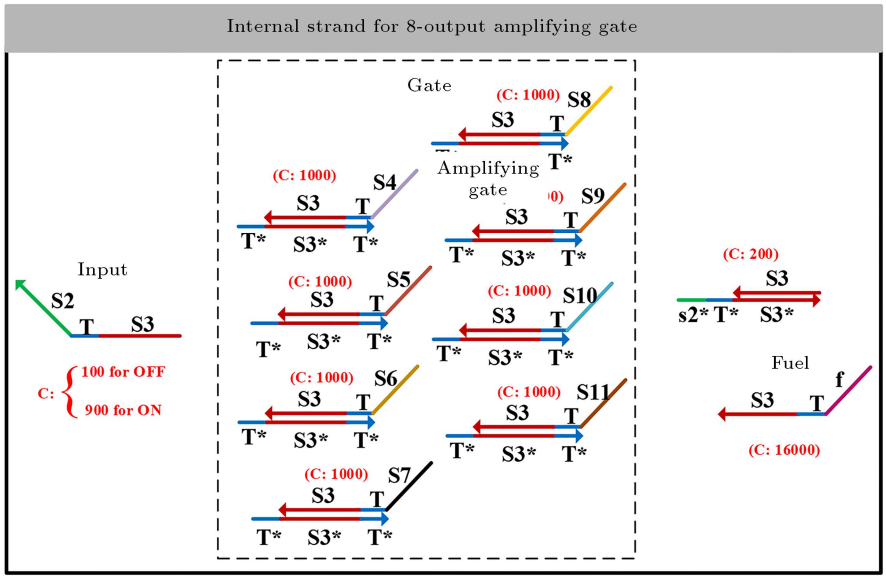
The main contribution of this paper is introducing an automatic design flow for large-scale DNA circuits based on the improved version of DENA micro- architecture [22]. We revised the conventional electronic design flow and added some new steps to it for adapting to the DNA-based logic design. Figure 7 shows the overall design process.

The suggested design cycle starts from an RTL description of the design. Berkeley ABC tool [34] is used as a technology-independent logic synthesis toolbox and TV-Pack [34] is modified to map the synthesized (output of ABC) netlist to DENA micro-architecture. After this mapping, the design cycle is divided into 2 sub-trees as follows:

- **Simulation and verification**: Left-hand sub-tree of RTL2DNA in Figure 7 involves simulation and verification. In the proposed flow, the synthesized netlist is converted into a sequence of reactions between the DNA strands that implements the circuit logic function. The DNA-based logic circuit can be simulated and verified by a stochastic DNA reactions simulator such as Microsoft Visual-DSD tool [35]. Visual-DSD is a well-known DNA strand displacement modeling and simulation tool



(a)



(b)

Figure 6. Structure and internal strands of the amplifying gate [34].

that is widely used in research related to DNA computing, especially for DNA logic gate design. It is worthwhile to note that the synthesized circuit is mapped to Visual-DSD input model based on DENA microarchitecture;

- **Physical design**: The right sub-tree of Figure 7 represents the physical design of the synthesized circuit on the microfluidic platform. Microfluidic biochips are responsible platforms for running a chemical assay (a sequence of chemical operations). On the other hand, each DNA computation consists of a sequence of biochemical operations that can be encoded as an assay. In RTL2DNA design flow, synthesized logic description is mapped to a biochemical assay to be executed on a microfluidic biochip. We used Static Synthesis Simulator (SSS) to implement the circuit on DMFB [36]. SSS is an open-source framework designed for supporting algorithmic and software-driven control for DMFBs. SSS toolbox [36] is revised in this research to place and route the mapped function on DENA micro-architecture over the microfluidic biochip platform. Finally, a placed and routed circuit on the microfluidic platform is generated which realizes the projected circuit on this platform.

The following subsections describe the components of RTL2DNA in more details.

5.1. Logic design flow

In this step, RTL description of the projected circuit (in Verilog) is synthe- sized into gate level and then, is mapped to DENA microarchitecture. It is notable that VTR) toolbox is a worldwide collaborative effort to provide an open-source framework for conducting FPGA architecture and CAD research and development [34].

We revised the TV-Pack (a component of VTR) for partitioning and mapping the design to DENA architecture. The mapped netlist shows the DC index

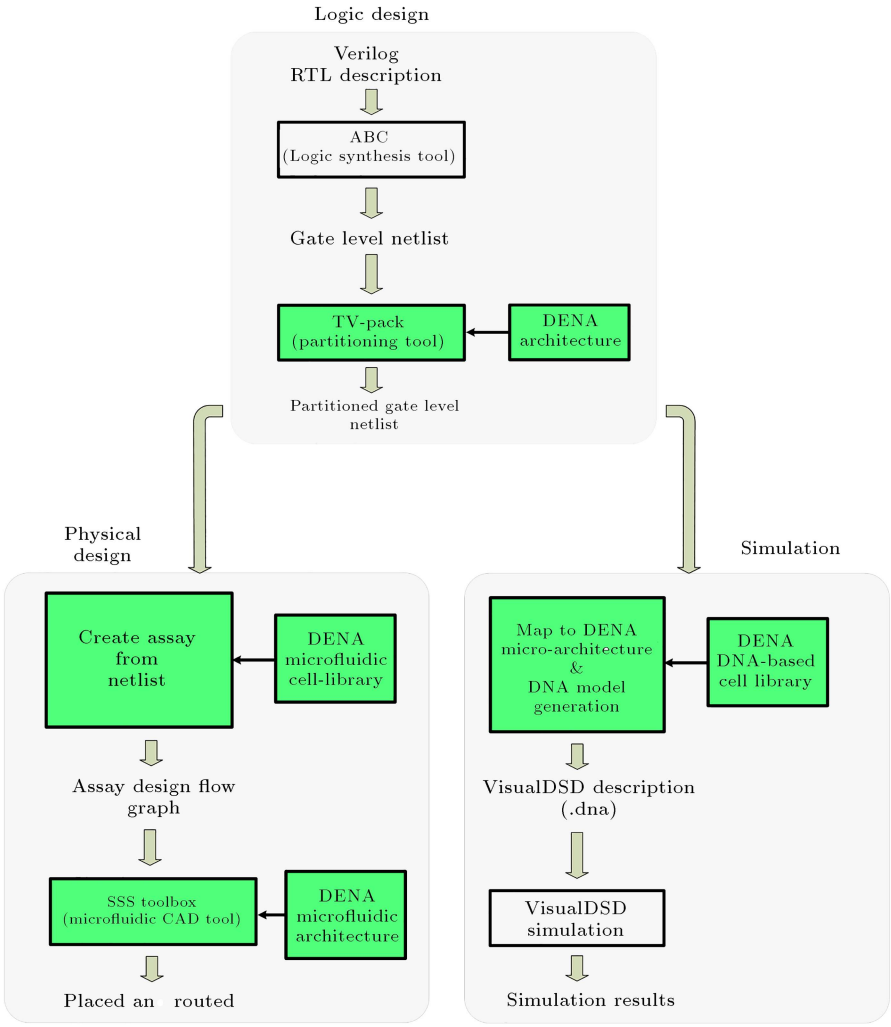


Figure 7. Overall process of the proposed automatic design flow.

of each cluster, inputs and output of each DC, and the connection between the DCs. Figure 8 represents the C17 circuit of ISCAS benchmark suite [37] in different design stages from RTL to DNA. Figure 8(a) and (b) show RTL and corresponding gate-level description, respectively. The partitioned and mapped circuit resulting in the revised TV-Pack is shown in Figure 8(c), which represents DC unique name and its index, configuration bitstream of the DC, and its I/O connections.

At the end of this phase, the partitioned netlist is ready for physical design and DNA-based functional simulation. This netlist can be used for not only simulation, but also the physical design of the synthesized circuit. The next subsections describe simulation and physical design process after the logic design.

5.2. Verification and simulation

Verification of a DNA-based logic system is done by analyzing the reactions between input and gate DNA strands. We used Visual-DSD toolbox for simulating the reactions between strands. In the first step, DSD file (input description file of VisualDSD) is generated for each synthesized DC. It is worth noting that DSD files describe different components of the DNA-based circuit that can be simulated with VisualDSD. In the second step, VisualDSD is used to simulate the DNA reactions between the DNA strands corresponding to the various parts of the synthesized circuit.

We proposed an algorithm to map the logical circuit to the DNA reactions. For this purpose, the synthesized netlist should be converted to a DSD description. The generated DSD file is a DNA-based implementation of the synthesized circuit, which can be simulated using VisualDSD. Figure 9 shows the presented algorithm for Visual-DSD code generation. The following paragraphs describe the proposed algorithm in detail:

Step 1. The description of unconfigured DLB structures is generated based on seesaw logic style (i.e., "DLB.DNA" file). In this file, each DLB is implemented in a DNA-based 4-input Seesaw lookup table;

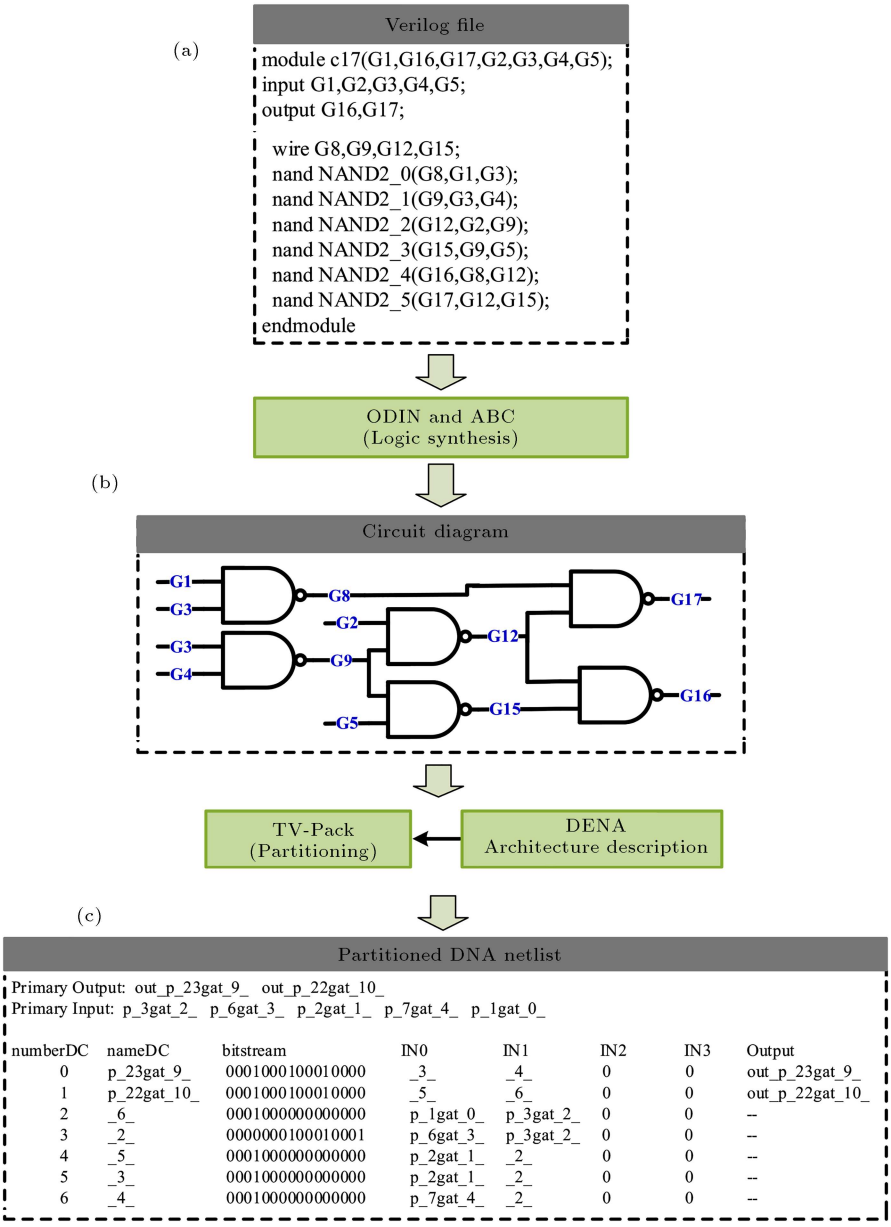


Figure 8. Design stages from RTL to DNA netlist for C17.

DSD code generation algorithm

WHILE end of netlist file LOOP
Step 1: DLB.DNA file is created.
Step 2: DLB is configured according to bitstream.
Step 3: NOT.DNA file is created.
Step 4: CONV.DNA file is created.
END LOOP

Figure 9. The proposed algorithm for DSD model generation from synthesized netlist.

Step 2. In this step, bitstream field of the final netlist is used to configure the lookup table. Minterms of DLB structure are configured by determining the concentration to the DNA strand corresponding to each minterm. At the end of this step, DLBs are configured by identifying the concentration of strands in the DSD description file;

Step 3. As mentioned before, each inverter includes two steps. Therefore, two different "NOT.DNA" files are created. Concentrations of input strands are determined according to circuit primary inputs at simulation time;

Step 4. "Conv.DNA" file is recreated in this step. The concentration level of the converter is equal to DLB output concentrations. Therefore, it will be determined by the simulation of DLB. As an illustrative example, generated files and the configuration of DC

0 for the IBM C17 benchmark [37] are shown in Figure 10. Due to the large size of files, only important parts of each file are shown. In DLB Configuration file (Figure 10(a), 3 minterms are ON based on the corresponding bitstream in Figure 8 (e.g., minterms 3, 7, and 11 in bitstream "000100010001000"). This figure represents the VisualDSD files for inverter, converter, and DLB (Figure 10(a) to (d)). Figure 10(e) shows the simplified circuit for the specified DLB.

DENA has a matrix-like structure such that each DC in this architecture is connected to its neighboring DCs. The DENA matrix cannot be simulated instantaneously because each DC can be configured and simulated if and only if all DC inputs be available. In this scenario, the concentration of the strands of each DC is configured and simulated based on the concentration level of its input strands. We generated a graph showing order of DCs in the simulation process. This graph called is DENA Cluster Priority Graph (DCPG) in this paper. Figure 11 shows the DCPG for C17 benchmark.

Figure 11 shows the proposed algorithm for generating the DCPG. This algorithm consists of two main steps. At the first phase, DCs are classified as primary nodes, internal nodes, and output nodes. Then, DCPG is created based on this classification at the second phase.

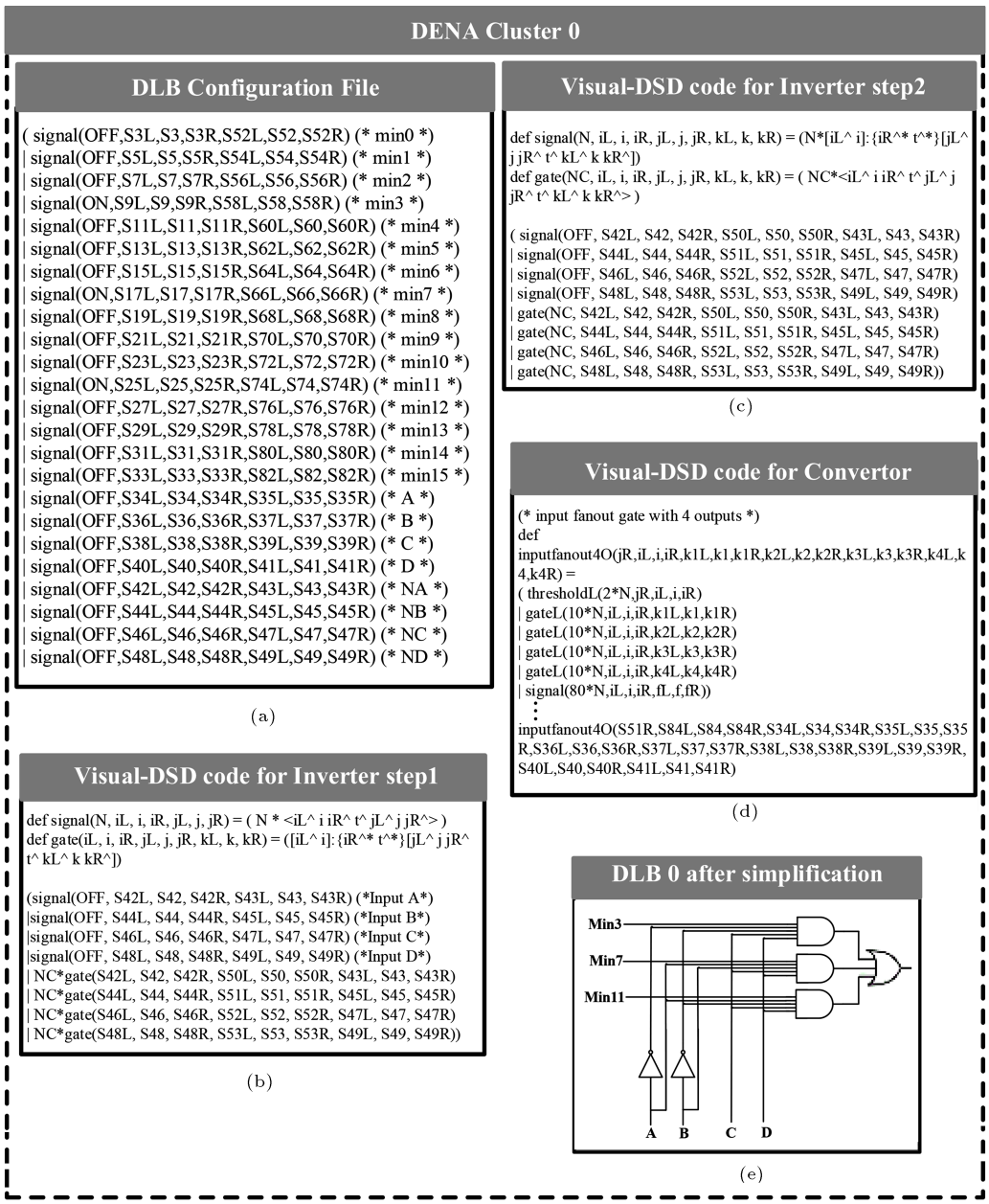


Figure 10. Generated files for DENA cluster 0 of IBM C17 benchmark.

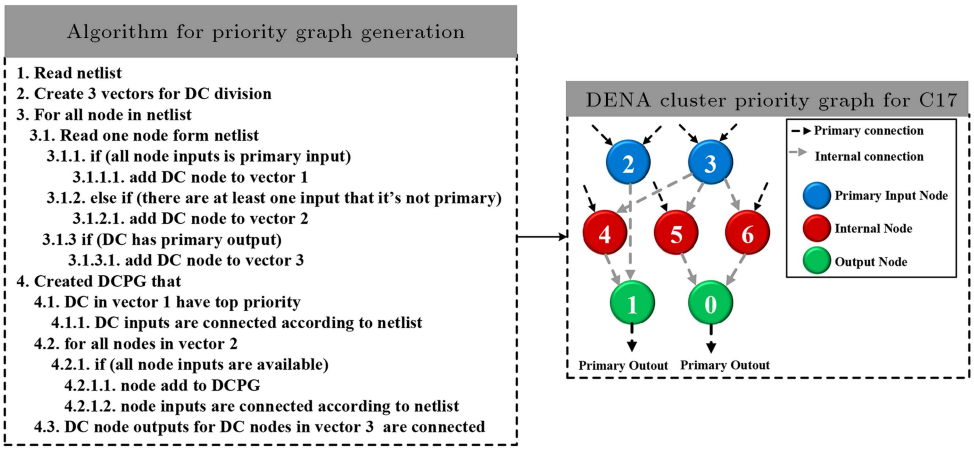


Figure 11. The proposed algorithm for generating the priority graph.

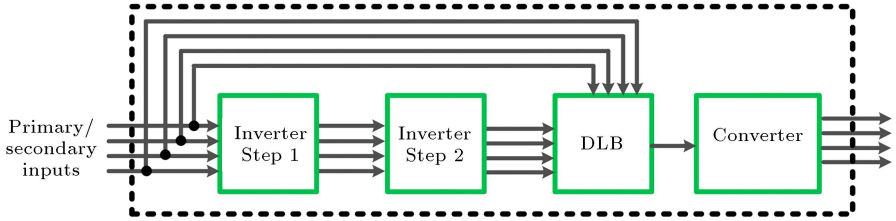


Figure 12. Simulation steps for each DENA cluster.

After creating the DCPG, primary input nodes should be processed first. Primary nodes are DCs with only primary input. They do not depend on any other strands. Therefore, they can be simulated parallel to the first step. All internal nodes must wait until their inputs have available. The output strands of the final nodes should be saved as primary output data. VisualDSD descriptive files simulate the whole circuit according to the DCPG. Now, each DC should be simulated based on the order specified by DCPG. In other words, an interactive simulation is scheduled such that the level ith DLBs are simulated and their output strands are fed to level $i+1$. Then, level $i+1$ is simulated.

As shown in Figure 12, DC inputs and their inverted signals are connected to DLB. DLB output is sent to the converter. Therefore, converter input concentration will be determined after the DLB simulation. The output of the converter can be sent to the next DC or primary output pins.

As mentioned before, a toolchain is developed to simulate the synthesized DNA-based logic systems. This toolchain converts the synthesized netlist into a sequence of DNA reactions that can be simulated using off-the-shelf DNA simulation tools.

5.3. Physical design flow

As mentioned before, a synthesized circuit is implemented on a digital microflu- idic biochip in the RTL2DNA flow. In this approach, the synthesized circuit is converted into the standard assay format that can be implemented on a DMFB platform. We used UCR SSS to imple- ment the circuit on DMFB [36]. SSS is an open-source framework designed for supporting algorithmic and software-driven control for DMFBs.

The physical design process of SSS framework consists of three main phases: scheduling, placement, and routing. The SSS performs the scheduling algorithm based on the input of bioassay protocol and DMFB architecture specifi- cations initially. This stage attempts to schedule microfluidic operations within the bioassay protocol given the available resources. Next, the placement algo- rithm executes and determines the location of scheduled microfluidic opera- tions on the DMFB array of electrodes.

Afterwards, the droplet routing algorithm is invoked to plan the moving pattern of droplets on the DMFB array of electrodes, either from input reservoirs to the modules, between the modules, or from modules to the output reservoir. The output of the design cycle is a sequence of electrode activations that execute the assay on the DMFB. In addition, the UCR SSS offers various Printed Circuit Board (PCB) wire-routing algorithms that aim to decrease the number of control pins and PCB layers required to fabricate the chip. The framework also includes a suite of visualization tools for debugging and producing graphical output from scheduling, placement, droplet routing, and wire-routing output files [36].

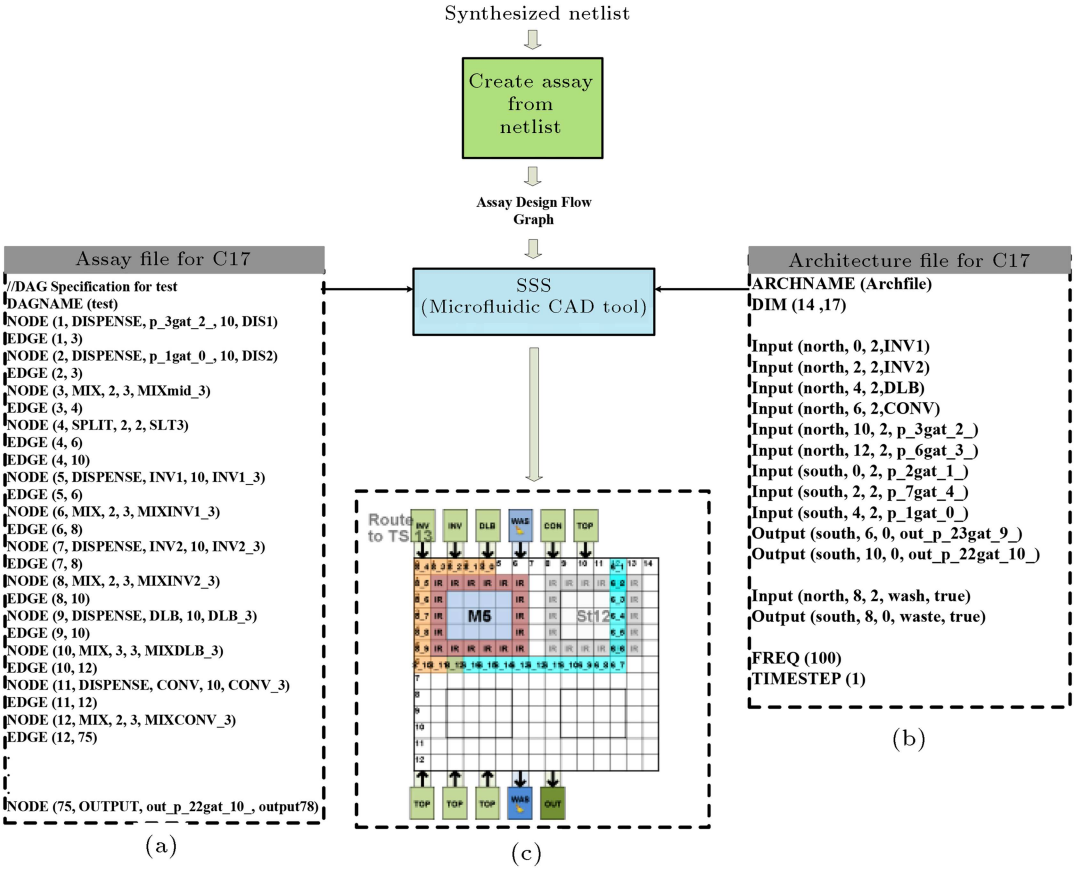


Figure 13. Generated architecture and assay file for SSS tool using RTL2DNA flow.

Figure 13 shows the inputs and outputs of the physical design flow for the C17 benchmark. These files are generated automatically in the proposed flow as an input of SSS tool. Figure 13(a) shows the generated assay from the synthesized circuit and Figure 13(b) specifies details of the architecture file corresponding to the target microfluidic platform for SSS tool. Assay file describes the DFG of the operation on the microfluidic platform. Nodes of DFG include four basic microfluidic operations (i.e., dispense, mix, split, and output). Architecture file determines the configuration of the microfluidic biochip. It specifies the input/output location, the time step of operation, drops frequency, location for the waste of useless strands, and washing for microfluidics.

The RTL2DNA flow enables the automatic design and verification of the large-scale DNA circuits over the microfluidic platform. This flow can help researchers to evaluate their ideas for future trends in DNA logic design.

6. Simulation results

This paper introduced an automatic design flow for design and verification of the large DNA circuits based on DMFBs. Table 1 illustrates the DENA cluster output concentration for C17 benchmark with 8 synthesized DCs for "10110" inputs. Each row of this table shows the information of a specified DC. Column Bitstream shows the synthesized configuration and bitstream/Boolean function of the DC. Columns IC1, IC2, IC3, and IC4 show the concentration of inputs, and column OC represents the concentration of output strand for the DC obtained through VisualDSD simulation tool. Finally, column LC shows the logical correctness of the DC output (Logic is correct if LC 'Y' (YES)). Of note, high-concentration ($\geq$ 800 nM) shows logic '1' and low-concentration ($\leq$ 100 nM) represent logic '0' in this table.

Table 1 shows that the logical output of the DC is correct. Moreover, the output concentration does not degrade upon increasing the logic circuit size and noise margin is acceptable for each logic circuit.

Figure 14 shows the details of output concentration for the C17 benchmark generated by Visula DSD. Green and blue curves show the output concentration for "00011" and "10100" inputs, respectively. These graphs prove the logical correctness of the synthesized circuit. Moreover, these curves show the noise margin of the output signals. It is worthwhile to note that DCs are simulated according to DCPG shown in Figure 11. Therefore, in the first step, DC2 and DC3

Table 1. Output logic and final output concentration of 8-bit binary adder.

DC ID	Bitstream (HEX)	IC1 (nM)	IC2 (nM)	IC3 (nM)	IC4 (nM)	OC (nM)	LC
2	1000 $\sum(12)$	100	900	100	100	75	Y
3	0111 $\sum(1,4,8)$	100	900	100	100	900	Y
4	1000 $\sum(12)$	900	75	100	100	76	Y
5	1000 $\sum(12)$	900	75	100	100	80	Y
6	1000 $\sum(12)$	900	75	100	100	75	Y
1	1110 $\sum(4,8,12)$	80	75	100	100	75	Y
0	1110 $\sum(4,8,12)$	900	100	100	100	900	Y

Table 2. Statistical information of attempted benchmarks.

Benchmark	Complexity	#IO	#DC	#Strand
C17	6	7	7	334
C432	160	43	182	334
C1908	880	58	294	334
C3540	1669	72	968	334
C6288	2406	64	1820	334

are simulated. The simulation results of DC3 are sent to DC4, DC5, and DC6 upon being simulated; DC1 is simulated by using DC2 and DC4 outputs; and DC0 is simulated using DC5 and DC6 outputs.

Table 2 shows the results of synthesis of some other IBM benchmarks [37] using RTL2DNA flow. In this table, column *Complexity* and *#IO* show the number of gates and IOs, respectively. Column *#DC* represents the number of DCs in the synthesized circuit and finally, column *#Strand* shows the required number of DNA strands in the implemented circuit.

As shown in Table 2, the number of required strands is constant for all the circuits in the proposed design flow because the number of strands inside each DC is independent of other DCs. This property is very critical to the feasibility of implementing real large circuits using DNA logic gates. Table 3 shows the physical implementation of IBM benchmarks and two full adders with 2 and 6 input bits on microfluidic biochip. The synthesized circuits are implemented on a Programmable Bio-Cell Matrix (PBCM) architecture [38] using SSS toolbox. In Table 3, column *Dimension* shows the number of rows and columns of the base biochip. Columns *#Electrode* and *#Pin* reprepresent the number of electrodes and number of pins, respectively, and finally, column *Total time* shows the total time of bioassay on the MFBC.

As can be seen in Table 3, physical properties of implementing benchmarks can be evaluated using RTL2DNA flow. Physical properties, in addition to

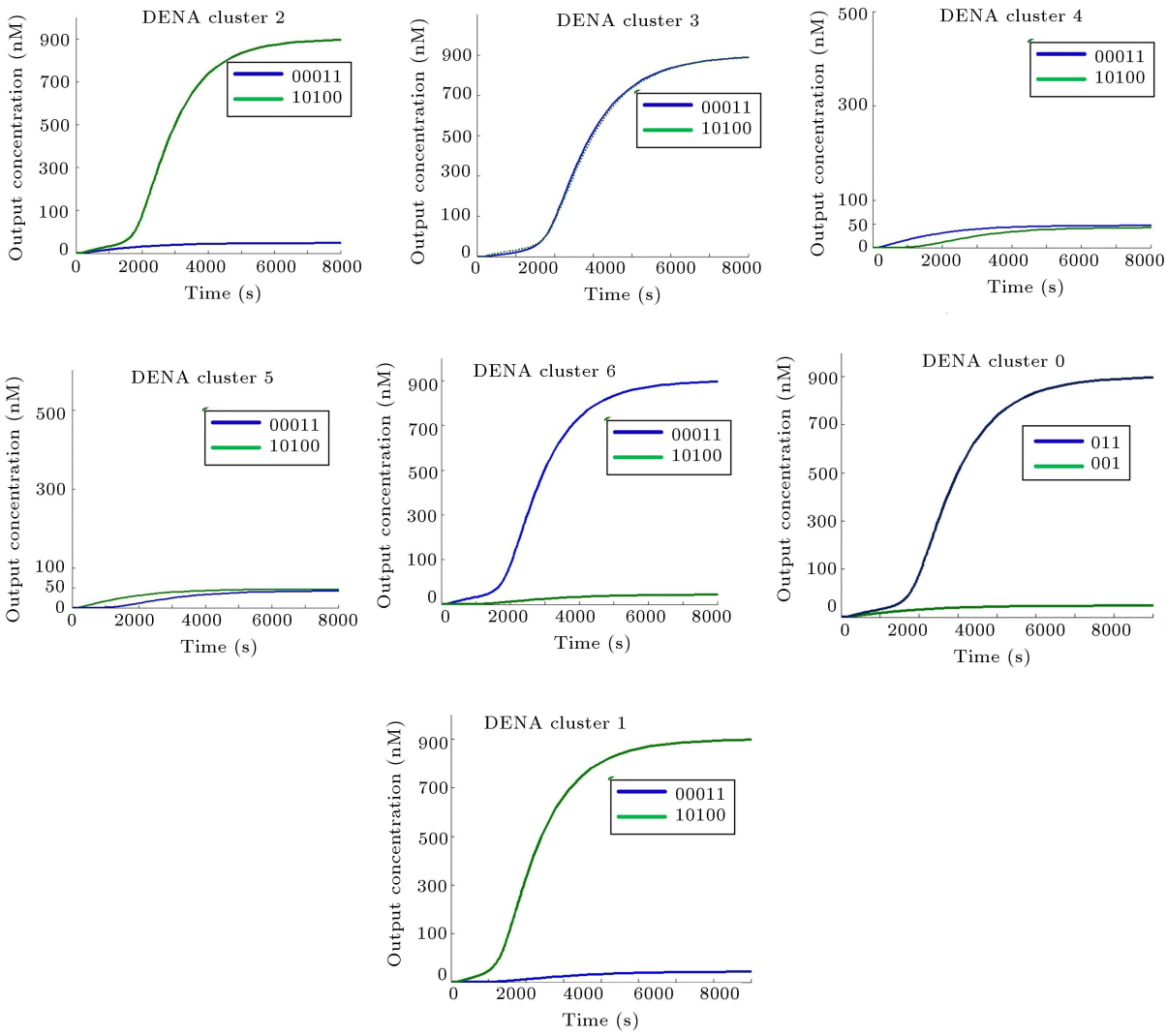


Figure 14. Concentration level of output signals of C17 benchmark obtained by VisualDSD.

Table 3. Physical properties of attempted benchmarks after realizing on the MFBC platform.

Benchmark	Dimension	#Pin	#Electrode	Total time (ns)
2-bit adder	12 × 21	22	63	21.33
6-bit adder	12 × 30	52	215	77.63
C17	12 × 25	32	116	66.19
C432	12 × 75	102	457	160.24
C1908	12 × 128	138	661	893.42

logical verification information, enable researchers to evaluate and compare various design choices quantitatively.

7. Conclusion

DNA computing is a fascinating multi-disciplinary technique that utilizes the highlighted features of DNA strands for logic design. A considerable number of methods for designing the DNA-based logic circuits have been addressed; however, no practical tool chain has been proposed for this purpose. In this paper, an automatic design flow and corresponding tool chain were proposed to facilitate the design and evaluation

of multi-stage large-scale DNA logic circuits on the microfluidic biochip platform. We used Verilog-to-Routing (VTR) tool for circuit partitioning and then, developed a tool chain to map the output of VTR to DNA strands. Placement and routing of DNA circuits on microfluidic platform were done using the Static Synthesis Simulator (SSS) tool. Results indicate the usability, simplicity, and flexibility of the proposed design flow.

An important contribution of the proposed technique is that the number of required strands is constant for all the circuits and does not increase for the large circuits. The proposed tool chain can open a horizon for design, implementation, and evaluation of logical DNA circuits on the microfluidic biochip.

References

1. Currin, A., Korovin, K., Ababi, M., et al. "Computing exponentially faster: implementing a non-deterministic universal Turing machine using DNA", In *Journal of the Royal Society Interface*, **14**(128) (2017).
2. Adleman, L. "Molecular computation of solutions to combinatorial problems", In *Science*, **266**(5187) (1994).
3. Sanches, C. and Soma, N. "A general resolution of intractable problems in polynomial time through DNA computing", In *Biosystems*, **150**(1) (2016).
4. Zhao, K., Wang, Z., Lu, Y., et al. "A new biologically DNA computational algorithm to solve the K-vertex cover problem", In *Journal of Computational and Theoretical Nanoscience*, **12**(3) (2015).
5. Seeman, N. "Nanomaterials based on DNA", In *Annual Review of Biochemistry*, **79**(1) (2010).
6. Sakamoto, K., Gouzu, H., Komiya, K., et al. "Molecular computation by DNA hairpin formation", In *Science*, **288**(5469) (2000).
7. Livstone, M. and Landweber, L. "Mathematical considerations in the design of microreactor-based DNA computers", In *International Workshop on DNA-Based Computters* (2004).
8. Qian, L. and Winfree, E. "Parallel and scalable computation and spatial dynamics with DNA-based chemical reaction networks on a surface", In *International Workshop on DNA-Based Computers* (2014).
9. Fan, D., Wang, K., Zhu, J., et al. "DNA-based visual majority logic gate with one-vote veto function", In *Chemical Science*, **6**(3) (2015).
10. Qian, L. and Winfree, E. "A simple DNA gate motif for synthesizing large-scale circuits", In *International Workshop on DNA-Based Computers* (2008).
11. Qian, L., Verilog-to-Routing Winfree, E., and Bruck, J. "Neural network computation with DNA strand displacement cascades", In *Nature*, **475**(7356) (2011).
12. Hemphill, J. and Deiters, A. "DNA computation in mammalian cells: microRNA logic operations", In *Journal of the American Chemical Society*, **135**(28) (2013).
13. Wu, L. and Xiaogang, Q. "Cancer biomarker detection: recent achievements and challenges", In *Chemical Society Reviews*, **44**(10) (2015).
14. Grissom, D. and Philip, B. "Fast online synthesis of generally programmable digital microfluidic biochips", In *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis* (2012).
15. Van, D. "A programmable molecular computer in microreactors", In *International Workshop on DNA-Based Computers* (2004).
16. Dwyer, C. "Computer-aided design for DNA self-assembly: process and applications", In *IEEE/ACM International Conference on Computer-Aided Design* (2005).
17. Rogers, W., Shih, W., and Manoharan, V. "Using DNA to program the self-assembly of colloidal nanoparticles and microparticles", In *Nature Reviews Materials*, **1**(3) (2016).
18. Botti, S., Rufoloni, A., Laurenzi, S., et al. "DNA self-assembly on graphene surface studied by SERS mapping", In *Carbon Journal*, **109**(1) (2016).
19. Selnihhin, D. and Ebbe Sloth, A. "Computer-aided design of DNA origami structures", In *Computational Methods in Synthetic Biology Humana Press*, **1989**(1) (2015).
20. Farhadtoosky, S. and Jahanian, A. "Customized placement algorithm of nanoscale DNA logic circuits", In *Journal of Circuits, Systems and Computers*, **26**(10) (2017).
21. Thubagere, A., Thachuk, C., Berleant, J., et al., "Compiler-aided systematic construction of large-scale DNA strand displacement circuits using unpurified components", In *Nature Communications*, **8**(1) (2017).
22. Beiki, Z. and Jahanian, A. "DENA: A configurable microarchitecture and design flow for biomedical DNA-based logic design", In *IEEE Transactions on Biomedical Circuits and Systems*, **11**(5) (2017).
23. Cannon, B.L., Kellis, D.L., Davis, P., et al. "Excitonic AND logic gates on DNA brick nanobreadboards", In *ACS Photonics*, **2**(3) (2015).
24. Amos, M., *DNA computation*, Doctoral Dissertation, University of Warwick (1997).
25. Yurke, B., Turberfield, A.J., Mills, A.P., et al. "A DNA-fuelled molecular machine made of DNA", In *Nature*, **406**(6796) (2000).
26. Zhang, D.Y. and Seelig, G. "Dynamic DNA nanotechnology using strand-displacement reactions", In *Nature Chemistry*, **3**(2) (2011).
27. Abdoli, A. and Jahanian, A. "Fault-tolerant architecture and CAD algorithm for field-programmable pin-constrained digital microfluidic biochips", In *CSI*

Symposium on Real-Time and Embedded Systems and Technologies (RTEST) (2015).

28. Wang, H., Liguo, Ch., and Lining, S. "Digital microfluidics: A promising technique for biochemical applications", In *Frontiers of Mechanical Engineering*, **12**(4) (2017).
29. Malic, L., Brassard, D., Veres, T., et al. "Integration and detection of biochemical assays in digital microfluidic LOC devices", In *Lab on a Chip*, **10**(4) (2010).
30. Christopher, B. and Landers, J.P. "Electrode materials in microfluidic systems for the processing and separation of DNA: A mini review", In *Micromachines*, **8**(3) (2017).
31. Hua, Z., Rouse, J.L., Eckhardt, A.E., et al. "Multiplexed real-time polymerase chain re-action on a digital microfluidic platform", In *Analytical Chemistry*, **82**(6) (2010).
32. Chang, Y.H., Lee, G.B., Huang, F.C., et al. "Integrated polymerase chain reaction chips utilizing digital microfluidics", In *Biomedical Microdevices*, **8**(3) (2006).
33. Qian, L. and Winfree, E. "Scaling up digital circuit computation with DNA strand displacement cascades", In *Science*, **332**(6034) (2011).
34. Rose, J., Luu, J., Yu, C.W., et al. "The VTR project: Architecture and CAD for FPGAs from verilog to routing", In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays* (2012).
35. Lakin, M.R., Youssef, S., Polo, F., et al. "Visual DSD: A design and analysis tool for DNA strand displacement systems", In *Bioinformatics*, **27**(22).
36. Grissom, D., Curtis, C., Windh, S., et al. "An open-source compiler and PCB synthesis tool for digital microfluidic biochips", *Integration the VLSI Journal*, **51**(1) (2015).
37. IBM Digital Analytics Benchmark. Available on https://www-01.ibm.com/software/marketing-solutions/benchmark (2017).
38. Taajobian, M. and Jahanian, A. "Higher flexibililty of recon gurable digital Mi-cro/Nano fluidic biochips using an FPGA-inspired architecture", In *Scientia Iranica*, **23**(3) (2016).

Biographies

Zohre Beiki received the BSc degree in Computer Engineering from Islamic Azad University of Mashhad and the MSc degrees in Computer Engineering from University of Science & Technology, Tehran, Iran in 2008 and 2011, respectively. She is currently an Assistant Professor at the Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran and her research interests are DNA computing and embedded system design.

Ali Jahanian received the BSc degree in Computer Engineering from University of Tehran, Tehran, Iran in 1996 and the MSc and PhD degrees in Computer Engineering at Amirkabir University of Technology, Tehran, Iran in 1998 and 2008, respectively. He joined Shahid Beheshti University, Tehran, Iran in 2008. His current research interests are focused on VLSI design automation and DNA computing.