SEMINAR REPORT
Seminar Embedded Systems SS25

Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

Molecular Computing

Thorsten Waldau
University of Kaiserslautern-Landau, Department of Computer Science

Note: *This report is a compilation of publications related to some topic as a result of a student seminar. It does not claim to introduce original work and all sources should be properly cited.*

Molecular Computing is a way to perform computations that does not rely on classical silicon-based logic gates but rather on chemical reactions. This report takes a look at the basics of molecular computing, showcasing a few basic circuits and finally ending by taking a look at different ways to perform matrix multiplications using molecular computing for both boolean and numeric matrices.

1 Introduction

Although classical Computing has made many improvements, some applications have been found, for which the usage of digital logic proves to be inferior to the application of other computing concepts such as quantum computing or, in some other applications, molecular computing [2]. It does not have to compete with traditional computing in these applications as Molecular computing provides a method of computing that can use chemical processes to power its computations. It has also seen remarkable progress in the last years, with progress accelerating faster than Moore's law, with this growth being referred to as Carlson's law [8]. Although the computation rates are 10 to 15 orders of magnitude slower than traditional computing, it is sufficient for the applications in which it is used, for example targeted drug delivery or monitoring of rates of change of concentrations [8]. Furthermore due to the chemical nature of the process, massive parallelization is possible [8]. This report is structured as follows. In section 2 an introduction into molecular computing is given. In section 3 some basic circuits, implemented by molecular computing, are shown and explained, including a Digital to Analog Converter (DAC) and an Analog to Digital Converter (ADC). Section 4 is used to show implementations of matrix multiplications, some of which are for boolean matrices, and others being for numerical matrices. Section5 discusses some related works and finally Section6 compares and discusses the different implementations of matrix multiplications and summarizes the results.

2 Molecular Computing

Digital computing mainly uses boolean logic. Therefore, to implement molecular computing, the different logic gates, that perform the basic operations of boolean logic have to be implemented. These gates are mostly shown in Fig. 1, only missing the NOT gate, which takes one input and returns the negated input, e.g. if we were to input a 0 into a NOT gate, the gate returns a 1. The first of the gates shown is the AND gate, taking two inputs and returning a 1 iff both inputs

are 1 as well. The second gate is the OR gate returning 1 iff at least one of the two inputs is 1. The logic for the other gates is similar, following the boolean formulas given under the gates in Fig. 1. The last notable gate shown is the NOR gate. Using only this gate it is possible to create all other shown gates, making it functionally complete. Using molecules to compute allow for information handling at the nano scale, especially solving the issue of connectivity, e.g. by using one module to catch the input, triggering a reaction to give the output [10]. In the first subsection some simple logic gates are introduced. The second subsection then shows logic gates, which can be reprogrammed, a necessity for computing. Lastly, the last subsection provides a way to work with numbers in molecular computing, another necessity to be able to use it to compute things.

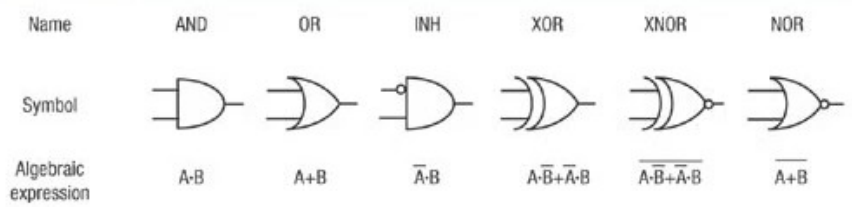

Figure 1: The different logic gates, taken from [10]

2.1 Molecular Logic

One way to create molecular logic gates is to use fluorescence of molecules. An example of this would be Fig. 2. This figure shows a logic gate based on photoinduced electron transfer (PET), having an aromatic anthracenyl ring system on one side, also known as fluorophore, shown here in blue. This molecule would normally fluoresce blue when exposed to ultraviolet light. However, this process is prevented in this case, as an electron is transferred to the anthracenyl group from the Nitrogen or the Oxygen atoms in the molecule (see Fig. 2 a). Should one of the receptors be occupied by either a hydrogen ion (N) or a sodium cation (O), then PET still happens, If on the other hand, both receptors are occupied by their respective partner, then PET is prevented and fluorescence can be observed [10]. This makes this molecule an AND gate, the output being '1' if PET has been prevented. This basic structure, namely fluorophore-spacer-receptor-spacer-receptor is a common structure. The fluorophore and the receptors can both be tuned to different guest species (in this case being the H^+ and the Na^+ ions) [10]. Light-absorbance should also not be ignored, as it is a simple effect. An example for this is seen in Fig. 2, showcasing a receptor-chromophore-receptor involving internal charge transfer (ICT) excited states [10]. The chromophore is shown here in blue while the receptors are shown in green. In this molecule capturing a guest species by one of the receptors causes a red-shift in the UV-Vis absorption spectrum, and the other receptor/guest pair causes a blue-shift instead. Should both guests be captured by their respective receptors then the absorption spectrum remains unchanged. This therefore showcases a XNOR gate with absorbance output. Considering the transmittance as output instead, this gate transforms into an XOR gate. Various different substrates, reactions, reagents and readout modes have already been considered. This can be seen in Fig. Molecular

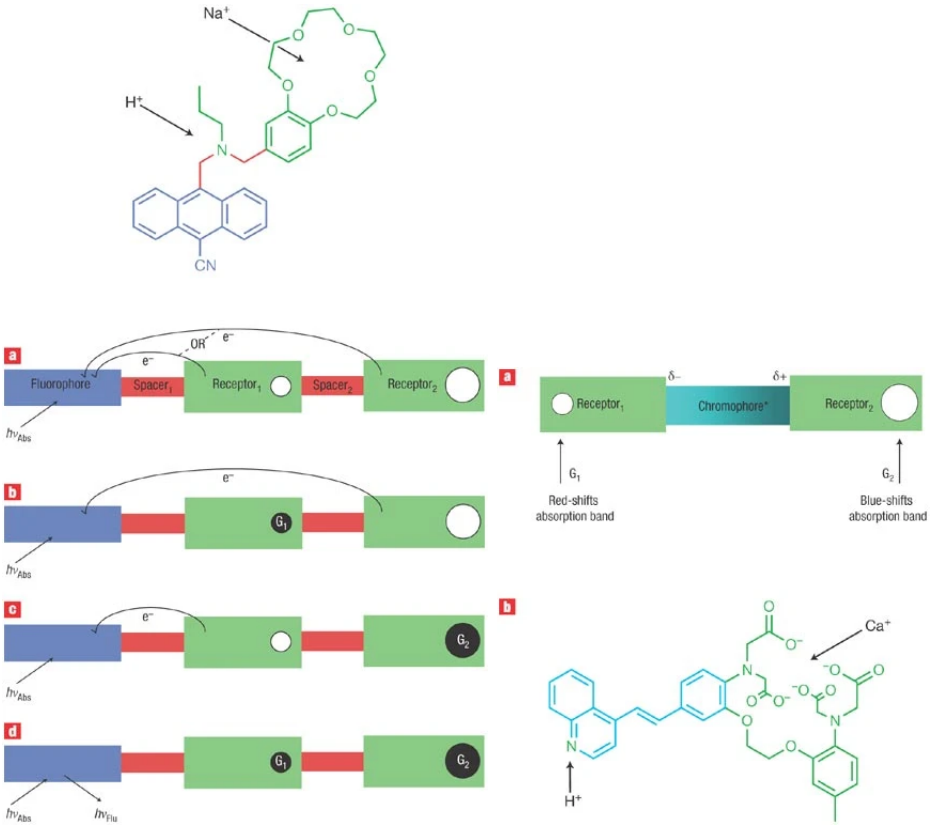

Figure 2: Molecular logic gate based on left: PET, right: ICT taken from [10],

logic is not without its share of problems. One of these issues is addressing, as in semiconductor based systems the signals flow along wires between the different gates, while in molecular based systems most are addressed differently, although there are some molecular counterparts that are starting to be addressed this way. A simple solution for this in molecular computing is the usage of diffusion in a solution [10]. The output of the gates is fluorescence, which is picked up by sensors, further discussed in 3. Another issue is stability and durability of the system. This is a problem as organical molecules decay in contrast to inorganic molecules, that do not. This can be reduced by e.g. reducing the intensity of the exciting light and increasing the efficiency of the detector detecting the emissions. Most of semiconductor-based computing hardware is based on a hard-wired logic circuit. Changing the configuration of the circuit on command is a more recent innovation. This can also be done by molecular logic. In fact, this can be done conveniently as many types of chemicals and many colors of light, with all being distinguishable from one another, are used as input/outputs [10]. PET and ICT structures can also be used for this.

2.2 Number Handling

Number handling is a very important part of computing, that all parts of computing have to address. This is mostly done by using a half-adder (see Fig. 3). This was first done in 2000 [10] by running a 2-input AND gate and a 2-input XOR gate in parallel. The gates used are shown in Fig. 4. The AND gate is a typical PET AND gate, using H^+ and Na^+ ions. The XOR gate on the other hand is based on a Sodium salt of $[Fe(CN)_5NO]^{2-}$ with a thiol of the structure RSH, in which R is a general organic group. The salt is colorless, until it reacts with the thiol, after which it produces the purple $[Fe(CN)_5N(O)SR]^{3-}$ which absorbs light with a wavelength of 520nm strongly [10]. If both cells return a low absorbance signal then the output is '0', else '1'. Older attempts were based on DNA molecules [10]. The half-subtractor (see Fig. 3) is a similar gate that instead of outputting a sum and a carry bit outputs a difference and a borrow bit. Integrating AND, XOR, INH and OR logic functions into one molecular entity using reconfigurable molecular logic, that switches its logical behavior at different wavelengths, it is possible to create a calculator that can add or substract different numbers.

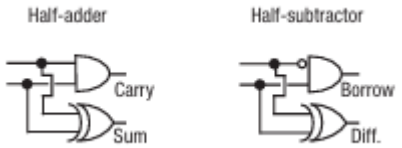

Figure 3: The circuit diagrams of a half-adder and a half-substractor. Taken from [10]

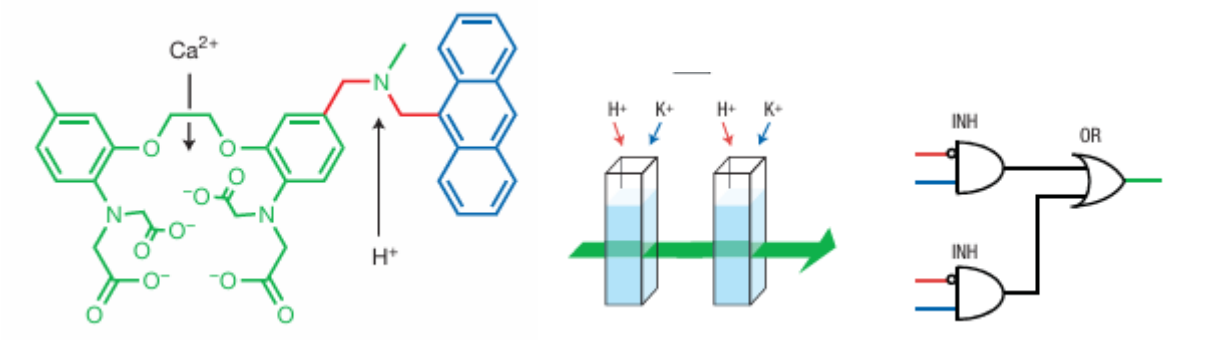

Figure 4: left: the molecule used as an AND gate, right: the used XOR gate, taken from [10]

2.3 DNA Strand Replacement

Using DNA strands as the molecules it is possible to implement reactions by strand displacement [8]. It is therefore one of the methods used to implement molecular computing. Fig. 5 shows

the implementation of this mechanism on the basis of the following reaction.

$$A + B \underset{r}{\overset{f}{\rightleftharpoons}} C + D \tag{1}$$

A, B, C and D are all DNA strands First, the toehold 1 of strand A starts binding with the toehold 1* of strand B. This is followed by a branch migration, with domain 2 of strand A replacing the domain 2 of strand 2-3. Lastly domains 3 and 3* are separated and two new strands C and D are produced. [8]

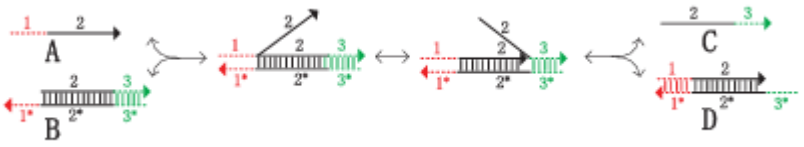

Figure 5: Implementation of the DNA strand-replacement mechanism, taken from [8]

3 Molecular Sensing

The Problem of signal transmission between the output of one molecular logic gate and the input of a second one is one of the bigger problems of molecular computing, as alluded to in section 2.1. The simplest solution to this problem are sensors, being taken from cell biology [10]. One example of this is the sensor fura-2, introduced in [4], monitoring changes of Ca^{+} in a millisecond scale in a micrometer space. It is able to be addressed optically and operates at a molecular level. The optical addressing functions as follows: the solution is put under a microscope and excited with ultra-violet light at a wavelength of 335 nm. While this is done, the intensity of the fluorescence in a specified area is measured. Concentrations near membranes can be measured in a similar matter by using hydrophobic effects to position molecular devices near these membranes [10]. There have also been successes with PET based sensors monitoring, for example, cellular pH, lead and mercury. These reactions are irreversible, making them unsuitable for some computational efforts, as they are not able to be reset. In other cases, where the target has no known receptor the formation of chemical bonds rather than reversible chemical reactions is used to detect these agents [10]. Another solution are delay elements, used to delay the computations to separate the inputs from the outputs.

3.1 Molecular Continuous-Time Systems

Molecular continuous-time systems are important as most chemical reactions are analog. Analog computations can therefore be efficiently implemented by an chemical reaction network (CRN). A molecular adder functions as follows: we take two input concentrations that have to be added. These are transferred to the same molecular type by two reactions [8]. These reactions take the initial inputs and perform independent reactions on these to transfer the input molecules into a

different output molecule. A similar unit would be the molecular multiplication unit. This unit can simply be implemented by two reactions:

$$x + y \xrightarrow{k_1} x + y + z \tag{2}$$

$$z \xrightarrow{k_2} \emptyset \tag{3}$$

In this reaction x, y and z are the concentrations of a corresponding molecular type with the k_i being the rate constants of the reactions. Using the mass action kinetics model, which says that for the reaction

$$S + E \xrightarrow{k} P + E \tag{4}$$

where S and E are reactants, P and E are products and k being the rate constant of this reaction, we obtain the following differential equations for the concentrations:

$$\frac{dP}{dt} = -\frac{dS}{dt} = kSE \tag{5}$$

$$\frac{dE}{dt} = 0 \tag{6}$$

Meaning, that the speed of the reaction is proportional to the input concentrations and the rate constant. Using the mass-action kinetic model, we obtain these differential equation:

$$\frac{dz}{dt} = k_1 xy - k_2 z \tag{7}$$

where x, y, and z are the molecular concentrations of the respective molecules [8]. In the steady state

$$\frac{dz}{dt} = 0 \tag{8}$$

we obtain that

$$z = \frac{k_1}{k_2} xy \tag{9}$$

Implementations of more complex functions are shown in [3].Any linear continuous-time system can be implemented by using integration, gain and summation [8]. All of the signals u are represented by differences in concentrations between two molecular types, u^+ and u^-, with these being defined as

$$u^+ = \begin{cases} u & if\ u > 0 \\ 0 & otherwise \end{cases} \tag{10}$$

$$u^- = \begin{cases} |u| & if\ u < 0 \\ 0 & otherwise \end{cases} \tag{11}$$

Integration can be implemented by the following reaction,showing the function $y(t) = \int_0^t u(\tau)d\tau + y(0)$:

$$u^{\pm} \xrightarrow{\alpha} u^{\pm} + y^{\pm} \tag{12}$$

Gain and Summation can be implemented by the reactions, where u_i represent the different inputs

$$u_i^{\pm} \xrightarrow{\gamma k_i} u_i^{\pm} + y^{\pm} \tag{13}$$

$$y^{\pm} \xrightarrow{\gamma} \emptyset \tag{14}$$

The function implemented by this is for $n = 1$ the gain block

$$y = k_1 u_1 \tag{15}$$

if instead $n \geq 2$ then the function represented is

$$y = \sum_{i=1}^{n} k_i u_i \tag{16}$$

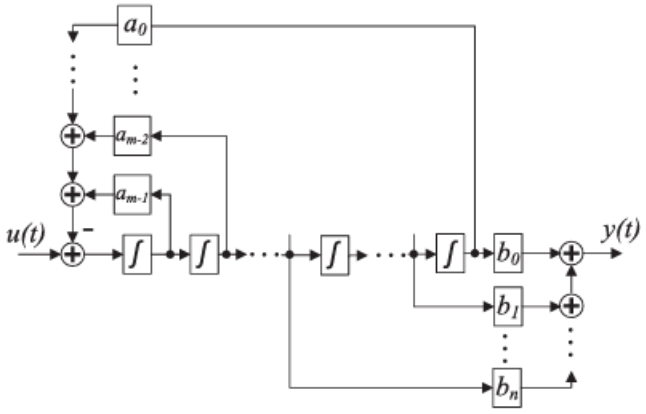

Figure 6: Constructing a linear I/O system, using gain, summation and integration blocks, taken from [8]

Fig. 6 shows the construction of a linear I/O system based on the transfer function

$$\frac{Y(s)}{U(s)} = \frac{B(s)}{A(s)} \tag{17}$$

where $Y(s)$ and $U(s)$ are the laplace transforms of the input and the output, $B(s) = b_n s^n + b_{n-1}s^{n-1} + ... + b_1 s + b_0$ and $A(s) = s^m + a_{m-1}s^{m-1} + ... + a_1 s + a_0$ and $m \geq n$ [8].

An additional unit would be a first-order low-pass continuous-time filter, implementing the transfer function $\frac{1}{s+a_0}$. The reactions of this unit are shown in Fig. 7, while a diagram of this unit is shown in Fig. 8.

$$x(t) = u(t) - a_0 y(t) \rightarrow \begin{cases} y^{\pm} \xrightarrow{\gamma a_0} y^{\pm} + x^{\mp} \\ u^{\pm} \xrightarrow{\gamma} u^{\pm} + x^{\pm} \\ x^{\pm} \xrightarrow{\gamma} \varnothing \\ x^{+} + x^{-} \xrightarrow{\eta} \varnothing \\ u^{+} + u^{-} \xrightarrow{\eta} \varnothing \end{cases}$$

Figure 7: The reactions for a first-order low-pass continuous-time filter, taken from [8]

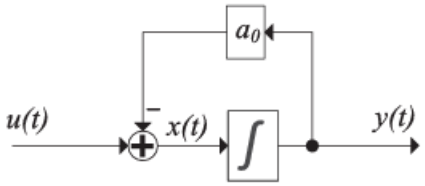

Figure 8: A first-order low-pass continuous-time filter, taken from [8]

3.2 Molecular Discrete-Time Systems

In discrete-time systems, the data has to be sampled at specific points in time [8], and all transfers of signals have to be synchronized. These synchronous transfers are called computational cycles [8]. Taking these things into account allows for 3 different synchronization schemes. Fully synchronous systems are synchronized by a two-phase clock. For this clock reactions are chosen that produce sustained oscillations [8]. Some examples of reactions that produce oscillations are Lotka-Volterra, Brusselator and Arsenite-Iodate-Chlorite systems, although these are not quite suitable for a synchronous clock, as a clock needs abrupt transitions and symmetrical signals between its states, and these systems do not [8]. Instead, multi-phase chemical oscillators are used. An example for a multi-phase chemical oscillator is the 4-phase oscillator, using four types of molecules: R, G, B and V. The reactions implementing this oscillator are shown in Fig. 9.

r, g, b and v are indicators for the absence of R, G, B and V, and are slowly continuously produced. The types R, G, B and V consume their partners quickly. The reactions on the right transfer the types from one type into another, changing the phase of the oscillator. This only happens, if the previous type is fully consumed. The reactions in the middle are then used to introduce positive feedback [8]. As the different clock phases should not overlap, two nonadjacent phases are chosen to be used as the clock phases [8]. Fig. 10 shows the different concentrations of R and B, obtained through differential equation simulation of the reactions in Fig. 9.

A globally-synchronous locally-asynchronous system uses a clock with 3 phases, in comparison to the fully synchronous system. This clock consists of 3 proteins, named R, G and B, with the clock cycle consisting of the transfer of R to G, G to B and B to R. This provides

$$
\begin{aligned}
2S_r &\xrightarrow{k_{slow}} r + 2S_r & 2G &\underset{k_{fast}}{\overset{k_{slow}}{\rightleftharpoons}} I_G & R + v &\xrightarrow{k_{slow}} G \\
2S_g &\xrightarrow{k_{slow}} g + 2S_g & R + I_G &\xrightarrow{k_{slow}} 3G & G + r &\xrightarrow{k_{slow}} B \\
2S_b &\xrightarrow{k_{slow}} b + 2S_b & 2B &\underset{k_{fast}}{\overset{k_{slow}}{\rightleftharpoons}} I_B & B + g &\xrightarrow{k_{slow}} V \\
2S_v &\xrightarrow{k_{slow}} v + 2S_v & G + I_B &\xrightarrow{k_{slow}} 3B & V + b &\xrightarrow{k_{slow}} R \\
R + r &\xrightarrow{k_{fast}} R & 2V &\underset{k_{fast}}{\overset{k_{slow}}{\rightleftharpoons}} I_V & & \\
G + g &\xrightarrow{k_{fast}} G & B + I_V &\xrightarrow{k_{slow}} 3V & & \\
B + b &\xrightarrow{k_{fast}} B & 2R &\underset{k_{fast}}{\overset{k_{slow}}{\rightleftharpoons}} I_R & & \\
V + v &\xrightarrow{k_{fast}} V. & V + I_R &\xrightarrow{k_{slow}} 3R & &
\end{aligned}
$$

Figure 9: The reactions implementing a 4-phase oscillator, taken from [8]

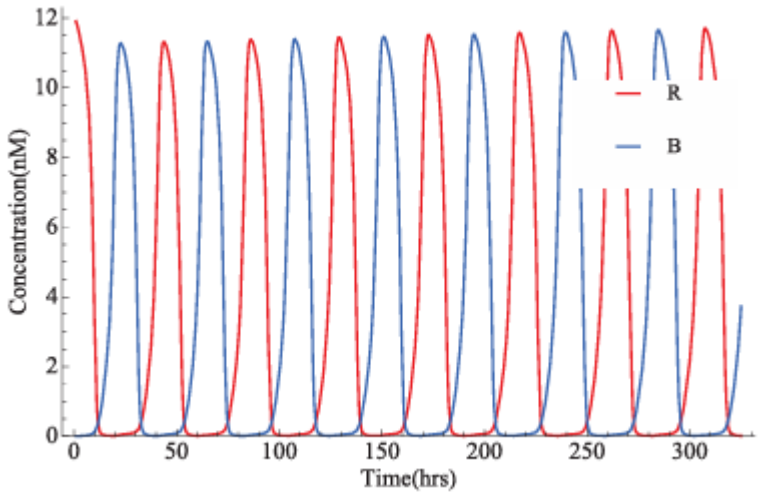

Figure 10: Simulation results for phases R and B of a four-phase oscillator, taken from [8]

global synchronization [8]. The system therefore does not contain a clock signal, instead the computations are "self-timed", starting the computations, when an external cause removes all of one of the proteins and supplies a quantity of the next protein [8]. This is shown in Fig. 11.

In a fully asynchronous system signal transfers and computations start from the input into the system and progresses its output in different phases, with each delay element used, consisting of two molecules, introducing two phases. Reactions for a phase are begun, as soon as the previous phase has been completed.

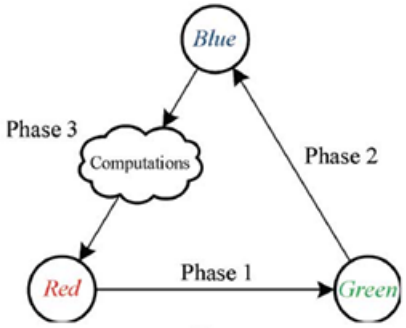

Figure 11: The cycle in a globally-synchronous locally-asynchronous system, taken from [8]

4 Matrix Multiplication

This section shows different methods to perform matrix multiplications, as a practical example of the possible applications of molecular computing. These are then compared in Section 6. The first two methods, shown in subsection 4.1 and in subsection 4.2 are used in the multiplication of boolean matrices, while the other three methods, shown in subsection 4.3, 4.4 and 4.5 are used for the multiplication of a matrix of integers. The first two methods are both based on DNA, while the other three methods use compartments and channels to connect the compartments to perform the multiplication. All of the methods are used in a lab-scale setting, with their output being visualized.

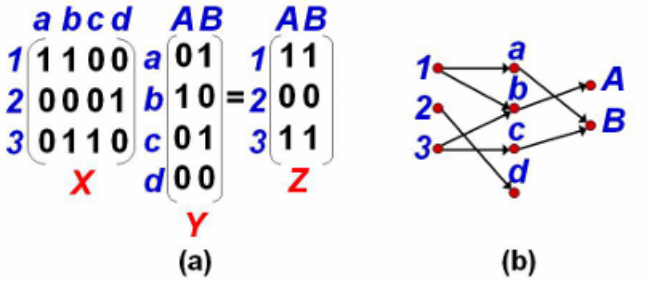

Figure 12: An example of a boolean matrix multiplication, (a) showing the matrices and the product, and (b) showing the directed graph equivalent of this multiplication, taken from [7]

For a 2x2 matrix there are 2 compartments used as inlets, connected to each inlet are two intermediates. Two outlets are each connected to one intermediate of both inlets [2]. This basic structure can be seen in Fig. 13.

This process uses two different molecules, named A and B, which are shown in blue and green respectively in Fig. 13. The molecules react according to the following reaction, molecule C is

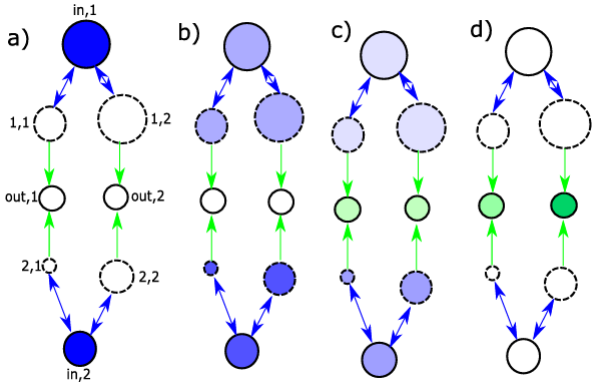

Figure 13: Sketch of the computation process, taken from [2]

a reaction partner for A.

$$A + (C) \longrightarrow B \tag{18}$$

In the intermediates molecules of A are transformed into molecules of B, after an initial concentration of A molecules has been added to the inlets. The connectors between inlet and intermediate allow only molecules of A and the connectors between the intermediates and the outlets only allow molecules of type B [2]. Using both

$$C_{in} = [C^A_{inlet1}, C^A_{inlet2}]^T \tag{19}$$

and

$$C_{out} = [C^B_{output1}, C^B_{output2}]^T \tag{20}$$

the relationship between the two can be calculated by

$$C_{out} = MC_{in} \tag{21}$$

The values in this matrix can be chosen arbitrarily [2]. As the final result of the matrix multiplication we obtain

$$C^B_{outputj} = \sum_{i=1}^{2} M_{i,j} C^A_{inleti} \tag{22}$$

As these methods were used in a lab-scale setting, the transfer between the compartments was done using manual transport, instead of diffusion as it would have at the micro/nano scale. This process can also be generalized to all matrices [2].

4.1 Hybridization-ligation

The Hybridization-ligation method was first introduced in an experiment by Adleman [7]. This method works by first designing the DNA sequences necessary to perform the multiplication. These strands are then used to generate the initial pool by performing a hybridization reaction, followed by a ligation reaction. This pool is then subjected to a polymerase chain reaction, which is an amplification technique widely used in molecular biology, to copy the resulting DNA strands. Finally the results are subjected to a gel electrophoris to visualize the output [7]. The initial DNA strands are subjected to kination in a thermal cycle followed by a heating to a temperature sufficient to split double strands but not high enough to damage the strands, after being put into a solution [7]. Allowing this solution to cool down gradually allows for hybridization, creating strands consisting of initial vertex strands, intermediate vertex strands, terminal vertex strands and directed edges, creating "paths". Finally incomplete strands are completed with hydrogen bonds by ligation [7]. This process is visualized in Fig. 14.

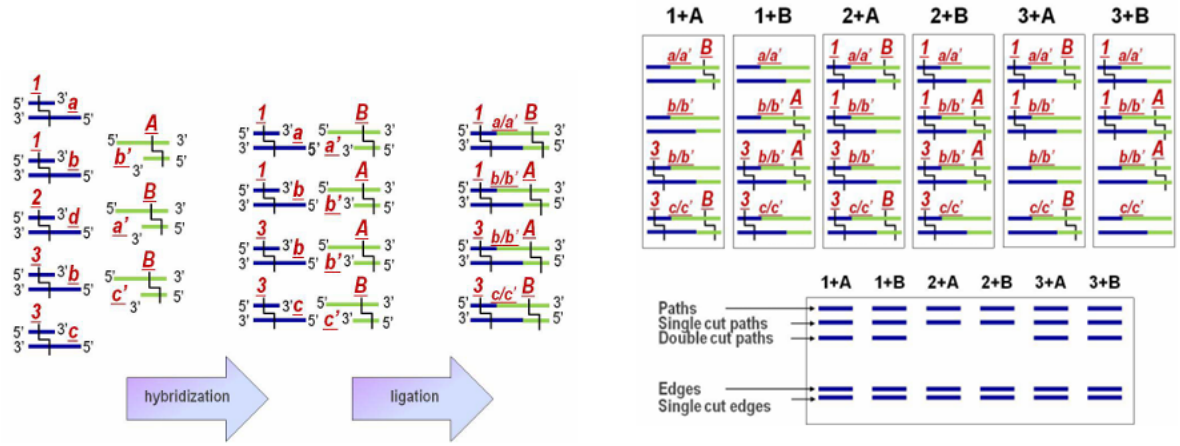

Figure 14: Left: Example of the hybridization-ligation method, right: Read-out for the hybridization-ligation method taken from [7]

On the left of this figure are the initial DNA strands. During hybridization the pieces will anneal to other pieces complementary to them [7]. This allows only sequences with the same intermediate vertex to anneal to each other. These are then ligated to each other, forming duplex DNA representing the path. The existence or non-existence of a path is represented by the "cutting" reaction, being done by a combination of restriction enzymes defined for initial and terminal vertices [7]. The PCR process is then used to amplify these cut strands for effective detection in read-out [7]. A path that contains only one of the initial or terminal vertices will be shortened on only one end, on the other hand a strand containing a valid path will be shortened on both ends [7]. In the case of both ends being cut, the value denoted in the matrix product is "1", otherwise it is "0" [7]. This read-out for hybridization-ligation, taken from the example in Fig. 12 is visualized in Fig. 14

4.2 Parallel-overlap Assembly

Stemmer introduced the parallel-overlap assembly (POA) method to facilitate in vitro mutagenesis and was successfully applied by Kaplan et al. for initial pool generation to solve a maximal clique problem [7]. The steps to compute the matrix using POA is similar to the steps used with hybridization-litigation. Although this method is similar to PCR with denaturation, annealing and extension happening in cycles, its characteristics are different. These differences are firstly that POA does not use a primer as compared to PCR and secondly that the number of DNA strands decreases each cycle instead of doubling [7]. It uses simple single stranded DNA oligonucleotides instead of the double stranded DNA strands of the hybridization-ligation method as representations of the vertices. The edges are represented simply as a concatenation of vertices strands [7]. They are designed as connector strands with partial complementary sequences for a portion of the vertices sequences [7]. Fig. 15 shows this process by showing both the start and the result of POA.

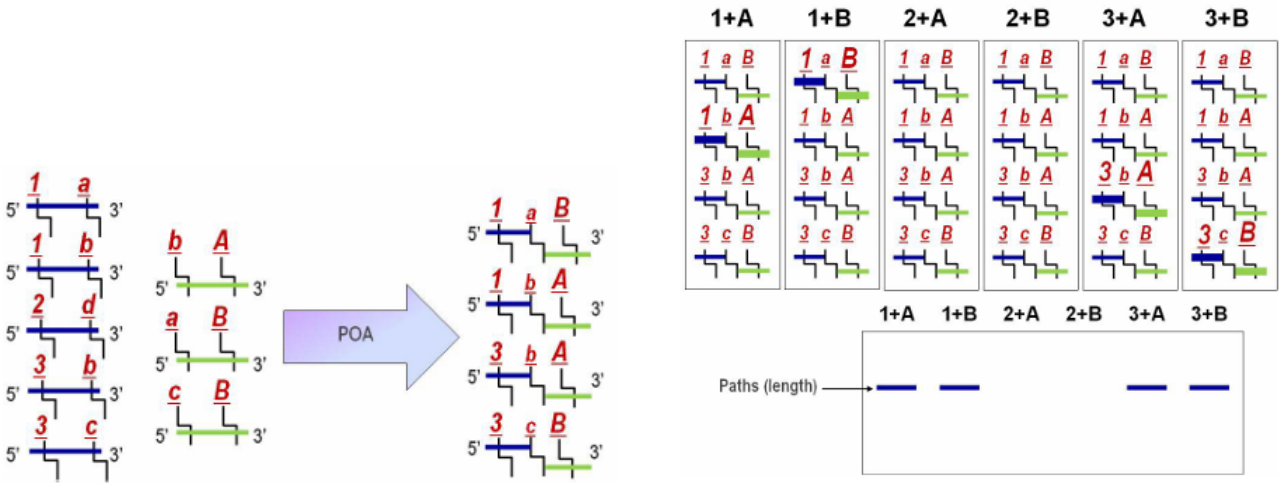

Figure 15: The process of POA on the left, and the read-out of POA on the right, taken from [7]

As POA does not require primers, it needs another way to represent which vertices are initial vertices and which are terminal vertices. This is done via the primers used for the PCR step which amplifies the resulting DNA strands. In this case the primers are only used on strands containing both an initial and a terminal sequence, disregarding all sequences in-between. The read-out is analyzed quantitatively by analyzing which primer combinations has yielded a band in the gel electrophoresis process. Those combinations without a band have not yielded any strands for amplification and therefore represent the value "0" in the matrix multiplication. Those that have yielded a strand therefore represent the value "1". This can be seen in Fig. 15.

4.3 Phase-Transition-Reaction

This method, shown in Fig. 16, uses that Dipyrromethene boron difluoride-based dye (BODIPY) is soluble in both water and in chloroform. BODIPY solved in water is named A and solved

in choroform it is named B. As this approach does not need intermediates the solutions are transferred directly to both outputs, with both having the same amount of chloroform. the outputs are then shaken to perform the transition of the dye from water to chloroform. The water is then removed and the chloroform evaporated. the corresponding flask is then put through UV-Vis spectroscopy [2].

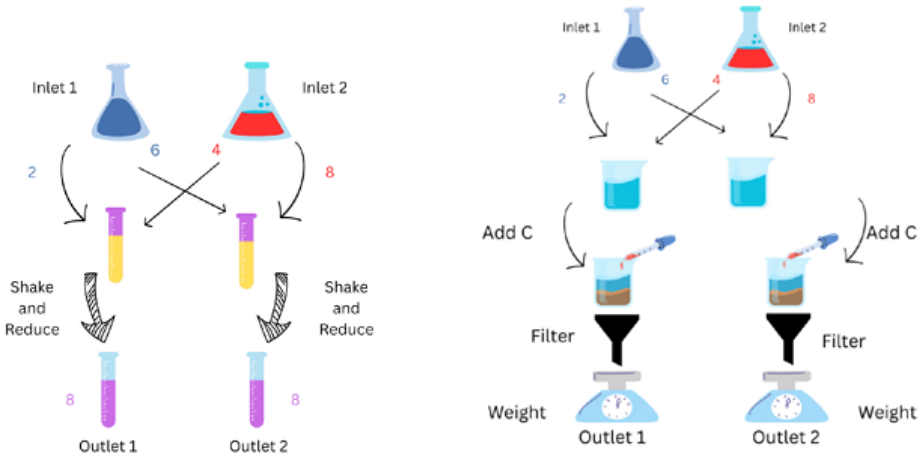

Figure 16: Demonstration of the phase-transition-reaction on the left, on the right the precipitation reaction, taken from [2]

4.4 Precipitation Reaction

The precipitation reaction , shown in Fig. 16, uses S^{2-} as A, CuS as B and, as the necessary reaction partner for A, molecule C is Cu^{2+}. A is solved in water and split to the intermediates according to the process, and to start the reaction C is added to the intermediate. The resulting molecules B precipitate to the bottom of the glass. The outputs are then filtered and the resulting substance is weighed. This weight is then converted to molarity by converting to mol and dividing by the volume [2].

4.5 Acid to Base Reaction

The last reaction is an acid to base reaction, performed similarly to the method shown in subsection 4.3. The molecules A and B are both phenolphthalein. The inlets contained HCL, with different amounts of A added to encode the initial concentrations. Furthermore, instead of shaking, 1M NaOH is added dropwise to change the pH from acidic to basic. The outlet concentration was then measured by a UV-Vis spectrometer [2].

4.6 Problems with transfer to the micro/nano scale

Some of the steps necessary to perform the methods mentioned in subsections 4.3, 4.4 and 4.5 are currently only done at a macro scale [2]. This is not necessary however, as the steps can also be performed using a microfluidic platform. The preparation of the solutions can be done in a reservoir connected to the microfluidic system [2]. Transport between compartments can be done by using microfluidic pumps with constant flow rates. Additionally mixing of the solutions can be done by, for example a zigzag mixing structure, or are not necessary as the amount of mixture used is small enough that the passive transition is fast enough to mix them [2]. As readout is not necessarily needed for the computation, they can be skipped [2]. Lastly Volume reduction devices, also known as microfluidic concentrators are a topic that is currently being researched.

5 Related Work

Molecular computing is a wide field, with new findings found yearly. Some additional papers to further understand are as follows. Paper [4] introduced the fura-2 sensor, used in section 3. Papers [5] and [9] implement some additional filters, and an 8-point FFT in a fully asynchronous system. Paper [6] first proposed using DNA to solve boolean matrix multiplications . Finally paper [1] introduces the theoretical framework for a molecular matrix multiplication of a positive valued 2x2 matrix. Furthermore the generalization to an arbitrary matrix is also included in this paper.

6 Conclusions, Results, Discussion

Although molecular computing can solve many in vivo or in vitro Problems. On the one hand, current progress is still mostly restricted to lab scale experiments proving feasibility instead of actually functioning systems. On the other hand, some systems have already reached maturity and can be used in other projects, such as the molecular sensors, in particular fura-2, implemented in [4]. This shows that, although there are still some big problems, for example the problem of connection between outputs of a module and the inputs of the next module, progress is being made. Continuing with research of this topic at the current rate will allow these problems to be solved in the next years, as can be seen in Section 3. Of the possible ways to perform matrix multiplications, for boolean matrices the parallel overlap assembly method has shown several advantages over the hybridization-ligation method, such as not needing heating/cooling cycles, restriction enzymes and the DNA sequence designs are more flexible. Additionally, in POA the values for the elements of the result are directly derived from the readout and the material consumption is lesser [7]. This shows a clear superiority of POA over Hybridization-ligation. For numerical matrices all three approaches show similar accuracies [2], therefore all of them are possible solutions to implement at the micro/nano scale. The indvidual steps that have to be transferred to the micro/nano scale have been shown in section 4.6. These steps, in particular the volume reduction, should be solved in some years, meaning that molecular matrix multiplication should work in a micro/nano scale in a few years, allowing for organic computers

used for massed matrix multiplications, for example, neural networks.

References

[1] Stefan Angerbauer, Franz Enzenhofer, Tobias Pankratz, Medina Hamidovic, Andreas Springer & Werner Haselmayr (2024): *Novel nano-scale computing unit for the iobnt: Concept and practical considerations. IEEE Transactions on Molecular, Biological, and Multi-Scale Communications* 10(4), pp. 549–565.

[2] Stefan Angerbauer, Nunzio Tuccitto, Giuseppe Trusso Sfrazzetto, Rossella Santonocito & Werner Haselmayr (2024): *Investigation of Different Chemical Realizations for Molecular Matrix Multiplications. IEEE Transactions on Molecular, Biological, and Multi-Scale Communications* 10(3), pp. 464–469, doi:10.1109/TMBMC.2024.3436905.

[3] Ramiz Daniel, Jacob R Rubens, Rahul Sarpeshkar & Timothy K Lu (2013): *Synthetic analog computation in living cells. Nature* 497(7451), pp. 619–623.

[4] Grzegorz Grynkiewicz, Martin Poenie & Roger Y Tsien (1985): *A new generation of Ca2+ indicators with greatly improved fluorescence properties. Journal of biological chemistry* 260(6), pp. 3440–3450.

[5] Hua Jiang, Sayed Ahmad Salehi, Marc D Riedel & Keshab K Parhi (2013): *Discrete-time signal processing with DNA. ACS synthetic biology* 2(5), pp. 245–254.

[6] John S Oliver (1997): *Matrix multiplication with DNA. Journal of Molecular Evolution* 45(2), pp. 161–167.

[7] N. Rajaee, Y. Kon, K. Yabe & O. Ono (2008): *Matrix Multiplication with DNA Based Computing: A Comparison Study between Hybridization-Ligation and Parallel Overlap Assembly.* In: *2008 Fourth International Conference on Natural Computation*, 4, pp. 521–525, doi:10.1109/ICNC.2008.604.

[8] Sayed Ahmad Salehi, Hua Jiang, Marc D. Riedel & Keshab K. Parhi (2015): *Molecular Sensing and Computing Systems. IEEE Transactions on Molecular, Biological, and Multi-Scale Communications* 1(3), pp. 249–264, doi:10.1109/TMBMC.2016.2537301.

[9] Sayed Ahmad Salehi, Marc D Riedel & Keshab K Parhi (2014): *Asynchronous discrete-time signal processing with molecular reactions.* In: *2014 48th Asilomar conference on signals, systems and computers*, IEEE, pp. 1767–1772.

[10] A. Prasanna de Silva & Seiichi Uchiyama (2007): *Molecular logic and computing. Nature Nanotechnology* 2(7), pp. 399–410, doi:10.1038/nnano.2007.188. Available at `https://doi.org/10.1038/nnano.2007.188`.