Characterization of Nanopores and Resensing of DNA Molecules Using a Logic-Driven Apparatus

Thomas St-Denis, Department of Physics

McGill University, Montréal

August, 2022

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Master of Science

Abstract

Solid-state nanopores are small holes on the nanometer scale that are widely used in biology, medicine and genomics to analyze various molecules. In particular, with a good knowledge of the characteristics of the specific nanopore used, like size, geometry and symmetry, one can determine basic features of molecules passing through the pores such as length, diameter and conformation. This project aims to better understand those characteristics by using a custom-built conditioning system capable of growing pores and rectifying their geometry while also tracking their conductance and size. Additionally, an FPGA system was made to control the direction of the flow of molecules to scan individual molecules several times (known as *resensing*). Not only does this system provide improvements to sensing accuracy from repeated sampling, but detailed information on transport mechanisms of molecules, such as knot-formation and folding probabilities, can be extracted.

Abrégé

Des nanopores à l'état solide sont des trous à l'échelle de nanomètres qui sont beaucoup utilisés en biologie, en médecine et en génomique. De façon plus précise, en possédant une bonne connaissance des caractéristiques d'un nanopore employé lors d'une expérience spécifique, comme ses proportions, sa géométrie et sa symétrie, il est possible de déterminer des éléments clefs de molécules qui voyagent au travers du nanopore, par exemple leur longueur, leur diamètre ainsi que leur conformérie. Le projet présenté dans cet ouvrage vise à approfondir les connaissances desdites caractéristiques des molécules en utilisant un système conçu pour agrandir les pores et rectifier leur géométrie tout en suivant de près les changements apportés à leur conductance électrique ainsi que leur diamètre. De plus, un system FPGA a été fabriqué dans le but de contrôler la circulation des molécules dans le but de détecter à plusieurs reprises une même molécule (un processus qui porte le nom de *redétection*). Non seulement ce system permet d'améliorer la précision de la détection par l'entremise d'échantillonnage multiple, mais il est également possible d'obtenir de plus amples informations sur le mécanisme de transport des molécules dans un fluide, comme leur probabilité de former des noeuds et de replier sur elles-mêmes.

Acknowledgements

I would like to thank my research supervisor, Walter Reisner, without whom none of this would have been possible. Professor Reisner has provided me with my very first research opportunity during my undergraduate degree, and has continued to support me until the end of my master's degree. I would also like to briefly thank Professor Peter Grutter, who co-supervised me as an undergraduate and has continuously provided me with mentorship and invaluable scientific knowledge. I would like to thank Professor Rob Sladek, Dr. Khadija Yazda, Dr. Yuning Zhao and Preethi Ravikumar for the the knowledge I have gained through many discussions and trainings with them.

I would like to thank Xavier Capaldi, who has given me an incredible amount of help throughout my research, from managing the laboratory, to helping me out with my research through both discussions and physical help, while also being a good friend for me to rely on.

I would like to thank Ioannidis Duchastel-Vassaramva and Mackenzie Pereira for their help in collecting the data shown in this work, as well as making life in the lab much more interesting.

I would like to thank my mom and my dad for providing me with food, a shelter for when I was writing my thesis, and for giving me their unconditional love. Additionally, I would like to thank the love of my life, my kitten Yuumi, who has waited for me to come back home after long days at the lab with a loving embrace. Finally, I would like to thank my friends Kiks, Señor Kachow, Froggeer, Matetcha, Professor Brabome and Grand Pooper for their support throughout my journey.

Statement of Contributions

The work done in this project is the culmination of two years of work that I did through my Master of Science. All the contributors to this project are from or affiliated to the Reisner Group. The entire project was under the supervision of my supervisor Walter Reisner, with valuable input and collaboration from Professor Rob Sladek

The pore fabrication (TCLB) setup was built by Yuning Zhang and subsequently improved by myself and Xavier Capaldi. The conditioning system and its initial code were developed by Xavier Capaldi. The improved version of the code was written by myself and by Ioannidis Duchastel-Vassaramva, an undergraduate student I was supervising. I performed conditioning experiments with Mackenzie Pereira, an undergraduate student who I was supervising, and the data shown in this thesis was obtained by the two of us. I also assembled the physical setup of the resensing FPGA system, with help from Xavier Capaldi, and wrote all of the LabVIEW code myself. Resensing experiments were conducted by both myself and Ioannidis Duchastel-Vassaramva, with tips from Xavier Capaldi. All the experiments presented in this work used nanopores fabricated using the TCLB method.

Table of Contents

Chapter 1

INTRODUCTION

Nanopores are small holes ranging from a couple to a few dozens of nanometers that can be found naturally in many different biological materials or can be man-made in a laboratory [1–5]. They are mainly used as single-channel recorders for molecular analysis; that is, ion flow from a salt solution as well as macromolecules such as deoxyribonucleic acid (DNA), ribonucleic acid (RNA) and proteins can be probed and studied using a single pore [6–8]. This is accomplished by reading the fluctuations in electric current generated when the ions and molecules go through the pore, a process known as translocation.

The idea of single-channel recording techniques first originated from work performed in the 70's by Bernard Katz and Ricardo Miledi [9]. They used membranes from human neurons which contained a minuscule channel (called acetylcholine channel), which would later be coined *nanopore*, and showed that ions from an electrolyte solution could not only pass through those channels, but also be probed to measure the size and resulting current signal of the opening in the membrane [6]. More specifically, the movement of charged particles of the same polarity through a medium is what constitutes an electric current, whose magnitude is directly correlated to the quantity of charged particles moving. The authors realized that given an opening small enough and a constant electric field across that opening, only a specific type of ion (either positive or negative) could move through the opening, which creates a very small current, given the fact that only a few

ions at a time are passing through the nanopore. This current could then be amplified by several orders of magnitude so that it can be analyzed, and important information on the conductance of the nanopore could be obtained.

Subsequently, it was theorized [10] and later shown [11] in the 90's that not only could this fixed current be read to analyze the nanopore itself, but fluctuations in the current could be detected and associated to specific events. In particular, careful analysis of pore conductance lead to the theory that the electric field created at the pore was strong enough to drive individual macromolecules such as DNA and RNA through the pore. Additionally, since DNA and RNA molecules have a much lower charge per volume occupied than an electrolyte solution with a high salt concentration, molecules moving across the pore prevent ions from flowing through the volume they occupy. This creates a reduction in the total amount of ions moving through the pore until the molecule leaves the pore, and this fluctuation can also be measured and subsequently analyzed. It was also found experimentally that those molecules move across the pore in a linear fashion, allowing the main features of molecules to be sensed individually. All of this combined demonstrated the potential of nanopores as tools to carefully study the structure of molecules. Since then, various studies have been conducted to better understand and improve the flow of ions and molecules through nanopores, as well as showing the viability of different kinds of pores, developing new techniques to fabricate and modify pores [1].

Originally, nanopore research was exclusively conducted using naturally occurring pores in biological material like living cells, such as α-hemolysin and Mycobacterium smegmatis porin A (MspA), which are called biological nanopores [12, 13]. However, in the 2000's, several methods to fabricate pores in insulated materials, known as solid-state nanopores, were developed [14–16]. To this day, the field of nanopore research is still divided between those two types of pores, which both provide their own share of advantages and disadvantages [4], and even though biological pores are capable of accomplishing more than their counter-part in terms of research, there are numerous reasons why the field might gravitate more towards solid-state pores in the future. Indeed,

many researchers have been and are still currently trying to bridge the gap between the capabilities of the two nanopore types [3, 17–19].

Among the research done to improve solid-state nanopores, some have worked on improving the quality of pores fabricated as well as give better control over their size [20], using a process known as conditioning. Nanopore conditioning relies on the application of a high voltage to the pore in repeated pulses, causing an alteration in its geometry. Using conditioning, several experiments that could provide advanced knowledge on the inner workings of nanopores and which have yet to be conducted, such as carefully tracking growth under different experimental conditions.

Additionally, another avenue that has been explored requires slight modifications to the experimental setup used in regular experiments which allows individual molecules flowing through the pore to be probed repeatedly in a way that potentially greatly improves sensing accuracy [21, 22]. The setup used in initial experiments was very rudimentary and proved to be too unreliable to be worth using in experiments, as just a few successful events could be obtained from thousands of molecules passing through the pore. However, new technologies developed since have shown the potential to be used to drastically improve resensing [23–25].

In this work, an overview of basic concepts at the core of nanopore-based research, such as ion transport, molecular translocation and sequencing, will first be introduced. After establishing these fundamental notions, an in-depth review of biological and solid-state nanopores, alongside their many advantages and disadvantages, will be presented. Then, the experimental setup and process used to fabricate pores and use them to perform basic experiments as well as conditioning will be detailed. An introduction to the software used to perform resensing experiments will be shown, alongside the detailed code and its subcomponents. Finally, preliminary results on conditioning and resensing will be presented, with this initial work demonstrating successful experiments using both systems and providing a glimpse of their potential.

Chapter 2

THEORETICAL BACKGROUND

2.1 Basic Principles of Nanopores

2.1.1 Ion Transport

Ion transport is the physical principle that lies at the heart of every experiment that uses nanopores as single-channel recorders. Indeed, the flow of ions through the pore generates a current, which is then probed and subsequently analyzed to deduce any and all information about not only the pores themselves, but also the molecules studied.

In a typical nanopore setup, the pore is placed between two isolated reservoirs filled with an electrolyte solution, typically a salt buffer such as lithium chloride (LiCl) or potassium chloride (KCl). Electrodes are placed in both reservoirs and are connected to a voltage source, which applies a bias voltage through the electrolyte solution via the electrodes (figure 2.1 a). This creates an electric field (figure 2.1 b) which one might expect would cause ions from both directions to flow through the pore. However, if the pore is small enough or if the salt has a low concentration, a certain kind of ions, either cations (positively charged) or anions (negatively charged) depending on the polarity of the bias voltage, can be prevented from crossing the pore (figure 2.1 c). This movement of charges, which is simply an electric current, is then read by the electrodes and sent to the analysis

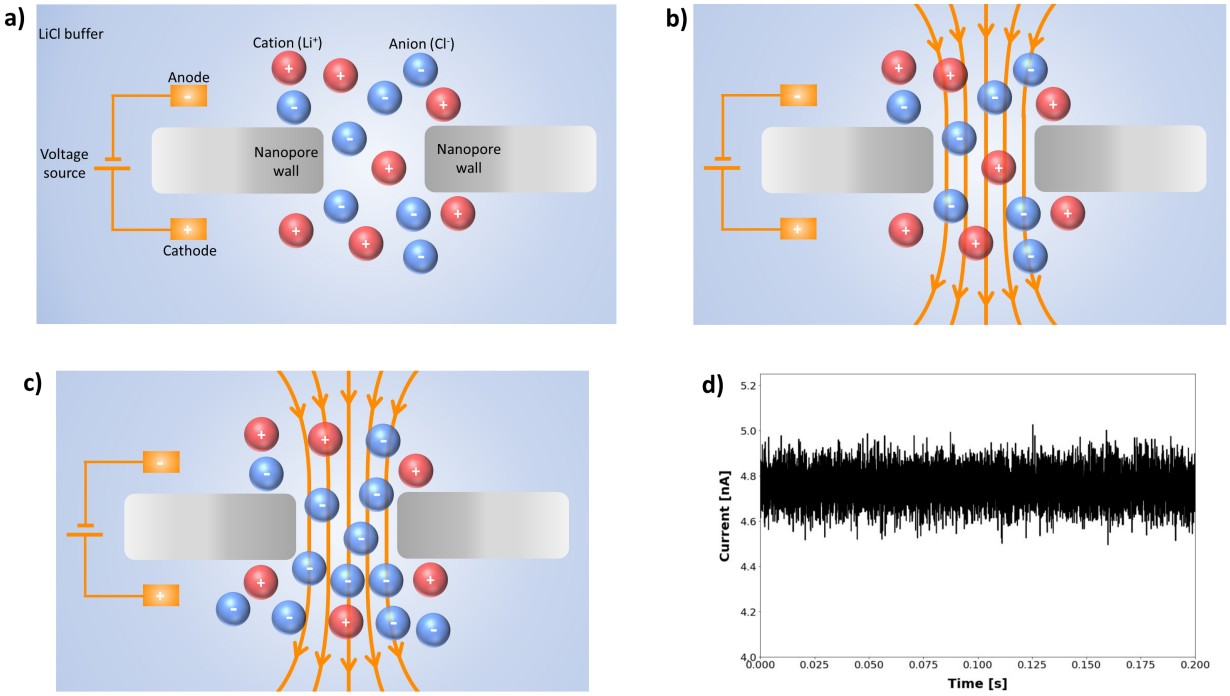

Figure 2.1: Ion transport in a nanopore. **(a)** Typical nanopore setup. The membrane (in grey) sits in an electrolyte solution (LiCl in this case) and electrodes (a positive electrode called cathode and a negative one called anode) connected to a voltage source are inserted in the buffer. The concentration of cations and anions on both sides is the same. **(b)** Generation of an electric field at the pore following the application of voltage. **(c)** Displacement of the charges. The negative charges follow the electric current, and there is a migration of negative charges across the pore in the same direction, which creates an electric field. **(d)** Typical current generated by the flow of ions. Since there aren't a lot of ions flowing though the pore (because of its small size), the generated current is on the order of nanoAmperes. Note the presence of a high amount of noise, which will be covered in section 2.1.2.

apparatus, generating a signal like the one shown in figure 2.1 d. In order to understand in mathematical detail how such a current is induced through a nanopore and what specifically can be deduced from this current, we will first look at the concept of Brownian motion.

Brownian Motion

Brownian motion corresponds to the random movement of a particle (known as "Brownian particle") present in either a gas or a liquid medium. Molecules comprising such mediums, called solvent molecules, are constantly jiggling and moving around due to their thermal energy. When placing a Brownian particle in this medium, it was discovered [26], and later modeled [27], that solvent molecules will periodically collide with the Brownian particle, in a completely nondeterministic, uncorrelated way. These collisions transfer the kinetic energy of the solvent molecules to the particle, which will then constantly move around in a random fashion. This can be mathematically represented using the following equation of the mean square displacement (the standard measure of the deviation of the position of a particle):

$$\overline{x^2} = 2\ D\ t, \tag{2.1}$$

where x is the displacement of the particle, D is the coefficient of diffusion of the medium (typically on the order of 10^{-10} m^2/s and t is the time.

Brownian motion is a phenomenon that can never be removed, and at the nanometer scale, where nanopores operate, it actually influences the recorded signal quite drastically. Indeed, particles moving across the pore will do so at different velocities, depending on their initial momentum gained from collisions. For instance, if a particle is pushed from behind towards the pore, its velocity while traversing the pore will be higher than if the collision happens in the opposite direction (pushing it away from the pore). In fact, since nanopores are large enough to let a few molecules of solution through at the same

time, Brownian motion still occurs while inside the pore, causing the molecules to move forwards and backwards while being sensed. This introduces an intrinsic noise to any signal, which can be quite problematic, as it will be discussed in section 2.1.2.

Ion Motion with an Applied Electric Field

With Brownian motion established, it becomes possible to determine the flux of ions through the pore, and thus establish what the signal generated by a nanopore system is. As mentioned in the previous section, a particle placed in a medium (in this case a liquid solution) of temperature T will have a Brownian motion. However, since this motion does not have any preference in its direction, over the course of a long enough time and by taking into consideration a high number of particles, Brownian motion will actually average out to zero (meaning that Brownian motion does not really affect the translocation of the particle). It also follows from Langevin dynamics, which has been shown to be a reliable approximation for such a system [28], that the particle will have an instantaneous velocity ν given by the equation:

$$\nu = \frac{\mathrm{d}x}{\mathrm{d}t}. \tag{2.2}$$

Since the particle is pulled by the electric field across an aqueous medium, there will be a hydrodynamic drag force F_{drag} opposing the movement. Furthermore, according to simple electrodynamics, an applied electric field E on the molecule of charge q will create an electric force given by:

$$F_{elec} = q\ E. \tag{2.3}$$

The system can thus be represented using Newton's second law in the following way:

$$m \cdot \frac{\mathrm{d}\nu_{drift}}{\mathrm{d}t} = F_{elec} - F_{drag} = q\ E - F_{drag}. \tag{2.4}$$

This system represents particles in a material in the presence of an electric field and thus follow the principle drift velocity. In particular, the drag force can be rewritten in terms

of a simple proportion from the total amount of particles α that move with the flow with a velocity ν_{drift}. This allows for equation 2.4 to be rewritten as:

$$m \cdot \frac{\mathrm{d}\nu_{drift}}{\mathrm{d}t} = q\ E - \alpha\ \nu_{drift}. \tag{2.5}$$

Since the applied electric field remains constant throughout the experiment, the system is unchanged in time and is thus considered in a steady-state. In other words, the drift velocity does not change in time, and thus the left-hand side of equation 2.5 becomes zero. This leads to the following equation:

$$q\ E = \alpha\ \nu_{drift} \implies \nu_{drift} = \frac{q\ E}{\alpha}. \tag{2.6}$$

With this, it becomes possible to find the flux of ions through the pore. The flux j of ions i, which have a concentration of c_i in a solution, is given by the following equation:

$$j_i = c_i\ \nu_{drift} = c_i \cdot \frac{q\ E}{\alpha}. \tag{2.7}$$

From thermodynamics theory, we know that electric force can be represented by the opposite of the position derivative of the electrochemical potential μ_{elec}:

$$q\ E = -\frac{\mathrm{d}\mu_{elec}}{\mathrm{d}x}. \tag{2.8}$$

Combining this with equation 2.7 yields:

$$j_i = -c_i \cdot \frac{1}{\alpha} \cdot \frac{\mathrm{d}\mu_{elec}}{\mathrm{d}x}. \tag{2.9}$$

Additionally, using the following Einstein relation linking the proportion of ions moving along the flow α to the diffusion of the ions in solution D_i, the universal gas constant T and the temperature T:

$$\frac{1}{\alpha} = \frac{D_i}{RT}, \tag{2.10}$$

the flux equation can be written as:

$$j_i = -c_i \cdot \frac{D_i}{RT} \cdot \frac{\mathrm{d}\mu_{elec}}{\mathrm{d}x}. \tag{2.11}$$

Finally, in a solution at constant pressure, the electrochemical potential can be approximated to first order as:

$$\mu_{elec} = \mu_{chem} + R\,T \cdot \ln \gamma_i + q_i \cdot F \cdot \Phi, \tag{2.12}$$

where μ_{chem} is the chemical potential (which is independent of position), γ_i is the thermodynamic activity coefficient (a unit-less factor), F is Faraday's constant and Φ is the (static) local electric potential [29]. By combining equations 2.11 and 2.12, and using the relation $E = -d\Phi/dx$, we can compute the flux of ions through a nanopore:

$$\begin{aligned} j_i &= -c_i \cdot \frac{D_i}{R\,T} \cdot \frac{\mathrm{d}}{\mathrm{d}x}\left[\mu_{chem} + R\,T \cdot \ln \gamma_i + q_i \cdot F \cdot \Phi\right] \\ &= -c_i \cdot \frac{D_i}{R\,T} \cdot \left[q_i \cdot F \cdot \frac{\mathrm{d}\Phi}{\mathrm{d}x}\right] \\ &= \frac{D_i\, q_i\, c_i\, F}{R\,T} \cdot E. \end{aligned} \tag{2.13}$$

From this final equation, one can deduce how the flow of ions through the pore is affected from various parameters. First, as expected, at higher D_i, the ions move around more freely and are thus more easily influenced to more through the pore, so the higher the flux. Also, the higher the driving electric field E, the higher the flux, which is also not surprising since the field attracts ions towards the pore. At higher temperatures, ions are more thermally excited, which increases their Brownian motion, and thus increases the drag force, which results in an overall decrease in flux. Finally, increasing the concentration c_i increases the total amount of ions available to flow through the pore, and thus the flux.

Pore Conductance and Size for Thick Pores

Equation 2.13 is extremely useful to understand the relation between the ionic conductance of a given pore and its size, as will be demonstrated below. The implication of this is that from a simple measurement of a current-voltage (IV) curve, it is possible to determine the size of nanopore precisely, without needing to use advanced imaging techniques such as transmission electron microscopy.

Conductance is simply given as the ratio of current over voltage applied, or I/V. It is also very dependent on pore geometry, and we will assume that the channel of the pore has a cylindrical shape, as it is the simplest and most studied geometry. It turns out that variations from this shape do not affect conductance in a significant way when attempting to find pore diameter [30].

In a simple electrolyte solution with a positive and a negative ion, the current density J is given by the sum of the current flux j_i of each ion (equation 2.13 adjusted with the right unit convention); that is to say:

$$\begin{aligned} J &= F\left[q_+\, j_+ + q_-\, j_-\right] \\ &= F\left[q_+ \cdot \frac{D_i\, q_i\, c_i\, F}{R\,T} \cdot E + q_- \cdot \frac{D_i\, q_i\, c_i\, F}{R\,T} \cdot E\right] \\ &= \frac{q^2 \cdot F^2 \cdot c \cdot E}{R\,T}\left[D_+ + D_-\right] \\ &= \left\{\frac{q^2 \cdot F^2 \cdot c}{R\,T}\left[D_+ + D_-\right]\right\} E. \end{aligned} \tag{2.14}$$

This whole equation is simply the solution conductivity σ multiplied by the electric field, which gives us the equation:

$$J = \sigma \cdot E. \tag{2.15}$$

The current density is the ratio between the current I through the pore and the cross-sectional area A of the pore. For a cylindrical pore, A is given in terms of the diameter d

as $A = (\pi d^2/4)$, so the current is:

$$I = J \cdot A = \sigma \cdot E \cdot \frac{\pi d^2}{4}. \tag{2.16}$$

To derive an equation for the electric field, we must find where the input voltage drops, which requires to know the impedance of the components of the system. In this case, there are only three components that have an impedance: the sensing electronics, the electrodes and the nanopore itself. It is a well-known fact from electronics [31] that the source impedance (here, that of the sensing and electrodes) must always be small in order to avoid loading and to provide the load (the nanopore) as much of the input voltage as possible. Thus, it is valid to approximate that the voltage drops entirely at the pore. In addition, for a cylindrical pore of small diameter, the field lines of the electric field through the pore are approximately parallel to one another (see figure 2.2), meaning that:

$$E = \frac{\mathrm{d}\Phi}{\mathrm{d}x} = \frac{\mathrm{d}V}{\mathrm{d}x} \approx \frac{V}{L}. \tag{2.17}$$

Combining equations 2.16 and 2.17 gives:

$$I = \sigma \cdot \frac{\pi d^2}{4} \cdot \frac{V}{L} \Longrightarrow G = \frac{I}{V} = \sigma \cdot \frac{\pi d^2}{4L}. \tag{2.18}$$

In typical experiments, the thickness of the membrane and the conductivity of the solution are always known, so by measuring the conductance of the pore, one can approximate its diameter from this simple equation in the case where the membrane thickness is much bigger than the diameter of the pore.

Pore Conductance and Size for Thin Pores

Now if we consider the case where the thickness and the diameter of a pore are on the same order of magnitude (which will be the case for all pores used in experiments presented in this work), we need to include field effects on the boundaries of the pore. These

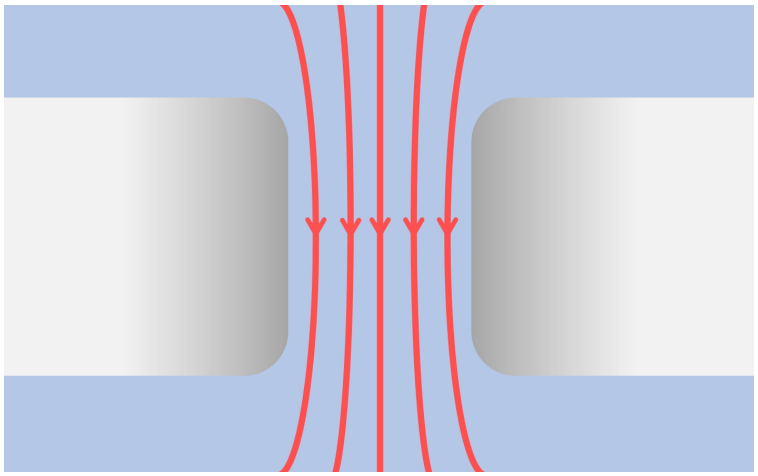

Figure 2.2: Electric field lines through a nanopore. Shown in grey is a cross-sectional view of a membrane, with the hole between the two blocks representing the nanopore, in red are the field lines from the electric field and in blue is the buffer solution. Far away from the pore, the field lines have their expected curvature, but because of the cylindrical shape of the pore and its small diameter, the field lines inside are parallel to one another through most of the membrane. The effects at the boundaries of the pore can thus be neglected when computing the conductance.

effects are known as *access resistance*, and one can think of it as the impedance the ions encounter as they access and leave the pore. In this case, the resistance across the pore itself, where in the previous case all the potential was dropping, is on the same order as the access resistance, so the combined equation for the total resistance of the system is:

$$R_{tot} = R_{pore} + 2 \cdot R_{access}. \tag{2.19}$$

The expression for the pore resistance is given simply by the relation $G = 1/R$, with the equation for G being the conductance from equation 2.18. The expression for the access resistance was found by James E. Hall in 1975 [32]. He argued that, since the region of impedance at the edge of the pores is an equipotential surface (that is, same potential everywhere), finding the access resistance amounts to computing the resistance between a conducting disk on an insulator and a hemispherical electrode with field lines extending to infinity (see figure 2.3). To do so, he first used the following relation [33] between the

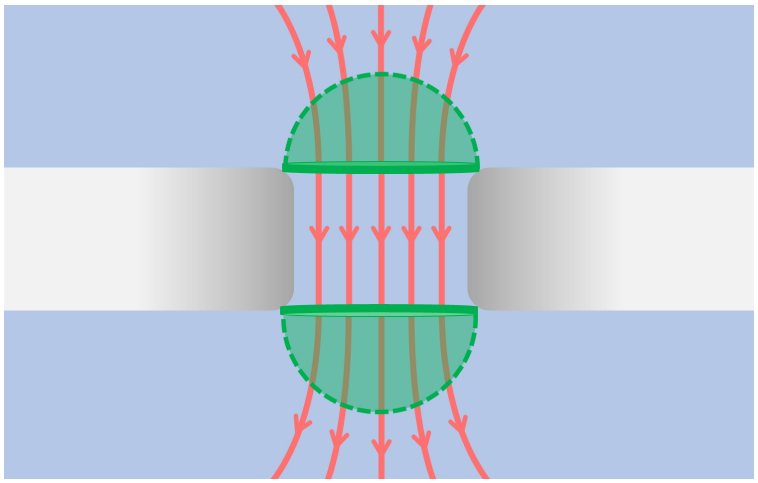

Figure 2.3: Field effects of a pore with edge effects. The parallel field lines inside the pore corresponding to R_{pore} are still present, but now edge effects are also taken into consideration. This is done by representing the impedance at the edge as the resistance between a conducting disk (in thick green) and a half-sphere electrode (in transparent green). Note that in this case, the field lines extend towards infinity, which is relevant when computing the integral leading to the result of equation 2.20.

resistance R of electrodes in a conducting medium and capacitance C of electrodes in an insulating medium of permittivity $\epsilon = \epsilon_r \epsilon_0$:

$$R = \frac{\epsilon}{\sigma \cdot C}. \tag{2.20}$$

The total capacitance of such a conducting disk is [34]:

$$C = 4 \cdot \epsilon \cdot d, \tag{2.21}$$

but this is given an electric field on both sides of the disk. In this case, the disk at the edge of the pore only has field lines on one side, so the capacitance used in equation 2.20 is half that of equation 2.21. Equation 2.20 then becomes:

$$R = \frac{\epsilon}{\sigma \cdot 4 \cdot \epsilon \cdot d} = \frac{1}{2 \cdot \sigma \cdot d}. \tag{2.22}$$

Therefore, the expression for the total resistance becomes:

$$\begin{aligned} R_{tot} &= \frac{4L}{\sigma \cdot \pi d^2} + \frac{2}{2 \cdot \sigma \cdot d} \\ &= \frac{1}{\sigma}\left[\frac{4L}{\pi d^2} + \frac{1}{d}\right]. \end{aligned} \tag{2.23}$$

Using the same relation between conductance and resistance mentioned before, we obtain the following equation for the total conductance of the nanopore:

$$G = \sigma \cdot \left[\frac{4L}{\pi d^2} + \frac{1}{d}\right]^{-1}. \tag{2.24}$$

This equation has been shown experimentally by Dekker's group [30] to be a very reliable way to determine pore size based on diameter. They have also shown that it holds fairly well even for hyperboloid-shaped pores, despite assuming a cylindrical shape, though the thickness used must be reduced compared to the membrane thickness. This is because an hourglass shape can be seen as a cylinder in the center, and a bowl of larger diameter on both ends. The flow of ions is constricted at the narrow part of the pore, which is the cylinder, so this part dictates the conductance. Since the cylinder is not the entire pore, the thickness L used in equation 2.24 has to be smaller than the thickness of the membrane.

Conductance is an extremely important aspect of experiments that focus on the pores themselves and their characteristics. However, it is also a key principle when using pores to study molecules, as it is the reason why molecules move through the pore in the first place, a process which is known as *translocation*.

2.1.2 Translocation

In short, translocation is the passage of any molecule through a pore, as shown in figure 2.4 a. More specifically, typical molecules used in translocation experiments are DNA, RNA and proteins, all of which possess a non-neutral charge. The principle of translo-

cation of charged molecules is very similar to what was described for the case of ion transport, where the electric field generated pulls the molecule through the pore.

As mentioned in the previous sections, since virtually all of the potential drops inside or very close to the pore, the potential far away from the pore is constant, and thus no significant electric field is generated. This means that molecules in solution far from the pore will not have a preference to move towards (nor away from) it.Also, the presence of a drag force caused by Brownian motion indicates the need for a minimum electric field strength in order for the molecule to move towards the pore (section 2.1.1). Since potential is inversely proportional to distance, the electric field close to the pore decreases with distance as well. This means that there is a specific distance from the pore where the electric force becomes strong enough to overcome the drag force, which is known as the capture radius. This capture radius is dependent on many factors that dictate how easily the molecule can move in solution such as molecule length and concentration, salt concentration and pH, as well as factors influencing the strength of the electric field generated, like applied potential and nanopore geometry and surface chemistry [7]. Generally, molecules will move around randomly due to Brownian motion until they cross the capture radius, after which they will inevitably translocate.

Since at high salt the charge of the molecule is typically much lower than that of the ions in the solution, translocating molecules do not increase the measured electric current. In fact, they have the opposite effect. The electric current through the pore, also called trans-pore current, is simply a measure of the amount of charge flowing across the pore. When a molecule translocates, it blocks a region of the pore proportional to the volume it occupies, and thus the net flow of ions through the pore is reduced. This results in a net decrease in current, called a blockade current. Figure 2.4 b illustrates the effect of a molecule on the trans-pore current before (I), during (II) and after (III) translocation.

In general, the current blockade is very proportional to the volume of the molecule. For example, a molecule that is twice as big as another will create a blockade twice as high, and a molecule twice as long will create a blockade that lasts double the duration.

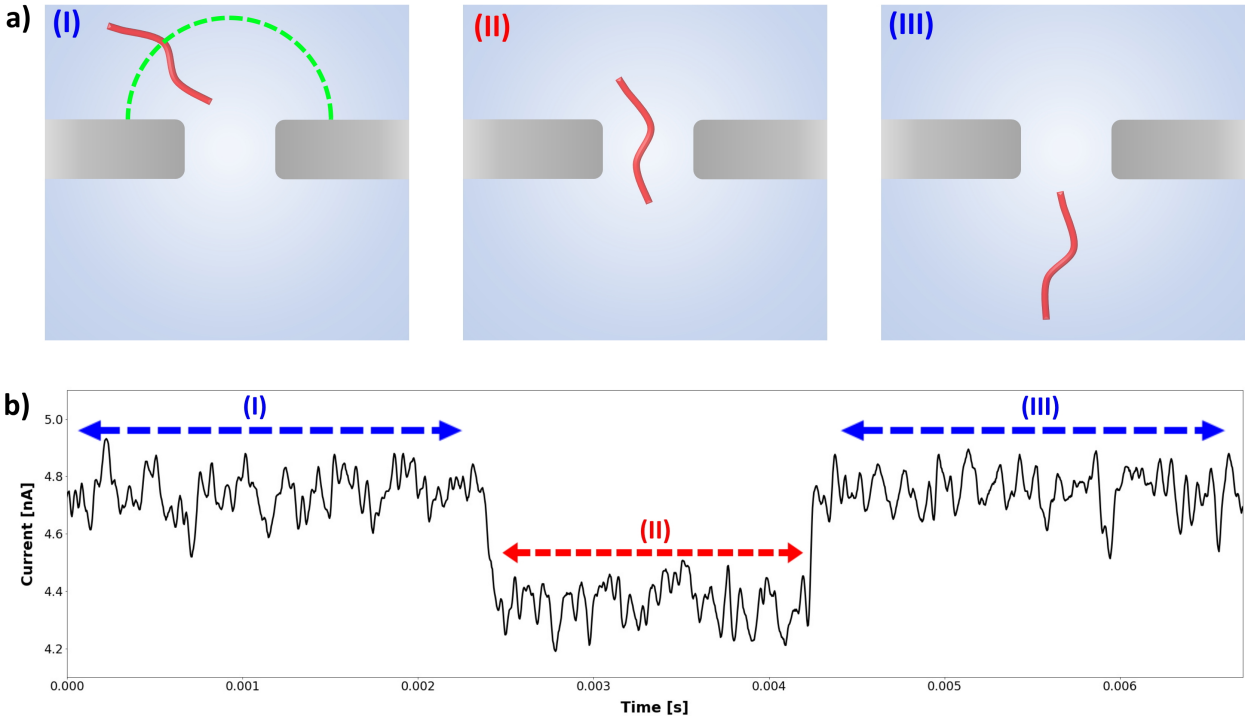

Figure 2.4: Molecule translocating through a pore. (a) Representation of the three main stages of translocation. (I) The molecule (shown in red), after moving around randomly in the solution for a while, crosses the capture radius of the pore (green hemisphere), at which points the electric field attracts the molecule towards the pore. (II) The molecule translocates through the pore. (III) The molecule escapes the pore and is never seen again. **(b)** Typical current signal from a translocation event. Regions (I) and (III) show the open-pore current, where the maximum amount of ions flows through the pore (and thus, the highest possible current is generated). Region (II) corresponds to the current blockade caused by the translocation of the molecule. When the molecule is inside the pore, the flow of ions is restricted, leading to a decrease in current. Note that the fluctuations in the signal are due to Brownian motion and are further discussed in section 2.1.3.

From this, it is evident that simple analysis of molecules using nanopores to study their sizes, geometry, length and so on is easily doable. However, sensing differences on the atomic level in characteristics of molecules, like is required to perform sequencing, turns out to be extremely challenging.

2.1.3 Sequencing

Before we can cover the complications related to nanopore-based sequencing, we must first understand the structure of nucleic acid (DNA and RNA, but we will focus on DNA) and introduce the basic underlying principles of behind sequencing. The goal of this section is simply to highlight what this process entails on an elemental level without going into details in the specific techniques used, as this would be more relevant to a study on biological nanopores. The key aspects of biological nanopore-based sequencing are reviewed in section 2.2.1.

DNA is a molecule part of the family known as "nucleic acids" alongside RNA, and is made from four nucleotides or bases, as shown in figure 2.5 a. Those four bases are cytosine (C), thymine (T), adenine (A) and guanine (G). Any DNA molecule is made of a specific combination of those four building blocks, and finding said combination is of crucial importance for several applications, like when studying specific diseases to better understand them and to find potential treatments or cures.

In particular, those four bases have slightly different sizes, which affects how many ions can flow in the pore. Indeed, when the biggest base translocates, fewer ions would be able to go through the pore compared to the other bases, which would create a bigger blockade. In theory, since nanopores can sense the flow of individual ions, any DNA translocation signal should have four different blockade levels corresponding to the four bases, as shown in figure 2.6 a.

However, as discussed in section 2.1.1, ions and molecules undergo Brownian motion, and this creates high fluctuations in the measured trans-pore current. In fact, it turns out that in regular translocation experiments, those fluctuations are much higher than

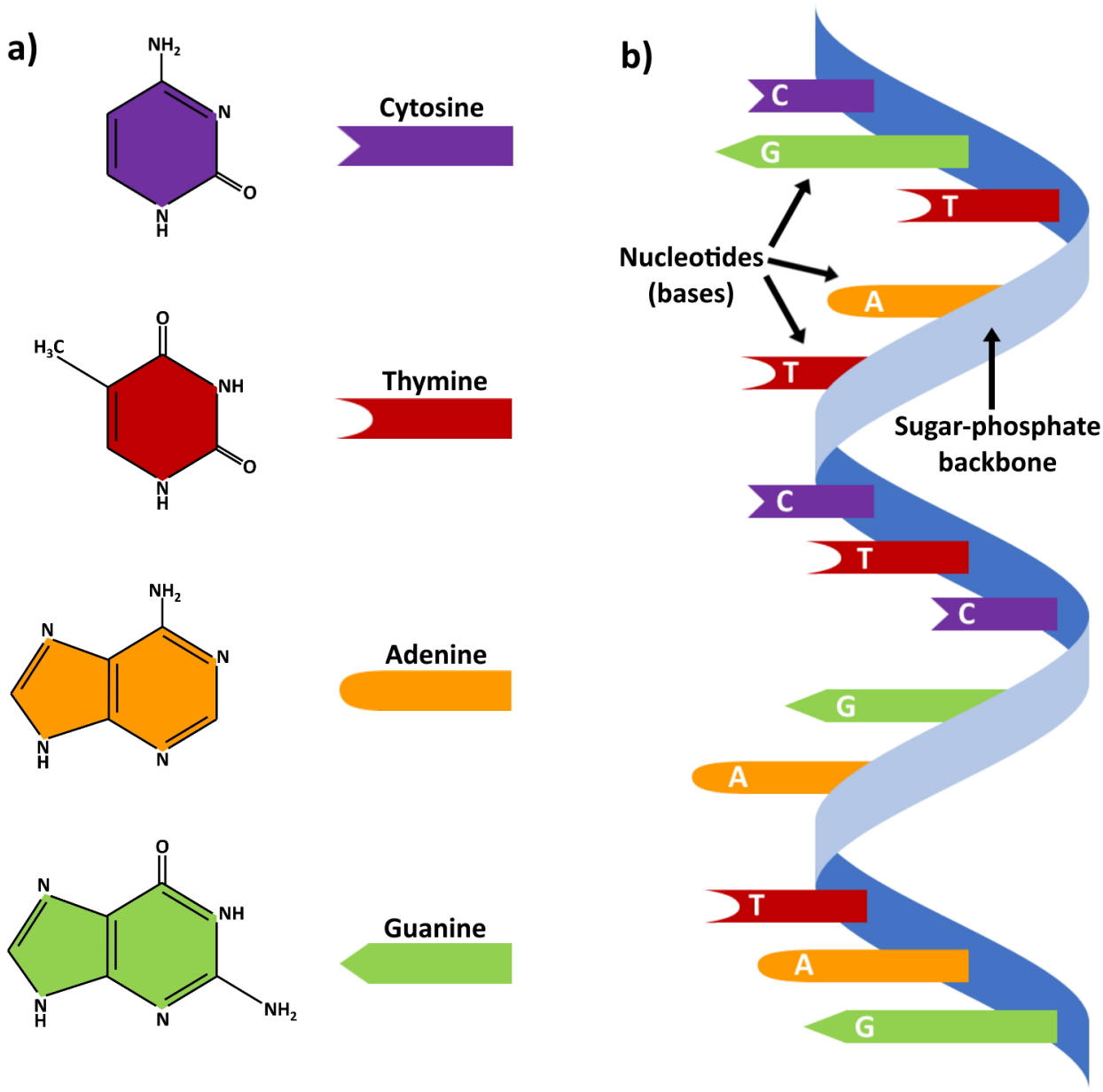

Figure 2.5: Structure of DNA. On the left is a molecular representation of the four nucleotides that make up DNA placed in order of size, with cytosine being the smallest and guanine the biggest. On right is a representation of a single-stranded DNA. It is comprised of a sugar-phosphate backbone on which nucleotides sit. Note that in this case, the sequence of the DNA would be "CGTATCTCGATAG" (read from top to bottom).

the differences in blockade between the bases (see figure 2.6 b). For the most part, this is caused by molecules translocating too fast through the nanopore (a typical translocation speed is one nucleotide per microsecond [3]). The faster the translocation, the broader the bandwidth of the sensing electronics must be to detect the event, and thus noise from bigger ranges of frequencies is introduced to the signal. Also, since the sampling rate of the apparatus is constant, faster translocations allow for fewer samples to be taken, and thus it becomes increasingly difficult to average out the noise. Currently, using state-of-the-art sensing technology and electronics, it is only possible to reliably measure translocation velocities up to one base per millisecond [4]. Thus, any nanopore-based experiment requires significant modifications in order to slow down translocation speeds by at least three orders of magnitude and successfully perform DNA sequencing, which is already possible for biological nanopores (see section 2.2.1) and is a hot topic for solid-state nanopores [35–38].

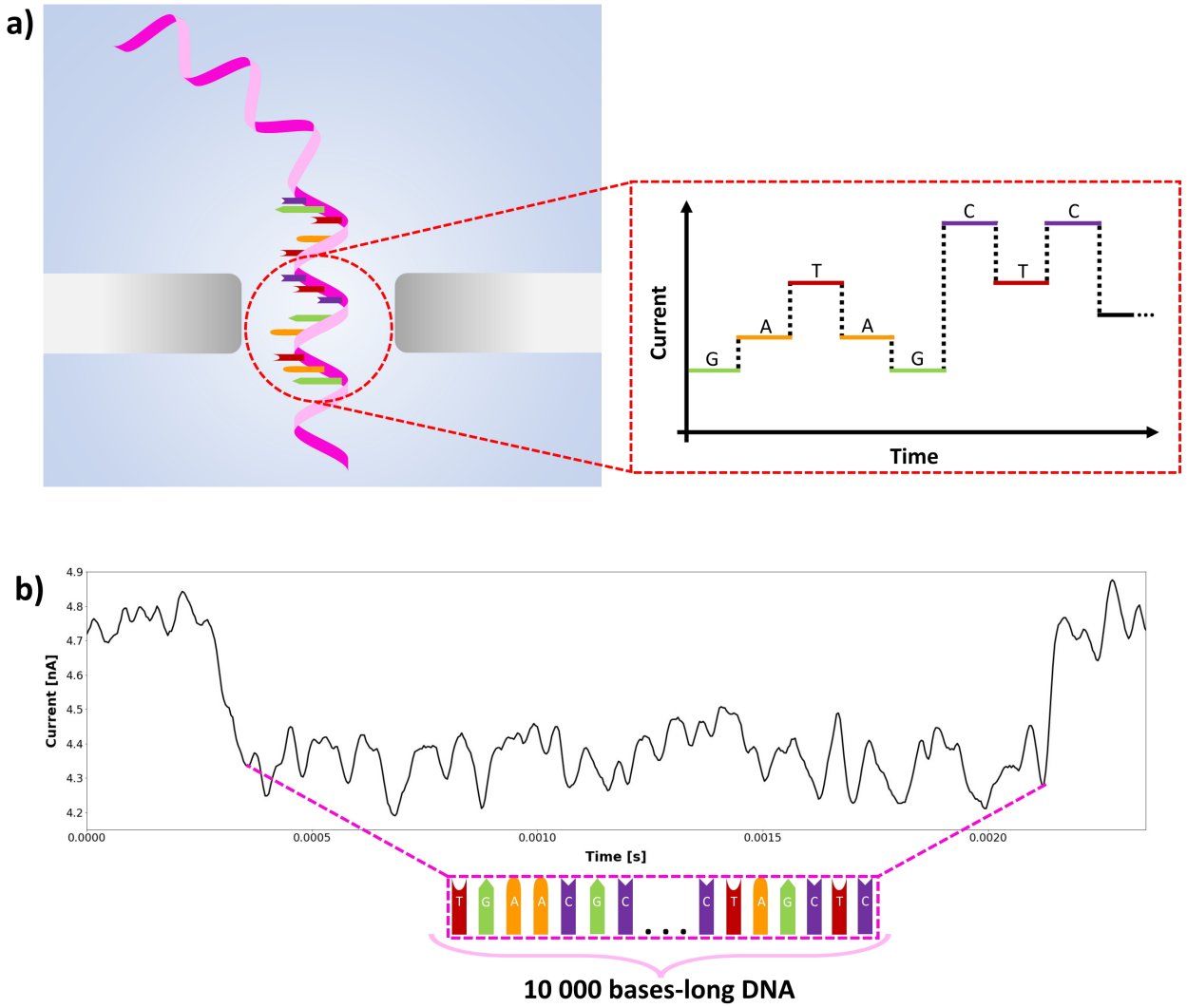

Figure 2.6: Nanopore-based DNA sequencing concept. **(a)** Translocation of a single-stranded DNA and its expected current (without the presence of noise). Since all four nucleotides have different sizes, they restrict the flow of ions in slightly different amounts. This should in theory create four plateaus corresponding to each base as they translocate, as shown on the right. **(b)** Typical unmodified translocation signal. This is a translocation of a 10 000 bases-long segment of DNA in an unmodified solid-state nanopore. Clearly, not only is it impossible to tell the bases apart, but there are nowhere near 10 000 different plateaus representing each base.

2.2 Types of Nanopores

2.2.1 Biological Nanopores

Biological nanopores are based on naturally forming trans-membrane proteins. Biological pores were the first type of pores ever used in research, and are used routinely today in research and commercially available instrumentation. One classic type of protein pore is based on "α-haemolysin," a protein toxin secreted by bacteria. The pore is formed in a lipid membrane that attaches itself to the outer layer of the protein (figure 2.7). It has a double-barrel shape, with the biggest opening being around 1.4 nm wide, which is big enough to let single-stranded DNA through, but not double-stranded. Most biological nanopores have very specific shapes, like that of α-haemolysin, which is composed of a double-barrel structure with a large opening at the top and a narrow opening at the bottom (figure 2.7), with a fixed diameter. The most important aspect of biological nanopores is their compatibility with other biological materials, which enable DNA sequencing.

Biological Nanopores for Sequencing

As mentioned in section 2.1.3, the main hurdle in the way of nanopore-based sequencing is the high translocation velocities. Simply modifying the running buffer or improving the sensing electronics is far from sufficient to slow down DNA molecules by three orders of magnitude. Instead, biological nanopores rely on the fact that due to the ease with which biological molecules can be interfaced, it is possible to engineer the pores by adding an enzyme in front (figure 2.7). Specifically, the molecule of choice is an enzyme called DNA polymerase, which controls the flow of DNA through the pore. In particular, the DNA polymerase ratchets the DNA through the pore one base at a time. This dramatically decreases translocation velocities to a point where base-by-base readings can be performed, and thus the sequence of the molecules under analysis can be found.

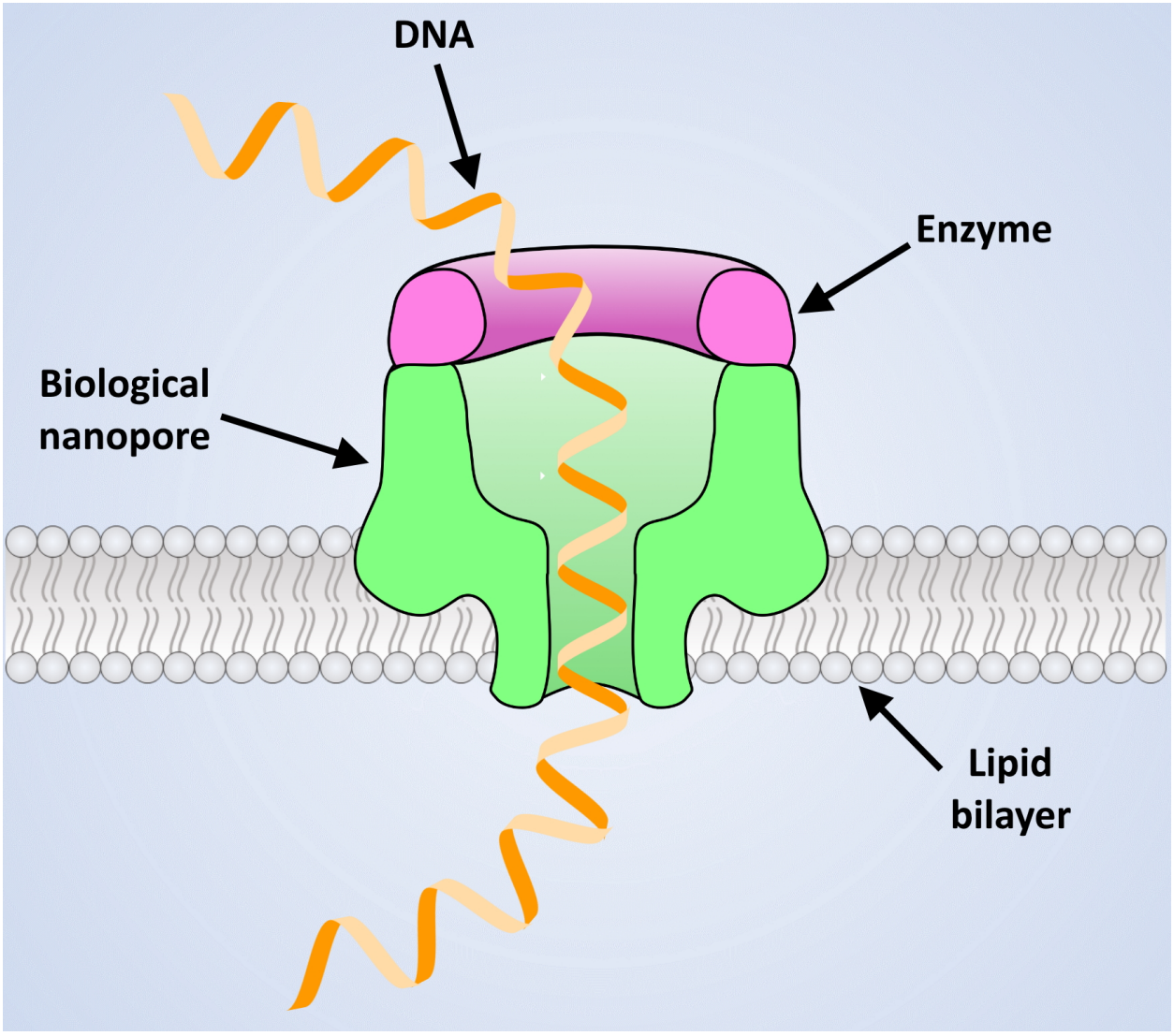

Figure 2.7: α-hemolysin biological nanopore. Shown here is a α-hemolysin biological nanopore used in a typical DNA sequencing setup. The pore itself, which comes from bacteria, is located inside a lipid bilayer which provides support for the pore. It lodges itself in the bilayer by moving the lipids out of the way and bonding with them all around, ensuring that molecules can only go through the pore. Attached on top of the pore is an enzyme, in this case DNA polymerase, which is used to decrease the translocation velocity of the DNA by reducing its kinetic energy.

Pros and Cons

The most important positive aspect of biological nanopore is their ability to perform sequencing. In particular, very long sequences can be obtained using nanopore sequencing in a fraction of time and cost that other methods would require (with the PacBIO system being the only technique comparable in terms of read leangth [39]). A nanopore-based sequencing setup can even fit inside a pocket, like the *MinION* device from Oxford Nanopore [40], which makes it very practical. The ability to perform sequencing is what elevates biological nanopores from just a tool in research to a state-of-the-art instrument for commercial applications [5, 41].

However, that is not to say that biological nanopores do not have their shortcomings. First, like it was alluded to before, those pores have a very specific size that cannot be changed. Certain experiments, such as protein sensing, can require a precise pore size, and unless a molecule that contains nanopores with the required diameter is found in nature, it is impossible to use a biological pore for that experiment [1]. Additionally, it is very difficult to integrate biological nanopores with devices, since the pores cannot be fabricated and need to be placed pre-made [42]. On a more fundamental level, biological pores are quite unstable to any change in the experimental conditions such as temperature and solution pH, and are not very thermally and mechanically robust [3]. Overcoming these weaknesses became the main goal of many scientists, and this is why solid-state nanopores were created.

2.2.2 Solid-State Nanopores

A solid-state nanopore is a hole fabricated in a thin membrane made from an insulated material. Typically, those membranes range from one to a few dozens of nanometer in thickness and the most popular material used is silicon nitride (SiN_x), though some experiments have been conducted using silicon oxide (SiO_2) and graphene [3, 43]. Shown in

figure 2.8 a is a transmission electron microscope image of a 5 nm pore in a 12 nm thick silicon nitride membrane.

Fabrication Techniques

Modern solid-state nanopore fabrication techniques are derived from the same three basic techniques: focused ion beam (FIB), transmission electron microscopy (TEM) and controlled dieletric breakdown (CBD, which stands for **c**ontrolled **b**reak**d**own).

The first major technique was developped by Golovchenko's group in 2001 [14] and utilised a focused ion beam instrument to sculpt a pore through a membrane. More specifically, this fabrication method used a home-built FIB with a high energy beam made of positive argon ions to bombard the surface of the membrane and physically pierce a hole (figure 2.8 b). Over the years, major improvements have been made to the FIB technique, and modern systems now typically use gallium ion FIB [44,45]. These systems also utilise the imaging capabilities of a scanning electron microscope coupled to the FIB system to carefully position the pore and track its growth during fabrication. The biggest disadvantages of this technique is that a FIB tool is extremely expensive (hundreds of thousands of dollars), and the whole process of mounting the chip in the instrument and making the pore takes several dozens of minutes, which is considered slow. Additionally, unless the system is modified even further (for example, by using low temperature helium ions), FIB can't easily make pores below roughly 15-20 nm [46,47].

As an alternative method, Dekker and co-workers created the transmission electron microscope approach [15], which is quite similar to FIB. Instead of a beam of ions, this technique uses lithography made by a beam of electrons from a TEM to etch a hole through the membrane (figure 2.8 c). Also similarly to FIB, this technique can fabricate pores at specific locations while carefully monitoring their size. However, TEM also has a very low throughput and the tool used costs in the millions.

As a way to solve the issues of both FIB and TEM, Tabard-Cossa *et al.* developped a way to fabricate a nanopore using a dielectric breakdown [16]. This method, called CBD,

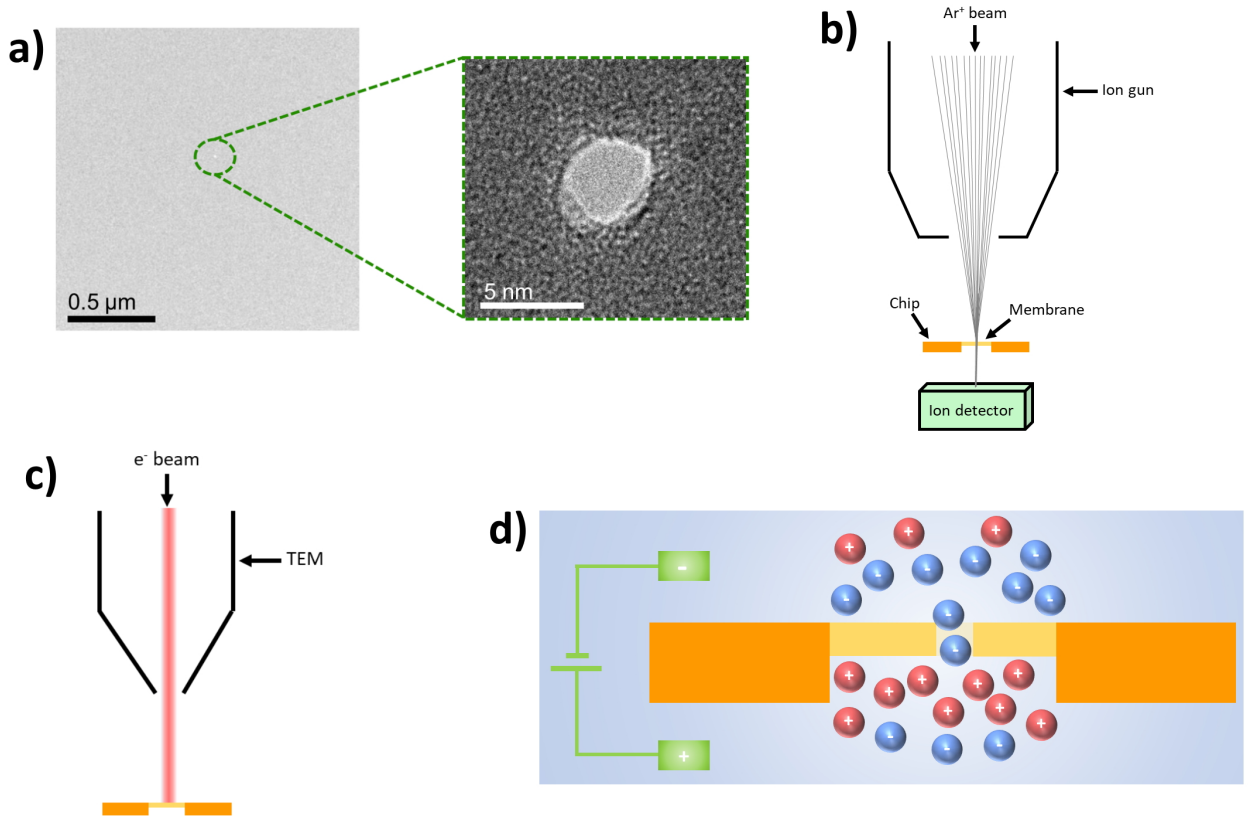

Figure 2.8: Solid-state nanopore fabrication techniques. **(a)** Image of a nanopore (fabricated by me) using a Transmission electron microscope. The pore has a diameter of roughly 5 nm. **b** FIB fabrication method. A beam of positive argon ions is shot onto the membrane via an ion gun, which mills a pore through the membrane. An ion detector is placed below the pore to track the formation process and growth. **(c)** TEM fabrication technique. A transmission electron microscope is used at high energy levels to shine a beam of electrons onto the membrane, which creates a pore via a dielectric breakdown. **(d)** CBD fabrication method. The membrane is submerged in an electrolyte solution, and a high voltage is applied through it. This results in an electric field which accumulates charges on both sides of the membrane, eventually leading to a breakdown of a tiny region of the membrane.

requires the membrane to be placed between two reservoirs filled with an electrolyte solution, the same way translocation experiments are conducted. Applying a high voltage (usually no less than 14 V) across the membrane induces accumulations of charges, and weaker, thinner or more conductive regions of the membrane will undergo a breakdown

first (figure 2.8 d). This results in a small pore in all of those regions. This technique is very cheap and can create pores within milliseconds, which is massive improvement compared to FIB and TEM. However, it has a major flaw: the pore can be formed anywhere on the membrane and at multiple spots at the same time. Since most experiments require that only a single pore be present in the membrane, this can become an issue. To combine the cheap and high throughput of CBD while also being able to position the pore and control the quantity made, our group developed the Tip-Controlled Local Breakdown approach [48, 49], which will be discussed in details in section 3.1.1.

Pros and Cons

The main strength of solid-state nanopores is the fact that they can be made at specific locations and with targeted diameters. Since they are made in inorganic materials, they are much more structurally solid, and are more mechanically, thermally and chemically resistant compared to their biological counter-parts. Depending on the technique used to fabricate them, solid-state nanopores are also a lot cheaper and easier to make. They can also be used in experiments that don't require sometimes complicated modifications to the running buffer.

However, solid-state nanopores have one major flaw: it is currently impossible to sequence DNA. Since inorganic and organic materials do not interact well, it is not possible to attach an enzyme to the pore to slow down the translocation of molecules and scan individual bases with high resolution. However, efforts have been made to either achieve sequencing or map specific regions of a given DNA sample using solid-state nanopores [24, 25], so there is still hope that nanopore-based sequencing could be performed using solid-state pores.

2.3 Nanopore Conditioning

2.3.1 Principle of Operation

Solid-state nanopores have their fair share of issues beyond just their inability to perform sequencing. Amongst others, pores made by certain fabrication techniques have been shown to become contaminated by residues and to shrink during experiments, making them quite unstable, and depending on the molecules analyzed, some can create clogging in the pore and block ions from flowing through indefinitely. As an attempt to solve these problems, Beamish *et al.* developped a process called conditioning where the pore is subjected to short voltage pulses in order to generate a high electric field at the pore that can clean its surface, make it grow to any desired diameter and remove clogged molecules [20].

More specifically, in this paper, the authors focus on the methodology of the conditioning process and showed the effect of a few conditioning parameters. In their experiments, they used 4-10 nm sized pores fabricated with TEM in 30 nm thick membranes and ran them using potassium chloride buffer.

2.3.2 Experimental Results

First, they found that generating electric fields between 0.15-0.3 V/nm inside of pores that were completely wet and free of any clogging worked well to condition the pores. For 30 nm thick membranes, this corresponds to an applied voltage between 4.5 V to 10 V, which they call wetting voltage, as opposed to the small voltage (200-400 mV) used to probe the conductance for size measurements, called measurement voltage. In their experiment, they used pulses of 200 ms of wetting voltage followed by 5 s of measurement voltage to track the size of the pore between each pulse. With this, they showed that pores could be enlarged to any size desired, very rapidly (within seconds or minutes) and with nanometer precision (see figure 2.9 a). The significance of this is the ability to use small

pores from any method of fabrication for experiments using small molecules like single-stranded DNA, or much bigger molecules like protein-ligand complexes with a simple conditioning step to modify their size. This also ensures a certain standard between pore sizes within the same experiment, which improves reproducibility.

Additionally, the authors demonstrated that conditioning can be used to wet pores and unclog them. They showed that after long applications of wetting voltage, an unwet pore, which is characterized by a conductance of up to two orders of magnitude smaller than expected, could become wet, and subsequently grown to a larger size, as shown in figure 2.9 b. Specifically, they created a 10 nm pore using TEM, which they imaged and confirmed to be 10 nm, and found that its conductance was that of a pore almost 100 times smaller, which they attributed to either residues from fabrication or air bubbles preventing the pore to be completely wet. After conditioning, they obtained a pore size of roughly 11 nm, which proved that conditioning helped to obtain the true conductance, although it did make the pore grow slightly. Also, the authors applied conditioning on freshly created and cleaned pores (figure 2.9 c) to grow them, and showed that the baseline noise was much lower following conditioning. Finally, they compared the frequency-based noise of clogged, unwet, cleaned and conditioned pores and showed that the noise of conditioned pores was significantly smaller (figure 2.9 d).

2.3.3 Possible Improvements

Since this paper is a proof-of-concept for conditioning, there are quite a few things to understand and to further study beyond what the authors covered. The most evident thing is the conditioning regimes and their associated voltages have been shown to work without having a solid understanding of why and without further experiments on other possible regimes. Also, pores fabricated using only one method have been studied, and it is not clear whether the observed behaviors would be present in pores produced using other methods. This could also be used to study whether a given method is actually causing additional, unintended damage during formation.

In-depth analysis, such as a theoretical or simulation-based approach, to understand the underlying principles effect of conditioning on pore growth remains to be done. In particular, many conditioning parameters can be varied, like the intensity and duration of the voltage pulse, but also the polarity and the shape (e.g. increasing slope, sinusoidal) of the pulse and the buffer in which the pore sits.

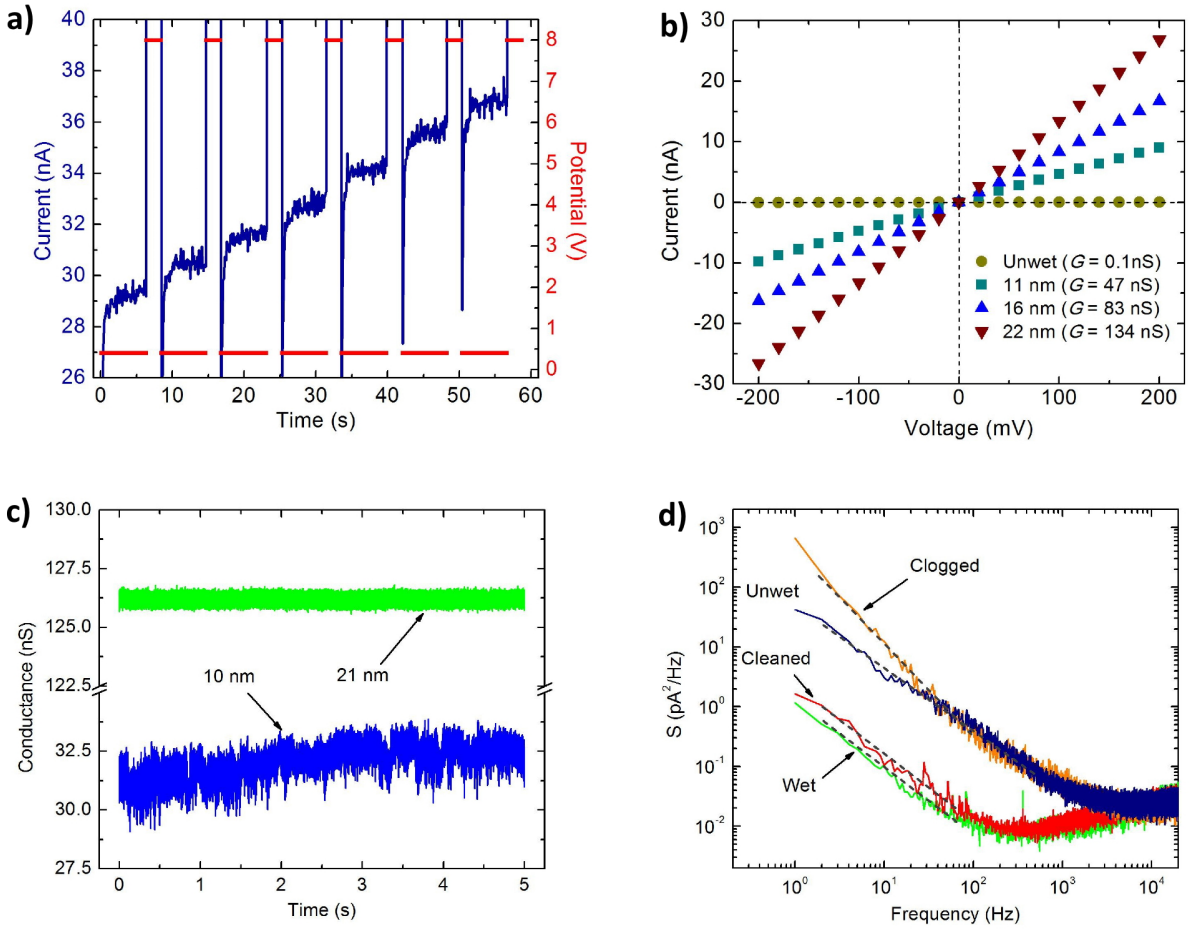

Figure 2.9: Results and analysis of conditioning on various pores. **(a)** Voltage applied (in red) for 2 s conditioning pulses at 8 V with 7 s measurement period at 400 mV between each pulse and resulting current (in blue). After each pulse, the current increases, indicating that the conductance increases and thus the pore size. **(b)** IV curves of a 10 nm pore at various stages of conditioning. The pore is initially unwet, as shown by the yellow line, which displays a conductance much lower than expected for a 10 nm pore. After conditioning, the pore becomes properly wet and displays a size of 11 nm, which is slightly larger than the original size. Subsequent conditioning is applied to grow the pore to 16 nm then 22 nm. **(c)** Baseline noise between a new cleaned pore (blue) and that same pore after using conditioning growth (green). The pore clearly becomes more stable after conditioning, but larger pores tend to be less noisy, so the effect of conditioning on baseline noise is difficult to isolate. **(d)** Power spectral density plot showing noise levels at different frequencies of a clogged, an unwet, a cleaned and a wet (conditioned) pore, with the conditioned pore having the least noise. Note that the behavior of the noise and its origin is briefly discussed in [20]. *Graphs taken from Beamish et al., 2013 [20].*

Chapter 3

EXPERIMENTAL METHODS

3.1 Basic Experiments

3.1.1 Nanopore Fabrication

One of the biggest advantages of solid-state nanopores is their ability to be fabricated rather than found as is in nature like biological nanopores. They can be fabricated in a multitude of materials, with any size desired and, depending on the technique used, at specifically chosen locations. Additionally, solid-state pores have the potential to be very inexpensive, since chips containing membranes can be fabricated in bulk with relative ease using standard materials (e.g. silicon, glass) and methods. However, as discussed in section 2.2.2, the three major basic fabrication techniques used in the field all have important flaws that take away some of the edge that solid-state pores have over biological pores. TEM and FIB are very slow and expensive and CBD has little to no control over the number of pores fabricated and their position. To make the most out of all the advantages that solid-state nanopores have to offer, our group has worked on a method to fabricate pores called *Tip-Controlled Local Breakdown* (TCLB). It relies on the precise positioning capabilities of an atomic force microscope (AFM), similar to that of TEM and FIB

while being much cheaper, to create a fast localized breakdown, like CBD, in the region in contact with the tip of the AFM.

TCLB Process

Compared to CBD, TCLB is known as a semi-wet breakdown technique, since only one side of the membrane is in contact with an electrolyte solution that carries current across the membrane. The other side is in a dry environment to allow the AFM tip to scan the chip and find the membrane. Specifically, the tips used are roughly 12 nm in radius (24 nm in diameter) and are made from doped diamond, which acts as a conductor (figure 3.2 c). The fabrication process (figure 3.1 a) goes as follows. First, a chip is mounted in the AFM and the scan process begins. Second, the AFM locates the membrane by scanning the chip in contact mode, where the tip is dragged along while remaining in constant contact with the surface. Once in contact with the membrane, a high negative voltage pulse is applied through the tip, which causes the breakdown across the membrane. Finally, the tip is retracted, leaving behind a nanopore.

During the pore formation process, the current across the pore is probed, an example of which is shown in figure 3.1 b. When the membrane is intact and no pore is formed (i.e. when the impedance of the membrane is very high), the ions from the electrolyte solution cannot flow through the membrane and create current, regardless of whether voltage is applied or not. Then, once a pore is made, the impedance of the membrane drops significantly (usually from dozens of gigaohms to megaohms), which lets current flow through, which can be seen by the sudden increase in the current magnitude. The membrane typically has an innate capacitance, which means that when voltage is applied through it, charges will accumulate in the membrane. Then, once the voltage application is halted, the membrane releases accumulated charges, which creates the RC decay.

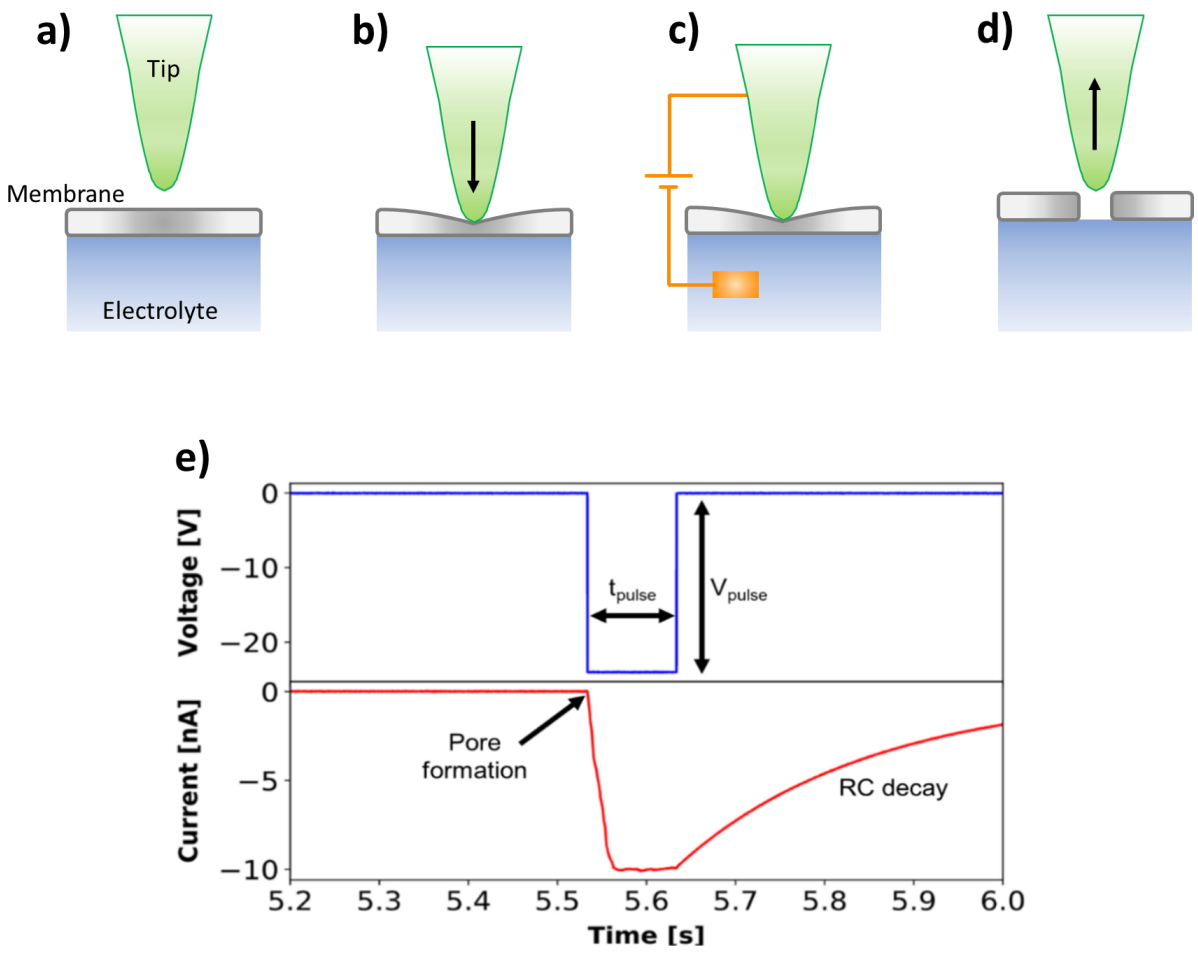

Figure 3.1: TCLB process and recorded data. The pore fabrication process using Tip-Controlled Local Breakdown goes as follow: **(a)** The chip, which sits above an electrolyte reservoir, is mounted in the AFM and the tip is positioned above the membrane. **(b)** The tip engages the scanning process, where it makes contact with the membrane and starts moving across while remaining in contact (this makes the membrane bend, but the force applied is never enough to break the membrane). **(c)** A negative voltage pulse is applied through the tip onto the membrane, with the electrolyte solution being in contact with the other end of the voltage supply via the current probe. This causes charges to accumulate a the region in contact with the tip, which induces a breakdown across the membrane, resulting in a nanopore. **(d)** The tip is extracted from the membrane without causing any physical damage. **(e)** Typical voltage and current signals of a TCLB pore fabrication obtained during one of my experiments. The precise behavior of the current during voltage application has yet to be analyzed in detail, but note that the *RC decay* once voltage stops being applied is due to the membrane capacitance.

TCLB Apparatus

In order to achieve TCLB, we use the apparatus shown in figure 3.2 d. A piece of polydimethylsiloxane (shortened to PDMS, which is a silicone polymer) with a hole punched through its center is glued to a metal disk, which forms what is referred to as a chuck, and the underside of the metal disk is covered with a sheet of mica so as to avoid any short-circuiting when the chuck is mounted in the AFM. This center of the PDMS is filled with an electrolyte solution, typically LiCl or sodium perchlorate ($NaClO_4$) such that it bulges outwards of the PDMS slightly at the top and touches the metal disk at the bottom. The backside of a chip (or sample), composed of a silicon nitride membrane (usually 0.01 mm by 0.01 mm in size and 12-30 nm in thickness) located on a silicon frame, is treated using oxygen plasma to make it hydrophilic. Then, the chip is placed on top of the chuck with its bottom side down, such that the hydrophilic part is facing the electrolyte and improves contact by pulling the electrolyte (through electrostatic forces).

The chuck with the chip on top is then mounted inside the headstage of the AFM, as shown in figure 3.2 a and b. A current probe records the current at the pore to record the formation process and sends its signal to an oscilloscope and then to a computer. That same computer communicates with the AFM controller module, which is the console that controls everything about the AFM. The computer sends commands to the controller module to move the AFM tip until the membrane is found, and once this is done, a script to apply the voltage pulse across the membrane and create the pore is initiated.

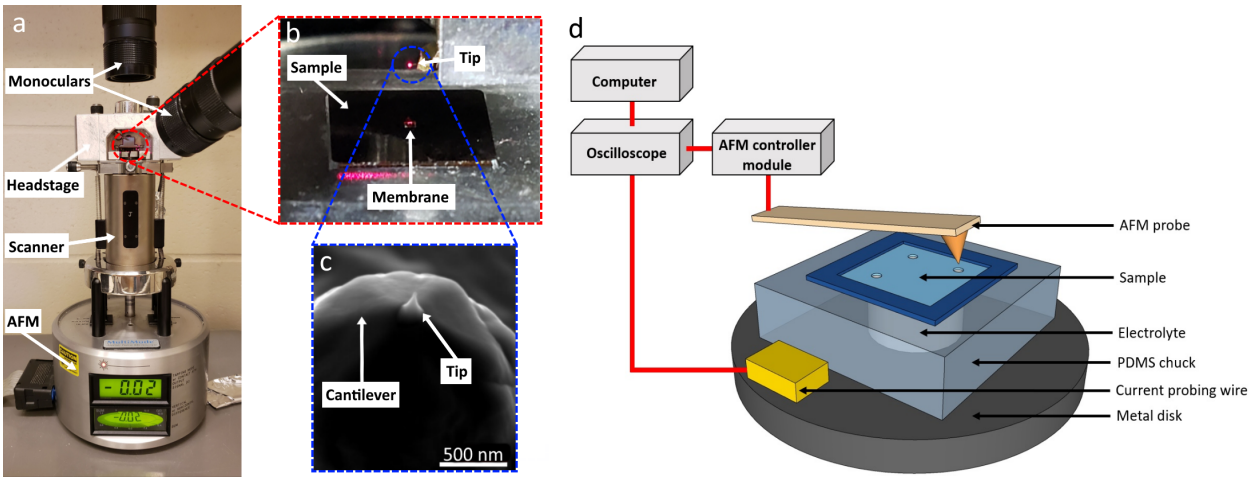

Figure 3.2: TCLB apparatus. (a) AFM setup used. The chip is loaded in the headstage and onto the scanner of the AFM. Monoculars are used to align the tip with the membrane on a macro level. **(b)** Zoomed-in view of the headstage. The tip rests above the membrane, which is located on the in the center of the sample, and comes in contact with it is engaged. **(c)** Scanning electron microscope (SEM) imaging of the cantilever, which contains the tip. **(d)** Sketch of the AFM setup and magnified view of the chip. The sample is placed such that the membrane sits on top of the electrolyte reservoir created by punching a hole through a sheet of PDMS. The PDMS is glued onto a metal disk, which ensures electrical contact between the electrolyte and the current probe. The tip scans the membrane, then applies a voltage pulse across the membrane, creating a pore. The current probe sends the breakdown signal to an oscilloscope, which relays it to a computer. The AFM setup is controlled with an AFM controller module, interfaced via a computer. Note that this schematic also highlights another benefit of TCLB, which is its ability to fabricate arrays of closely spaced nanopores on the same membrane and within seconds. This utilizes the imaging capabilities of the AFM to position the pores with nanometer precision. *Figure taken from St-Denis et al., 2019 [49].*

3.1.2 Conductance Measurements

Once a pore is fabricated, it can be used in experiments. The two most important experiments, conductance measurement and molecule translocation, require identical hardware and software, with the only difference being in the script executed in the software, which will be covered in section 3.1.2.

Physical Setup

The pore must first be placed in a chuck in order to run experiments. To do so, the pore is first briefly dipped in isopropyl alcohol (IPA) and dried using a wipe. This is to ensure the chip is clean and can bond easily to a pair of o-rings placed such that the chip is sandwiched between them. Then, the stack of o-rings is placed on one half of the chuck shown in figure 3.3 a, and the other half is screwed on top in a way to make sure it is tight enough to avoid leaks, while also not being so tight that the membrane cracks. The chuck has two holes to insert the screws as well as a central channel in a v-shape so as to flush out any large bubbles when the electrolyte is inserted. The electrolite solution used in all the experiments of this work is 2 molar lithium chloride, which provides good conductivity with reduced noise compared to other buffers such as potassium chloride, with 10 millimolar trisaminomethane (tris), used to keep the pH constant, and 1 millimolar ethylenediaminetetraacetic acid (EDTA), used to prevent degradation of DNA, both of which are added to improve stability in DNA translocation experiments. A top view of the filled chuck with the mounted chip inside is shown in figure 3.3 b.

Once the chuck has been set up, it can be connected to the hardware to start the experiments. To do so, a pair of electrodes made of silver wire soldered on a copper wire are inserted in the reservoirs. The electrodes must be submerged in bleach for several dozens of minutes to cover them in chloride, which is then exchanged with the electrolyte to generate and measure the current. The electrodes are connected to a measurement system known as "patch-clamp". The patch-clamp is used to probe the current at the pores using a voltage-clamp, which fixes the voltage and allows the current to fluctuate. This current

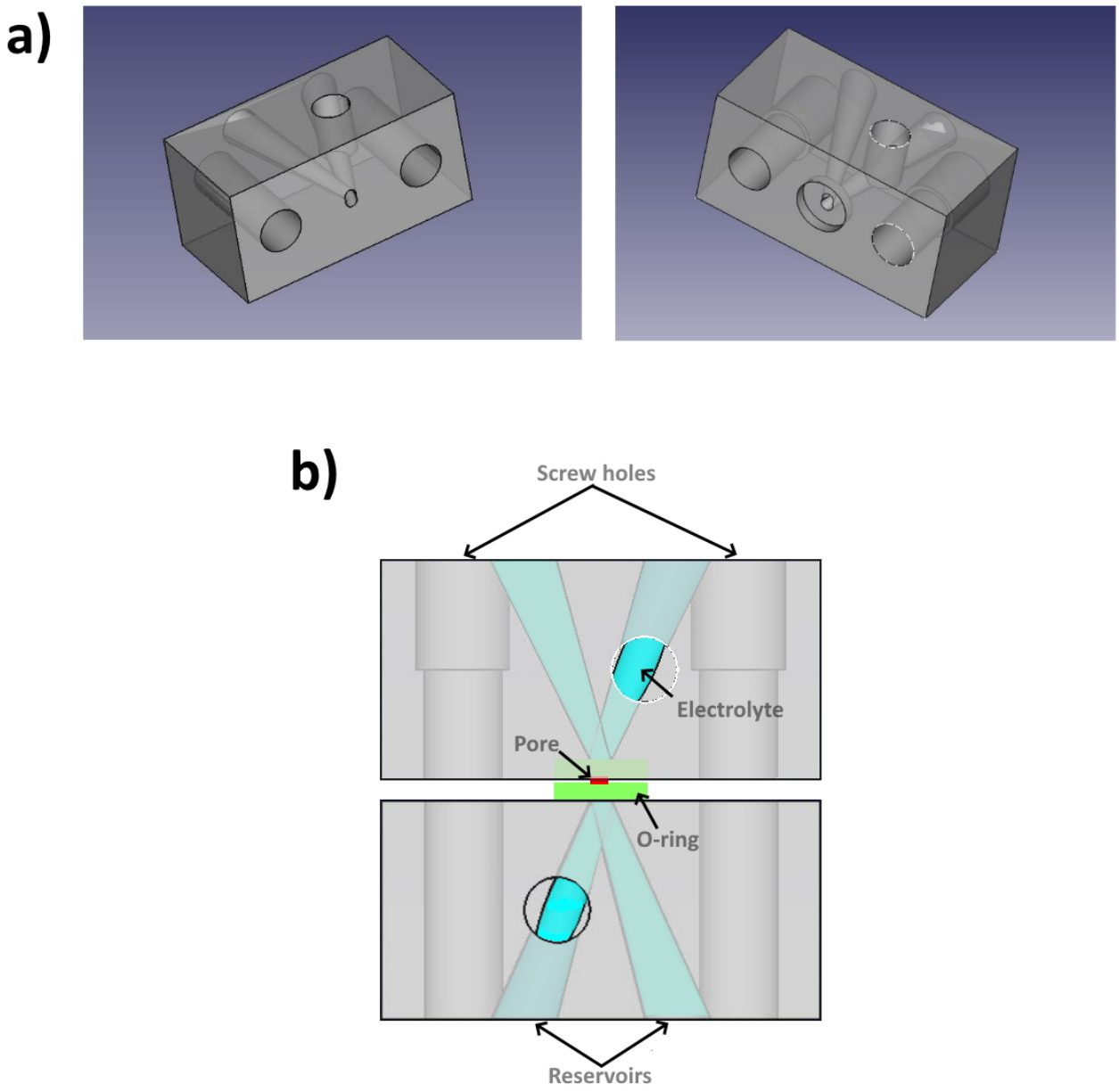

Figure 3.3: Chuck used for conductance measurements and translocation detection. (a) Side view of both parts of the chuck. **(b)** Top view of the chuck. Two screw holes are located on the edge of the chuck to clamp the chuck together and seal the o-ring (by pressure alone). The reservoirs shaped in V are filled with the electrolyte solution and meet at the center of the chip. The chip is sandwiched between o-rings, which are located in the center of the chuck. The outer ring of the o-rings is bigger than the diameter of the tip of the channels to avoid leakage. Holes at the extremities of the reservoirs are used to fill them and place wire electrodes, and holes at the top are used to place pellet electrodes (that do not fit in the other holes).

signal is then sent to a digitizer, which is used as a digital-to-analog converter to convert the data so that it can be accessed on a computer. The digitizer is also used as the voltage source providing the voltage at the pores (which is sent through the patch-clamp and the electrodes). A picture of the specific setup used in our experiments is provided in figure 3.4.

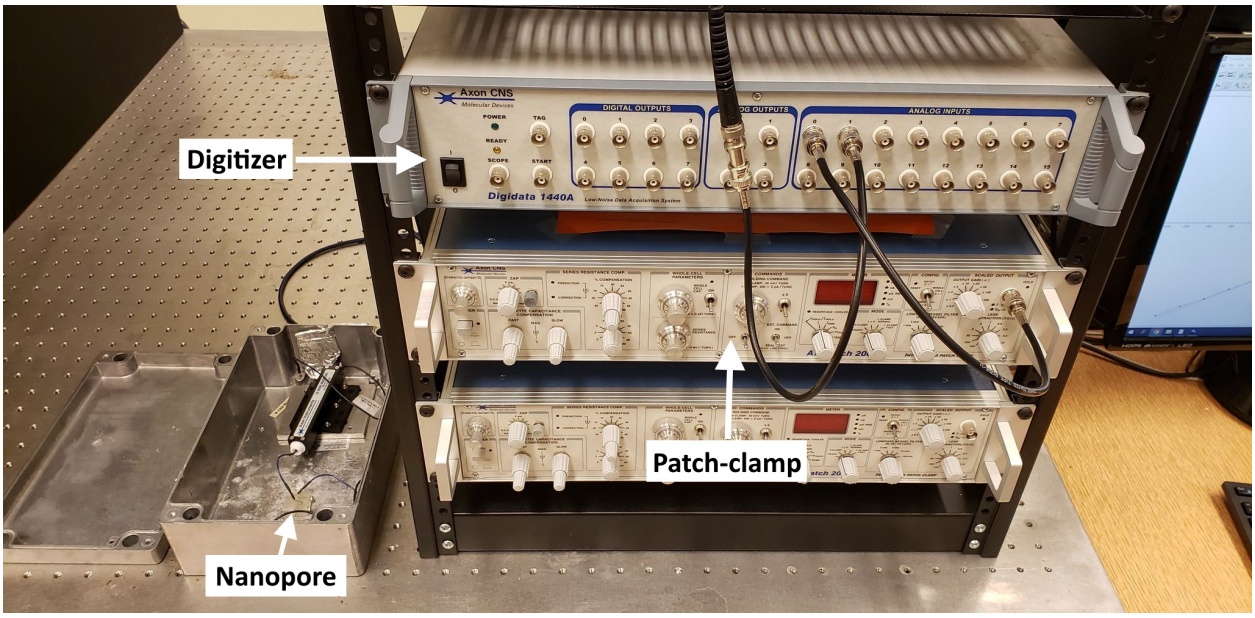

Figure 3.4: Physical setup used for conductance measurements and translocation detection. The chuck containing the nanopore is placed inside of a metal box to reduce noise and electrodes are placed in the reservoirs. The electrodes are connected to a patch-clamp system (Axopatch 200A), which reads the trans-pore current and sends it to a digitizer (Digidata 1440A), which transmits the data to a computer. The computer also controls the output voltage of the digitizer, which is applied via the same electrodes.

Measurement Software

The digitizer is accessed via a software called "Clampex" (figure 3.5 a). The software can both execute scripts or output fixed voltages to control the voltage sent to the pore as well as plot the resulting current signal in real time. For conductance measurements, the script used outputs a fixed voltage at 200 mV for 3 seconds and repeats this process by decreasing the voltage in steps of 20 mV until it reaches -200 mV (an example of the

signals generated by this script is shown in figure 3.5 a). This generates 21 current signals at different voltages, giving a very precise IV curve. For simple translocation experiments, the current is probed indefinitely at a fixed voltage, which is set on the left-hand side of the software.

Once a signal is acquired, it is then analyzed in a software called "Clampfit" (figure 3.5 b). This software first shows the same data from Clampex, but allows the user to place cursors and select specific parts of the data to be analyzed. As it can be see in the figure, where an example of a typical IV curve signal is shown, there is a important capacitive effect when a voltage is applied, which does not correlate with the actual pore conductance. As such, the cursors are always placed away from this capacitance transient (usually between 1 and 3 seconds of the start of the voltage application). Then, in the case of an IV signal, the software can average the current for each voltage line, generate an IV plot, perform a linear fit and give the slope of the curve, which corresponds directly to the nanopore conductance. In the case of translocation data, the software can also be used to apply a multitude of filters to mitigate the noise and clean up the data.

3.2 Conditioning

3.2.1 Hardware Setup

The setup used for conditioning experiments is very similar to that of conductance measurements (discussed in section 3.1.2). However, since the digitizer used can only output less than 1 V and the patch-clamp can only read signals around the same range, and that conditioning nanopores requires several Volts, both the digitizer and the patch-clamp are replaced by a Keithley 2400 sourcemeter, which can output several dozens of volts. However, the output voltage is much less stable than that of the digitizer, and its reading precision lacks compared to the patch-clamp, so the conditioning setup cannot be used reliably to perform precise experiments (like obtaining sub-nanometer precision from con-

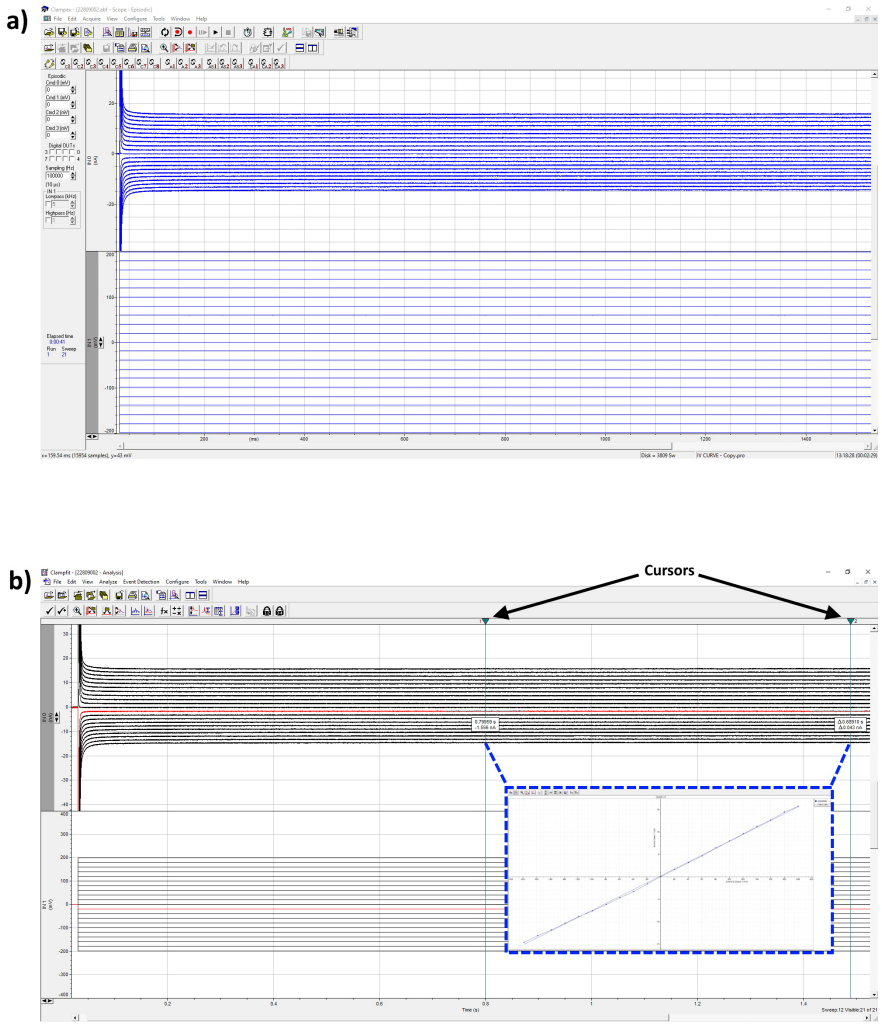

Figure 3.5: Example of an IV curve signal as acquired and analyzed in the software. **(a)** Example of a typical IV curve measurement in Clampex. On the left is where the digitizer's voltage output channels can be controlled (for example, during translocation experiments, where a fixed, constant voltage is required). On the right, a graph shows real-time current data as measured by the patch-clamp on the top and the output voltage sent to the pore at the bottom. **(b)** Example of the same IV curve signal as shown in Clampfit. The same current and output voltage signals shown in figure **(a)** are shown in the center of the software. Cursors can be placed to select specific parts of the data, and when doing so, an an accurate IV curve can be generated. The software can then apply a linear fit, which provides the conductance, as shown at the bottom right graph.

ductance or doing DNA translocation experiments). Nonetheless, it is a great tool for conditioning applications.

Like before, the chip is placed between PDMS o-rings, which are placed inside of a 3-D printed chuck. The chuck is filled with the same regular electrolyte solution (in our case, LiCl), and an electrode is placed in each reservoir. The electrodes are connected to wires, and the chuck is placed in a metal box to reduce noise. The wires are plugged in the Keithley device, which is connected to the computer where the conditioning script is executed and the data is recorded. A picture of the physical setup is provided in figure 3.6.

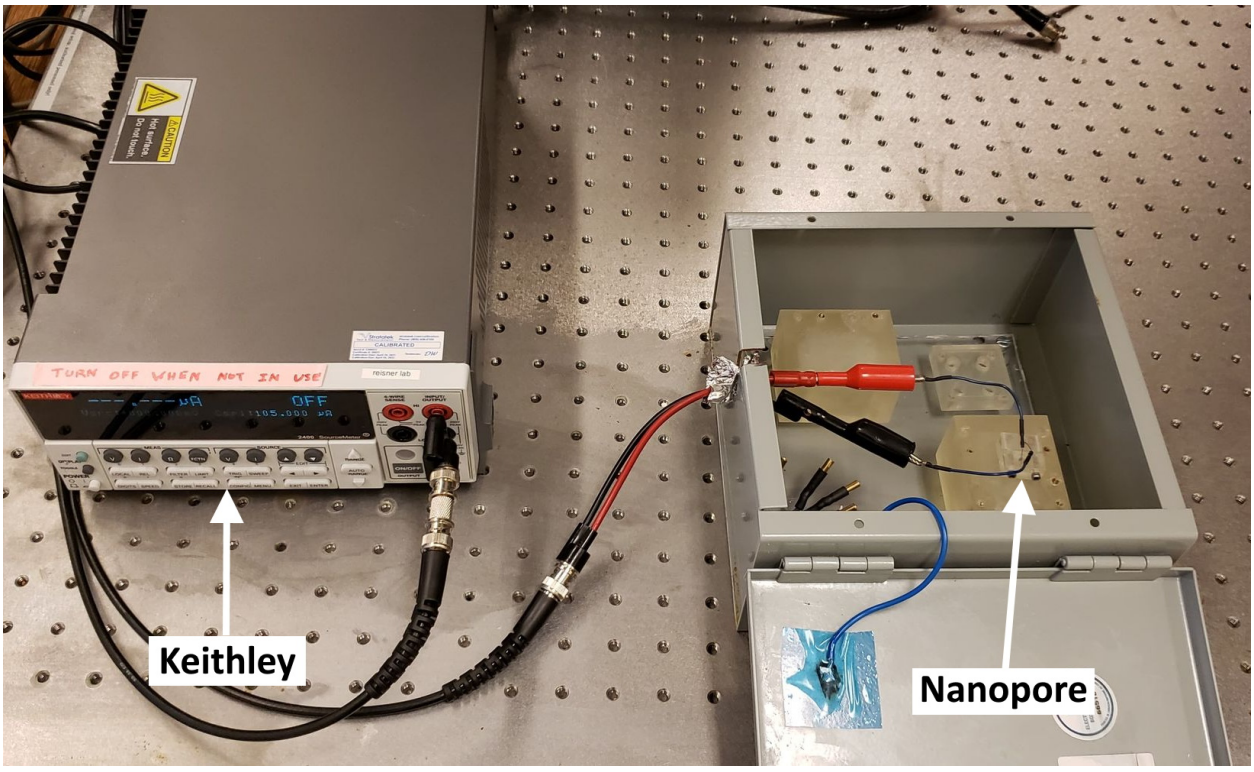

Figure 3.6: Conditioning setup. The chip is placed inside of a chuck filled with an electrolyte solution. The chuck is put in a metal box to reduce noise, and electrodes, which are connected to a 2-in-1 voltage source and scope Keithley device, are placed in the chuck. The Keithley device receives its command script from the computer on the right.

3.2.2 Conditioning Script

The script used to perform conditioning is based on the PyMeasure Python library [50], which creates an interface (figure 3.7 a) based on a code file written in Python. In this file, it is possible to program the Keithley to apply the voltage in virtually any way, based on a vast number of parameters. For the conditioning setup however, the only relevant scripts for the present study are the "IV curve" and the "Grow to dimension" scripts.

IV Curve

This code is important simply because of its practicality. Although IV curves generated with the Keithley device are less accurate and precise than with the conductance setup from section 3.1.2, they are still fairly accurate, usually being inaccurate in the computed size by less than 1 nm. For this reason, if one wants to estimate the size of the pore used (before, during or after conditioning), the IV curve script can be used instead of having to switch to the other setup.

Like before, the IV curve code tells the Keithley device to output -200 mV for 3 seconds and record the resulting voltage. Afterwards, it repeats the process by increasing the voltage in steps of 20 mV, until it reaches 200 mV. The script also takes the average of each current line for the last 2 seconds, and graphs it as a function of applied voltage in real time. Once the IV curve is done, the script uses the input parameters (solution conductivity and membrane thickness) and computes automatically the pore size. An example of the voltage output as a function of time is shown in figure 3.7 b.

Grow to Dimension

According to the basic study done on conditioning [20], the voltage used to condition pores should be in short pulses of high voltage. As such, we have written a simple script based on the observation of Beamish *et al.* and to reproduce their experiments. This script, which we call "Grow to dimension," allows the user to pick a target final diameter

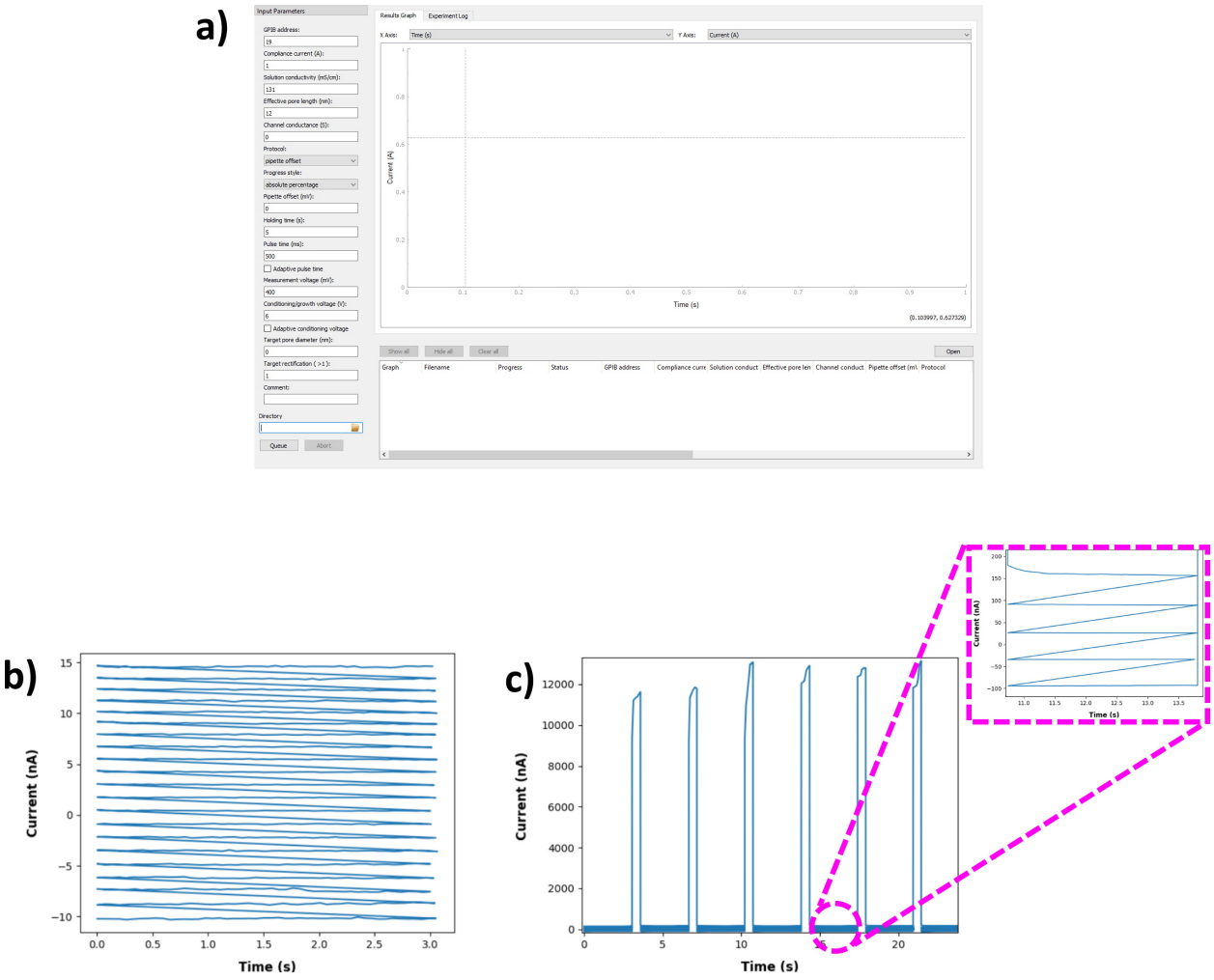

Figure 3.7: Conditioning script. **(a)** Interface of the conditioning software. On the left, a list of parameters used to compute pore size and to set the conditioning script can be modified to fit any desired experiment. The conditioning data is plotted in real time on the graph on the right, where the axes can be changed to show voltage vs. time or current vs. voltage. Below is a list of current and previous executions of the script. **(b)** Current vs. time graph for the IV curve script. The sweeps are superposed on top of each other for clarity. **(c)** Current vs. time of the grow to dimension with IV script. The script executes a full IV between each conditioning pulse (a zoom of one of those IV is shown under the graph) to carefully monitor changes in conductance and size.

of the pore post-conditioning, as well as set the pulse and measuring times and voltages. The script estimates pore size using a crude two-point IV (based on the measured current during the measurement voltage period and using the (0,0) point, assuming that there is

no voltage offset), and then repeatedly applies the voltage pulse until the pore grows to its desired size.

Since we want to focus on the effects of conditioning on pore size and geometry, it becomes crucial to keep track of conductance at low voltages in the most precise way possible, and the two-point IV approach described in the original conditioning paper does not suffice. As such, we have also implemented an improved script called "Grow to dimension with IV", that performs an IV curve of any number of sweeps set in the list of initial parameters. That way, every time a pulse is applied and the pore is slightly modified as a result, this modification can be tracked as accurately as possible. This script is shown in figure 3.7

3.3 Resensing

As discussed thoroughly in chapter 2, normal translocation experiments in solid-state nanopores are plagued with noise so high it is impossible to tell small features apart. The main cause of this noise is the high translocation velocities, but it is very difficult to fix in solid-state nanopores without introducing other substantial flaws. For instance, it is possible to reduce potential applied at the pore to reduce translocation speeds, but because of hydrodynamic drag, there has to be a minimal electric field generated in order to move the DNA through the liquid, which depends on the size and charge of the molecule. As it turns out, this minimum field has to be orders of magnitude larger than what is required to drastically reduce noise. Also, it is possible to modify the electrolyte solution used in order to reduce translocation speeds, but not by an amount significant enough to enable sequencing [51–53]. For that reason, some people have turned towards modifying the experiment as a whole rather than just its basic parameters to refine nanopore experiments, like coupling nanopore sensing with electrical tweezers [54] and fabricating devices containing two pores [24, 25].

One approach of interest is the concept of resensing, developed in 2009 by Marc Gershow and Jene Golovchenko [21], which focuses on improving the accuracy of measurements by probing each molecule many times. They showed that following a successful translocation, reversing the polarity of the driving voltage leads to a recapture of the same molecule, and doing so many times generates several signals for each individual molecule. However, the system they used was very rudimentary in that it would only look for a translocation, and then periodically flip the voltage based on a fixed delay, hoping it wouldn't be too short for the molecule to stay inside the pore or too long for the molecule to escape the capture radius. Also, the authors allude to the benefits of obtaining many blockade signals for each molecule for the purpose of ameliorating data analysis, yet they do not present evidence for it and instead focus solely on demonstrating that they successfully accomplished resensing. An interesting possibility that has yet to be studied is the potential for partially or completely averaging out specific sources of noise like that of Brownian motion; that is, separating the blockade signal of the molecule, which should be fairly consistent across multiple resensings, from background noise.

For that reason, I have decided to reproduce their experiment, while also improving the logic of the system to drastically increase recapture rate, in hopes of later using the data collected to develop a better analysis method. To accomplish this, I have used the same setup used in regular experiments conductance measurements (described in section 3.1.2), and I have combined it with a field-programmable gate array (FPGA) system that provides the logic required to read the translocation data and control the polarity of the driving voltage. The general logic of the code and the specific implementation of it on the FPGA software will be discussed in section 3.3.2, but before that, we will begin with an introduction to the FPGA software: LabVIEW FPGA.

3.3.1 LabVIEW FPGA

Laboratory Virtual Instrument Engineering Workbench (LabVIEW) is a programming language based on visual representations of the code rather than being text-based. In

particular, LabVIEW FPGA is a language compatible with a wide range of FPGA devices which provides an interface for the user to interact with the FPGA without having to program the FPGA from the ground up (typically using assembly). In this section, we will cover the basics on how to get started with LabVIEW FPGA, as well as very simple functions and implementations that will be used in the final resensing code. The intention of this section is to provide a guide for current and future group members to rapidly be able to modify the resensing code that I wrote to fit their needs as well as write their own from scratch if necessary.

LabVIEW Setup

To write and execute a code on the FPGA, it is required to have a project that targets the right FPGA device and that includes the code to run. To do so, one must first launch the LabVIEW software and create a new blank project. Projects in LabVIEW are the equivalent of directories in that they provide a place to store different codes, and have those codes interact together if need be. Then, the FPGA device (also called FPGA target) needs to be added to the project under "My Computer" to let the code know which device it should be compiled on. FPGAs usually have a couple of connectors, each with several pins representing different input and output channels and grounds, but usually one connector is used at a time. The FPGA used for the experiments presented in this work has three connectors, with "Connector0" being used as it is the only one with both input and output channels. Thus, "Connector0" needs to be added to the project, so we can later reference its channel in the code.

With this setup complete, the code can now be created under the FPGA target. LabVIEW uses "Virtual Instrument", or VI, codes, which are made of a block diagram, where all the code is written, and a front panel, which acts as the interface that the user can interact with to change parameters in the code while it is running (e.g. increase or decrease output voltage or modify a delay). Once the code has been written, it can be compiled onto the FPGA, and depending on which type of execution mode used, it can either be

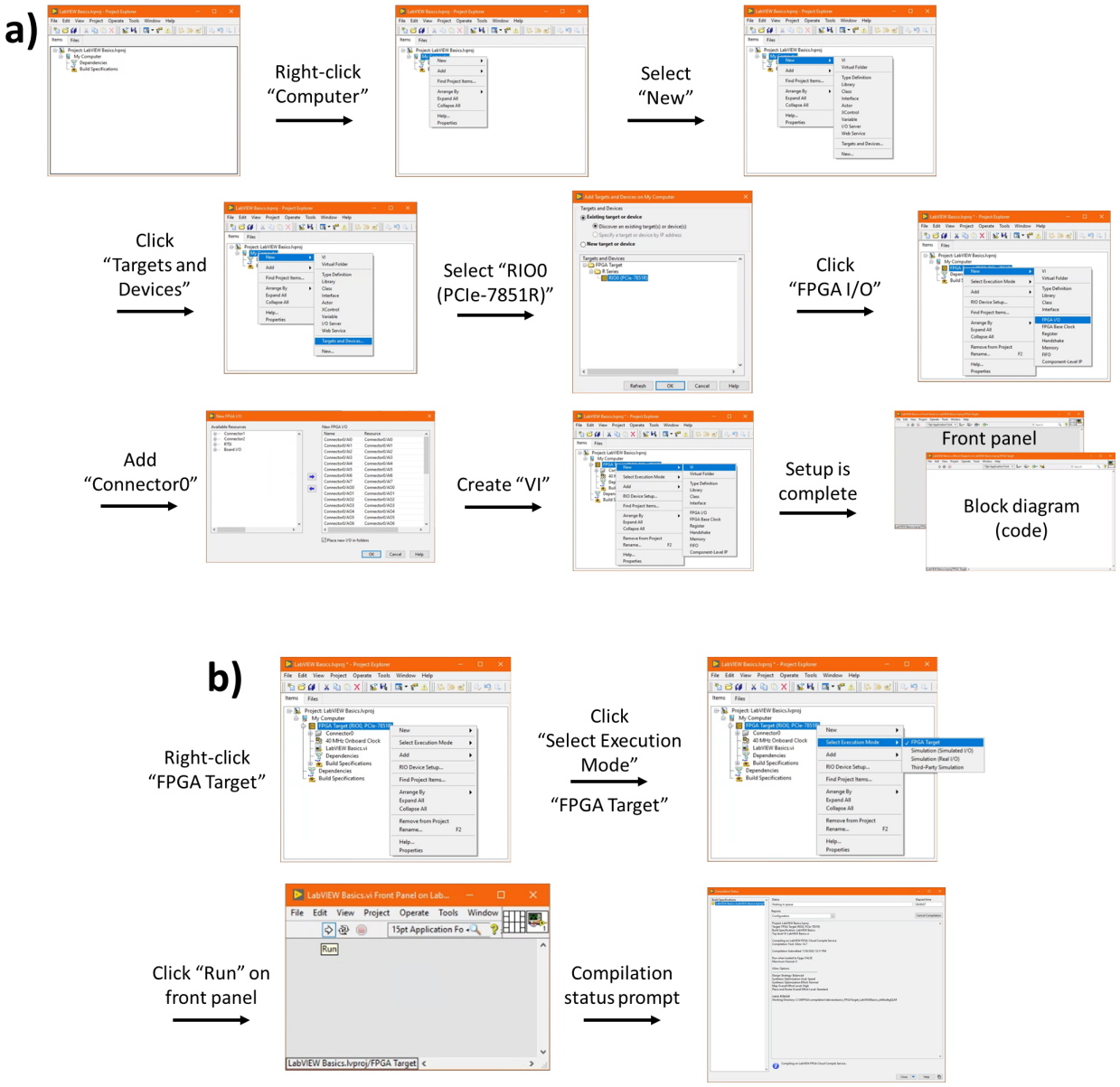

Figure 3.8: LabVIEW Setup. **(a)** List of actions required to write a code. First, a project must be created and launched in LabVIEW. Then, the FPGA must be added to the project, along with its connectors. Finally, the project (called "VI") can be created, which is comprised of the front panel and the block diagram. The front panel is an interface for users to interact with when the code is running, and the block diagram is where all the code is written. **(b)** Code compilation on the FPGA. To run the code on the FPGA, it must first be compiled onto it. To do so, the project must be set to be executed on the FPGA, and then the code can be run and uploaded onto the device via an online compiling server.

ran as a simulation or uploaded to the real FPGA target. Selecting the latter will compile the code onto the device by using an online compiling server provided by the FPGA company (National Instruments). After the compilation is over, the code will be in memory and can be executed anytime. A visual flowchart of a more detailed, step-by-step guide to get LabVIEW running is presented in figure 3.8.

Basics LabVIEW Functions

Once the VI has been created, the code can now be written in the block diagram using the various functions provided. Those functions act similarly to that of other programming languages, but like mentioned before, they are represented with icons rather than words. A list of icons of all the functions and structures that will be described below (and that are part of the resensing code) is offered in figure 3.9.

In LabVIEW, the way to apply a function to a certain variable is by using what is referred to as *terminals*. All functions, constants, variables and so on have input and/or output terminals. Input terminals are used in functions that perform a certain action on given variables (for example, a square function will square whatever it receives to its input terminal, and an addition function will add the each variable received to its two input terminals) and output terminals are where the results of functions can be accessed as well as where variables can be referred to. Terminals communicate between each other via wires, which have different colors based on which kind of value they refer to, with blue and orange wires referring to numbers (16-bits word and 32-bits single-precision floating-point) and green referring to booleans.

The most basic numeric functions are *Add*, *Subtract*, *Multiply* and *Divide*, which are self-explanatory, and they all have two input terminals and one output. *Increment* simply adds one to its input, *Invert* changes the sign of the input and *Square* raises the input to the second power. *Remainder* is used as a modulo operator, where one input is the variable and another input is a numeric value used as the divisor, and the outputs are the quotient

and the remainder. A more specific implementation of the *Remainder* function will be explained in section 3.3.1.

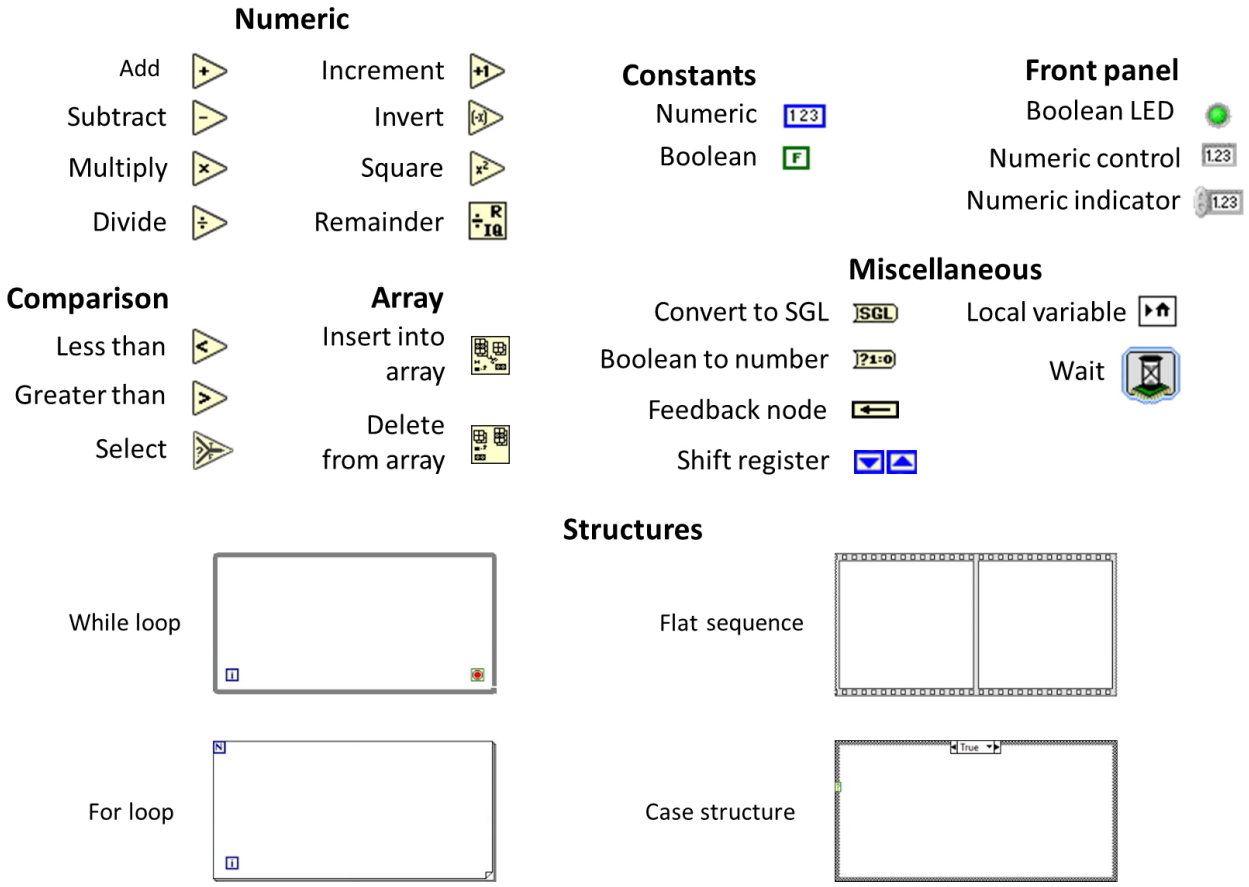

Figure 3.9: List of LabVIEW icons. This is a list of all the LabVIEW functions and structures used in the final resensing code. They are separated in their respective categories (numeric, constants, comparison, array, front panel and miscellaneous). All of them refer to their icons in the block diagram, except for those under the "Front panel" category.

Comparison functions used are the *Less than* and *Greater than* functions as well as *Select*. Select receives two inputs, one at the top and one at the bottom, and outputs one of those inputs based on a third boolean selection (true outputs the top input and false the bottom input).

To build and manipulate arrays, the two functions used are *Insert into array* and *Delete from array*. The former starts with an initialized array (an array of fixed size filled with zeros), adds to this array a data input at a given position and outputs the new array. The

latter removes a subarray of a given size and at a given position from the main array and outputs the shrunk array.

Constants are also very simple. The first type is *Numeric* constant, which are simply fixed size number outputs that can have different representations (like word or single-precision float). They are used for referring specific positions, resetting implementations like counters to zero or setting them to specific values and so on. The first type is *Boolean* constant, which is either true or false, and behaves exactly as expected.

Other miscellaneous functions used in the code on the front panel include *Convert to SGL*, which is used to convert numbers from word representation (mostly input voltage signals). This is important when modifying numbers using functions, for instance when taking a large sum, that would exceed the 16-bit limit of word representations. *Boolean to number* is another converter function that transforms a boolean input into a binary output. The *Feedback node* and *Shift register* functions serve to transfer information from one iteration of a loop to the next. For instance, if a certain result computed at the end of a while loop has to be reused at the start of that same while loop in the next iteration, a shift register will be used to carry that information. The only difference between the two functions is that one is inside of the loop and the other sits on the border at the end of the loop and warps back at the start of it. A *Local variable* is used as a shortcut to refer to another variable to replace long wires and tidy up the code. Finally, *Wait* is simply a delay function that pauses the code for a duration equal to a chosen input connected to it in microseconds.

Structures are environments where code is executed in a specific way. The most basic one is a *While loop*, which contains a stop function (called "loop condition") on the bottom right that has to be connected to a false boolean constant for the loop to run and an iteration counter on the bottom right. Since we want the code to run indefinitely, a while loop has to be used to encompass the rest of the code. *For loops* work the same way than with other coding languages, where it iterates for a set amount of loops at the top right, or if an array is added to the for loop, the loop can be set for each element in the array

by changing the entry point of the array in the loop (more on that in section 3.3.1). The FPGA will always try to execute everything in the code simultaneously in parallel, but if certain parts are required to run in an order, then those parts have to be placed inside of a *Flat sequence*, where the code will be executed from left to right. A *Case structure* is the equivalent of an "if/else" statement, where the input dictates which case to execute. It can take as an input either a boolean constant, where the case structure will have two cases, or a numerical constant, with each number having their associated case to refer to.

On the front panel, only three functions are ever used. The first one is called *Boolean LED* and is a way to control a specific boolean constant. For instance, if a while loop has to be manually stopped, one can connect an LED to the loop condition and change the state of the LED to stop or restart the loop. The two final ones, *Numeric control* and *Numeric indicator*, are input and output numeric constants respectively. The first one serves to adjust specific numbers in the code while it is running, while the second shows on the front panel the number that the indicator is connected to (it is mostly only useful for troubleshooting).

Simple LabVIEW Implementations

With basic functions and structures covered, we are ready to look at a few simple implementations combining some functions that are crucial in the resensing code.

The first implementation revolves around making a counter. More complicated versions of this involve where the counter resets following a specific action for example, but the fundamentals remain the same as in figure 3.10. The code lies in a while loop, with a boolean false constant linked to the loop condition so that it runs indefinitely. The counter itself is simply made of a numeric constant acting as the initilized value of the counter (if the counter has to start at a specific value, then the numeric constant can be changed to fit this value), which sits outside the loop and is thus only referred to once, and an increment function connected to a shift register. For the first loop, the value of the counter starts at 0, then during the loop it becomes 1, then it reaches the shift registers, which warps the

value back to the start of the structure for the second loop. Then, for the second loop, the counter starts at 1, is increased to 2 and is stored and so on. Note that in this simple implementation, the counter taken after the increment function and the output of the loop iteration have the same value.

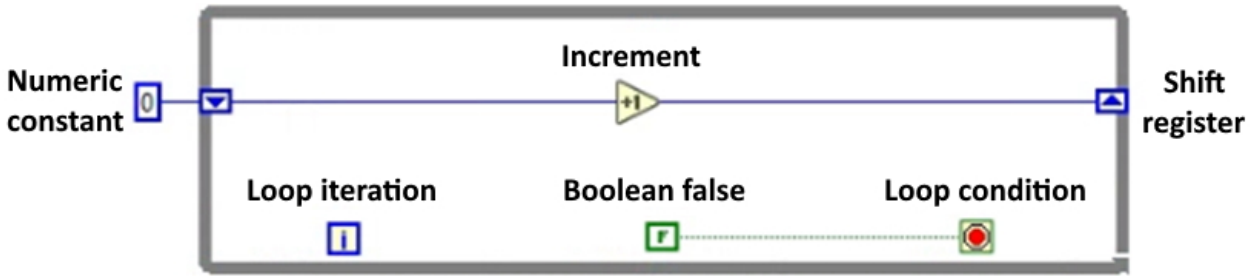

Figure 3.10: LabVIEW counter. Pictured is the schematic of a counter written in LabVIEW. An increment function is connected to a shift register inside a while loop such that the increment function takes what was stored in the shift register during the previous loop, adds 1 to it and stores it again in the shift register.

A common way to use a counter is with the remainder function, as shown in figure 3.11. In this case, the code sits again in a while loop with a false boolean constant such that it is repeated indefinitely, and a remainder function is added to the counter, with its output also connected to a case structure. The remainder function is connected to a divisor (in this case 4), which makes the output of the remainder go from 0 to 3 and then back to 0. The remainder dictates which case structure is selected, and here there would be 4 cases in the structure, since there are 4 possible inputs. So, for the first loop, it would be in case 1, then case 2, then case 3, then case 4 and back to case 1 during the fifth loop.

As it is the case with all programming languages, arrays are a fundamental way of storing and manipulating data. When using an FPGA though, it gets a bit more complicated as the array size must always be defined and remain constant from loop to loop (otherwise the FPGA won't be able to execute the code). Therefore, FPGAs can only use "shifting arrays," which are arrays of a fixed size that move along with the data, an example of which is shown in figure 3.12. The code uses an initialized array full of 0s and with a size fixed in the front panel. When an iteration of the loop is performed, the delete

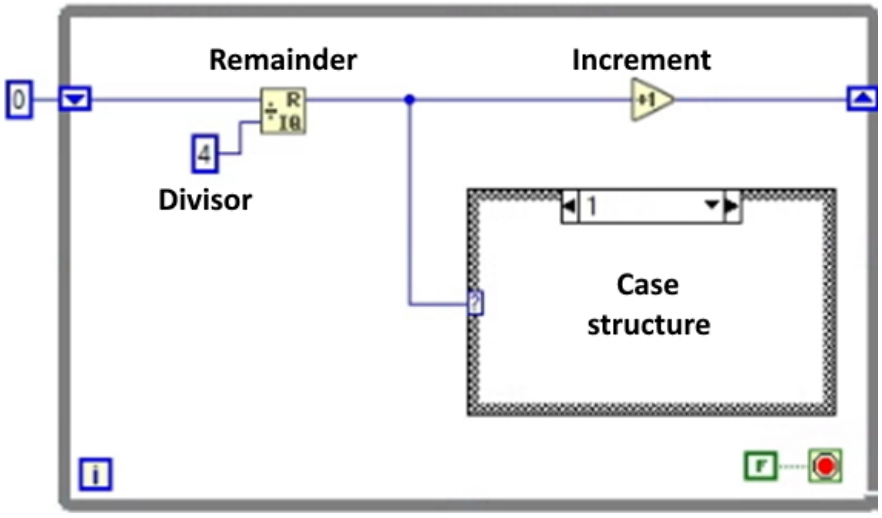

Figure 3.11: LabVIEW Remainder function. Schematics combining a counter and a remainder function. The remainder is connected to a divisor of value 4 and to an increment function. This means that the output of the remainder increases by 1 each loop, but resets to 0 every time it reaches 3. A case structure is connected to the output of the remainder, which controls which case will be executed for each loop.

from array function first takes an array of size 1 at position 0 and removes it from the array in a way that shifts all the remaining data down by one position. Then, the insert into array function takes the new data point, adds it on top of the array and outputs this new array. It also stores this array into the feedback node, which will then feed this new array back into the delete from array function in the next loop. So, if the array has a size of 5 with elements occupying positions 0 through 4, positions 0 through 3 will always be occupied, and the delete from array function creates an empty place at position 4, which insert into array fills right back with the new data point. This means that every new data point replaces the oldest data in the array, which shifts the array along with each new point.

With an array in hand, it becomes possible to perform specific computations on the array as a whole. One of those computations is simply to calculate the mean of the array (figure 3.13). To do so, a for loop is required, with the input array sitting outside. when connecting the array to something inside of the loop, the tunneling mode must be

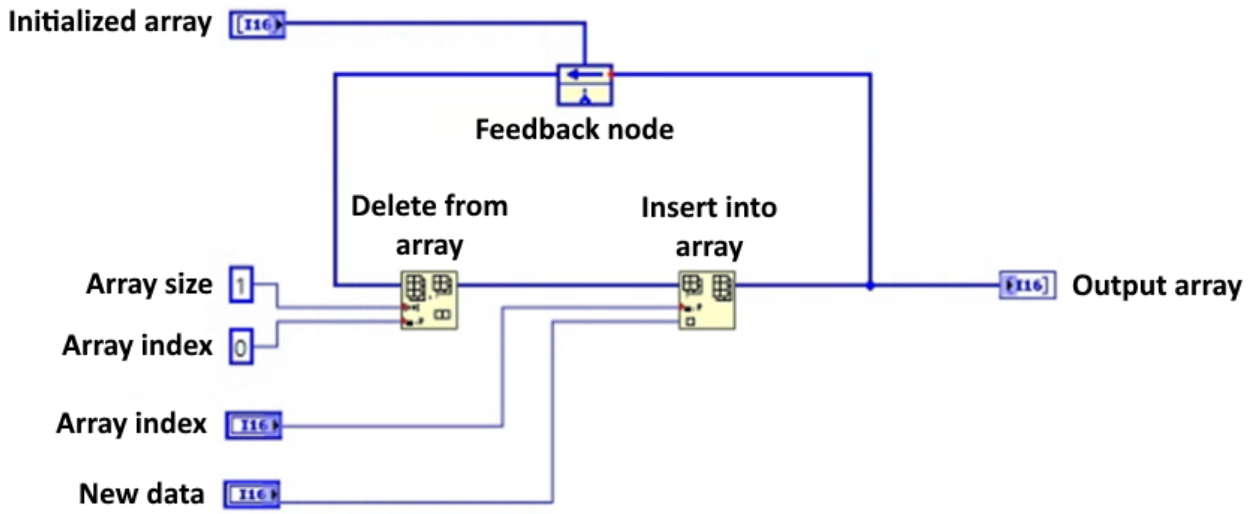

Figure 3.12: Shifting array in LabVIEW. Shown here is the simplest code required to create a shifting array. This array has a fixed size and each loop, a new data point is added at the top of the array, and the oldest point is removed from the array, pushing everything down by one position.

changed to "auto tunneling index", which essentially fixes the number of iterations of the loop as well as which element to take at any given loop based on the array. So during loop 1, the first element will be added to the sum, during loop 2, the second element and so on. The array is then connected to the add function, which is connected back into itself with a feedback node (initialized with a 0). This allows the for loop to add individual data at any position to the combined sum of the data at previous positions. Then, once the sum is computed, a divide function is used with the array size to obtain the mean.

Another important thing that requires both an array and its mean is computing the variance. Similarly to the mean computation, the code sits in a for loop, with the input array connected to the loop via auto tunneling index. For simplicity, the code in figure 3.14 assumes the mean has already been computed and is given as a fixed number. To compute the variance, one must add the difference squared between each data point and the mean of the array, and divide by the array size. Thus, in the code, the mean and each individual point are subtracted from one another using the subtract function, then squared using the square function, and finally added to the sum of the computed squared differences,

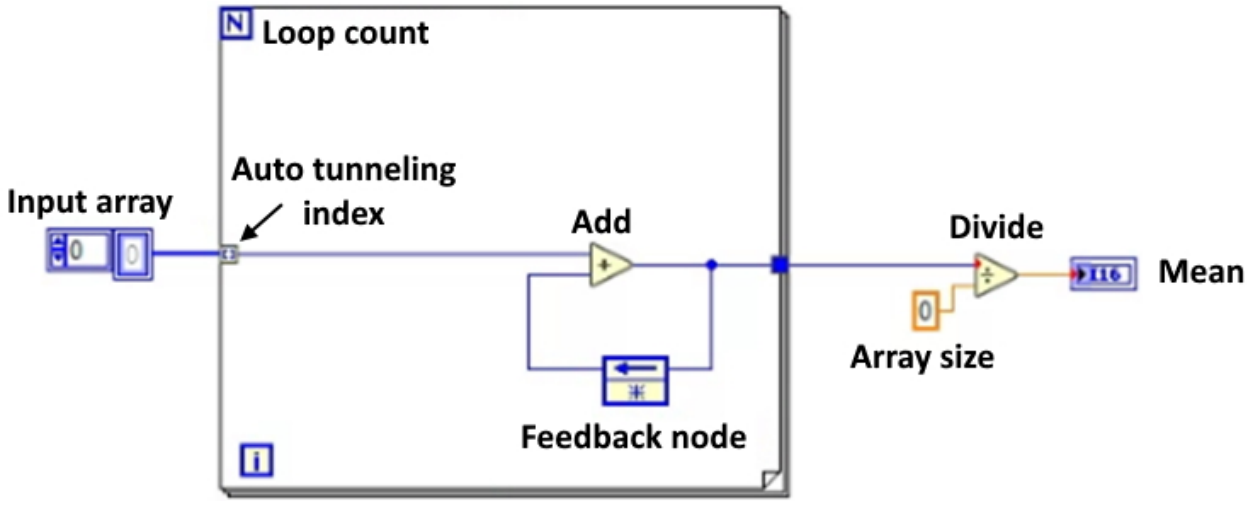

Figure 3.13: Mean computation in LabVIEW. To compute the mean of an array, said array must be connected to a for loop using an auto tunneling index. The for loop adds all the values inside of the array, then a divide function is used to divide the sum by the array size.

which is stored in a feedback node. Again, once the full sum has been computed, it is divided by the array size to give the variance. Note that the variance is the square of the standard deviation, which can be used to find outlier points (i.e. points that are offset from the mean by a significant margin, like blockade current compared to open-pore current). However, since squaring a number is orders of magnitude faster for the FPGA than computing the square root of a number, the resensing code will only use the variance to detect outliers.

3.3.2 Resensing Code

Code Logic

Before we can look at how all the functions, structures and implementations are used together to create the resensing code, we must first understand the logic required from said code to catch translocations while distinguishing them from noise, reverse the flow of DNA, catch the re-translocation and repeat the process over and over. To accomplish the resensing process, the code relies on an algorithm known as local peak detection. In short,

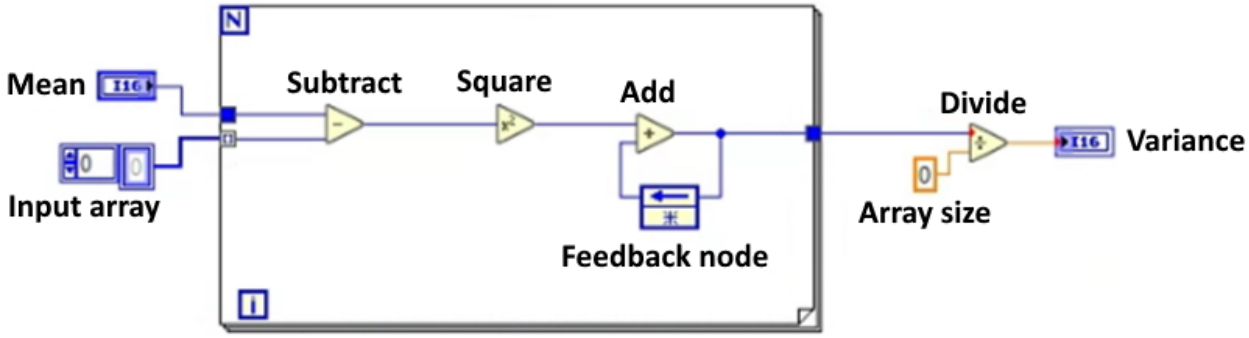

Figure 3.14: Variance computation in LabVIEW. Given an array and its computed mean, a for loop can compute the difference of a point with the mean, square it and add it to the squares of all the other differences between the mean and each data point in the array. Dividing this sum by the array size gives the variance of the array.

the algorithm scans locally for a data point that is an outlier (beyond a fixed threshold) compared to the other data points, and triggers a response when such an outlier is found.

More specifically, the code starts by generating what is known as a shifting array. Since the baseline noise of nanopore experiments tend to shift by several nanoamps over time, and a typical current blockade is on the order of 0.1 nA, taking into consideration data points across several minutes invalidates the data. For this reason, a shifting array, like the one shown in section 3.3.1, is required. The array has a fixed size determined before running the code and keeps this size constant throughout, but constantly replaces old data points with new data, such that it *shifts* over time. When the array is full, its mean and variance are computed. As mentioned before, the code considers only the variance because the processing power required to perform square roots and obtain standard deviations is too costly. However, since it is standard to discuss outliers in terms of number of standard deviations, or sigmas, off from the mean, we will use those terms moving forwards.

Therefore, once the standard deviation has been computed, a new data point is acquired, and it is compared to the set number of standard deviations which has to be chosen beforehand. This number can be quite tricky to determine, because choosing a number too low will cause the code to trigger on false positive (interprets noise as translo-

cation events) and choosing a number too high will make the code miss translocation events. With the threshold carefully chosen, the data is compared to the standard deviation. If it is within the set number of sigmas, the oldest data point is removed from the array and the new one is added. If the point is outside, this means that there is a translocation event occurring, and a new loop begins.

A good way to increase the number of successful resensing events is by having the delay used before reversing the voltage polarity start after the translocation is complete (and not as it starts, like it was done in Gershow and Golovchenko's paper [21]). This is especially useful when using a solution that has molecules of different sizes. As such, once a translocation event has been successfully identified, the code stores the threshold and starts taking new data points. These data points are then compared with the threshold, and as long as they remain outside of it (that is, as long as the translocation is still happening), new data points keep being acquired. However, once a point is back within the threshold, the code stops taking in new data.

Then, a fixed delay is waited to allow the molecule to move away from the pore. This is because solid-state nanopores tend to be very capacitive, so rapidly inverting the polarity creates a huge capacitance transient. If a translocation happens during this transient, the data becomes corrupted by the transient and no useful information can be extracted. Thus, the delay needs to be long enough such that the translocation backwards happens when the capacitance transient is gone. However, the delay must not be too long, because if the molecule moves out of the capture radius of the pore, it will be lost. Hence, once the right delay is waited, the code inverts the polarity of the voltage at the pore, waits some more time to avoid taking data points during the transient (since it would be irrelevant anyways) and the whole process begins again. A flowchart explaining this process as well as a step-by-step application of this algorithm to actual data are provided in figures 3.15 and 3.16 respectively.

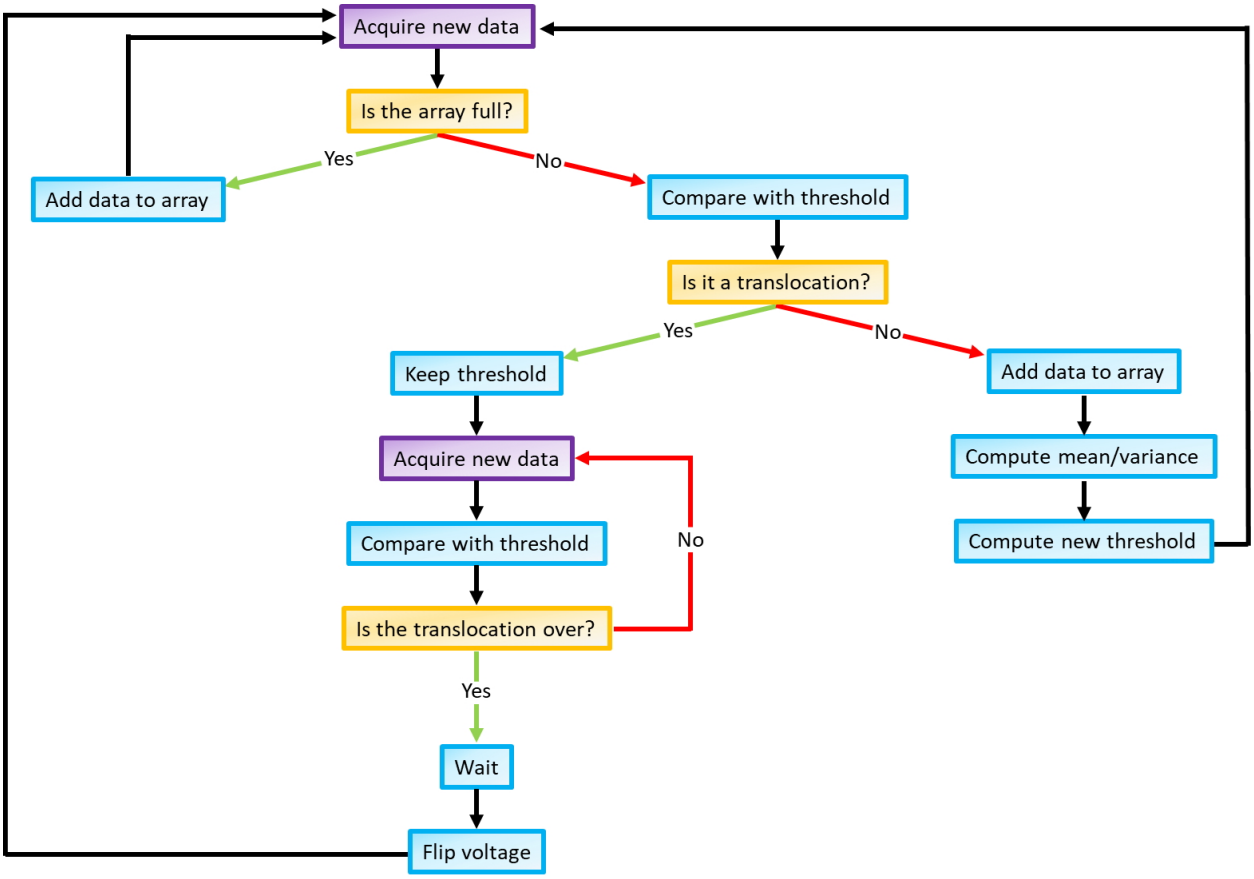

Figure 3.15: Flowchart of the FPGA code. The code starts by acquiring a new data point and verifying if the array is full. If it isn't, the data point is added to the array and the loop ends. This is to avoid pointless computations while a valid mean and variance can be computed on a full array. Once the array is full, the data is compared with the threshold. Note that the very first loop after the array fills up, the comparison with the threshold is not made because the threshold has yet to be computed. If the data is inside the threshold, it means there is no translocation event; otherwise, it means a translocation started. The threshold is then stored in memory, because once translocation ends, the current will go back to what it was before. Thus, new data points are compared with the old threshold, and as long as the translocation is not over, new data keeps being acquired. Finally, once the translocation is over, a delay is set, the polarity of the voltage is reversed, and the whole process begins again.

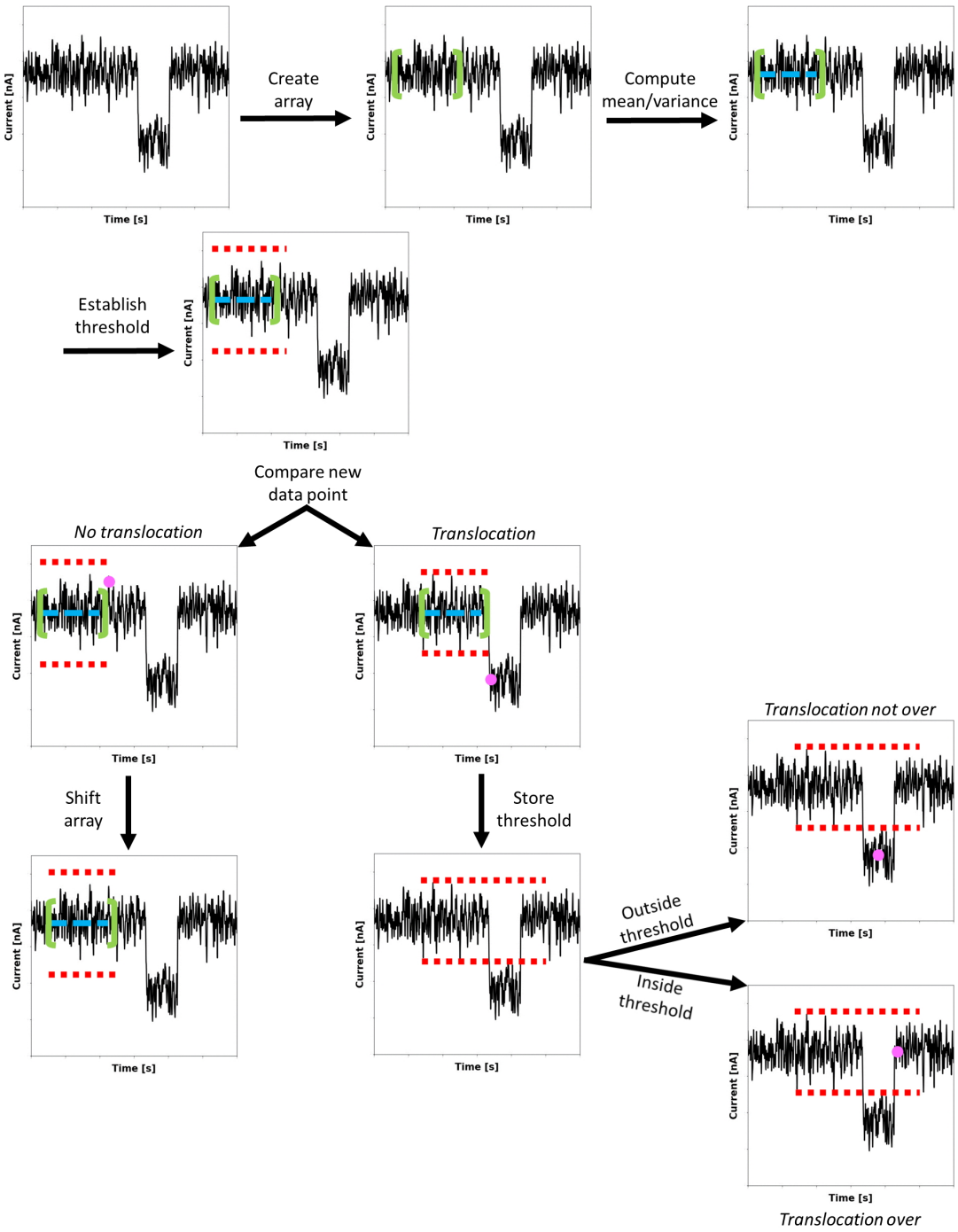

Figure 3.16: Resensing code algorithm with data. Flowchart of the major steps of the algorithm for DNA resensing. Once the array is created and important parameters are computed, the algorithm checks to see if a new data point is outside a set threshold, in which case that threshold is stored in memory. Then, new points are compared to this threshold, and if one falls back inside of the threshold, it means the translocation is over and the next step of the protocol can be engaged, which is to wait and then flip the voltage.

FPGA Code

With all the LabVIEW basics covered, as well as a good understand of the logic of the code, we can now go over the specific implementation of that code onto the FPGA to achieve resensing. For the sake of clarity, we will cover the general structure of the code as a whole, with a visualisation of it being provided in figure 3.13, and then the detailed LabVIEW code will be dissected into its many components.

First, like mentioned before, since the code has to run indefinitely, the overarching structure used is a while loop, connected to a false boolean constant. Then, to be able to manually reset or turn off the code, for example in the event that it gets stuck searching for a molecule that has escaped the capture radius to translocate back in the pore (which will not happen), a boolean LED is used to control a case structure. One case contains the code, and the other is empty aside from having numerical constants set to 0 to reset all the important values (like the mean and variance). When the button is clicked, the code switches to the empty case and no data is read and no voltage is output at the pore. In the other case structure, an input channel, or connector (pink rectangle on the left of figure 3.17), that is physically connected to the nanopore setup, reads new data and transfers it to the rest of the code.

Resetting values in LabVIEW FPGA can be difficult, as every value is stored in memory until it is specifically assigned a new replacement value. In particular, feedback nodes and shift registers are only initialized the very first time they are called, and if they have a value stored in memory, they will not automatically erase it, even when calling them in a new iteration of a loop. For instance, if the code to compute the mean shown in figure 3.13 were to be placed in a while loop, one might expect the following to happen: 1) the input array is sent to the for loop, 2) the for loop indexes the array, 3) the first iteration of the for loop adds the first data point of the array with the initialized value of the feedback node (which is 0 as set by default), 4) the rest of the sum is computed as expected, 5) the mean is found, 6) a new iteration of the while loop begins, with the array being sent and indexed in the for loop, 7) the first data point is added to the initialized value of the feed-

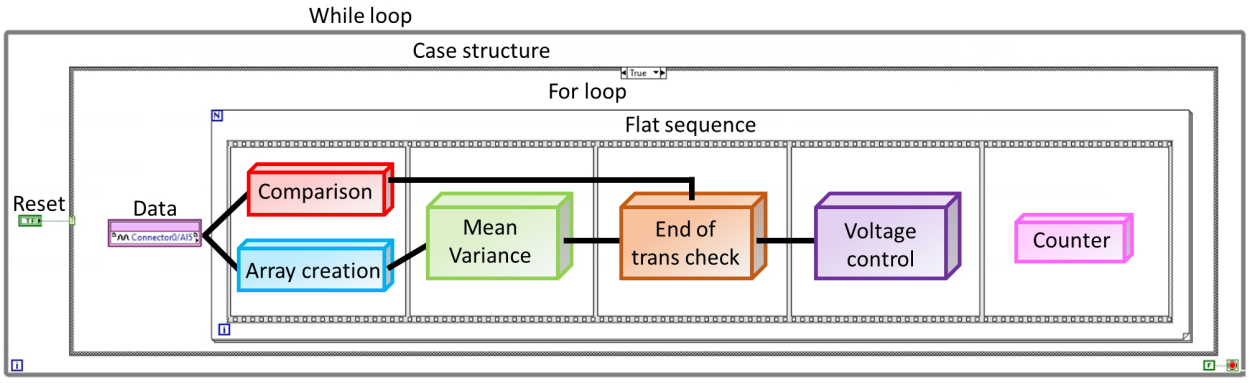

Figure 3.17: Resensing code structure. Schematics of the structure of the FPGA code in LabVIEW. Data is sent to the main components of the code, which sit inside of a for loop (to reset the mean and variance after every loop) and a flat sequence (to run the code in the desired order). This is surrounded by a case structure, with the condition not shown being empty, that serves to reset the code on command. A while loop encompasses everything such that the code runs indefinitely.

back node (which we would require to be 0, otherwise the sum will not be correct), and so on. However, despite being in a for loop, the feedback node and its initializer are not *explicitly* reset to their default value, and so what would actually happen is the sum from computed in the first while loop is stored in memory, then it gets added to the sum of the second while loop, and then this combined sum is added to the sum of the third while loop and it snowballs out of control. As such, in order to reset feedback nodes, the simplest and least expensive computation-wise method is to place the code from figure 3.13 inside of another for loop with a single iteration and perform an action called "Move initializer one loop out," which calls for the feedback node to reset its initializer every time this new for loop is entered.

Therefore, the bulk of the code is placed in a for loop. Then, it is split into five different cases of a flat sequence, in order to execute the code in the correct order. For the first case, the data is sent to both the array creation function as well as the comparison function, as they are both independent and can be executed in parallel without slowing down the FPGA. So, even if the data corresponds to a translocation and does not need to be added

to the array, it does not cost anything to do so regardless. The second case contains the loops to compute the mean and variance of the array. The third one contains the code to verify when a translocation ends. The fourth one is in charge of selecting which voltage to output to the pore. Finally, the fifth case is where a new increment of the counter, or a reset when necessary, is performed.

With the general structure out of the way, we can look at the particular code piece by piece. Note that since it is written in a horizontal, left-to-write fashion, and that it is quite long, it cannot be shown in a single figure and will instead be separated into different parts.

The first part is pictured in figure 3.18 and includes the comparison and array creation on the right and the starting part of all the shift registers on the left. Those include the counter, which increments by one every loop unless it is reset, which will be explained later, the mean and variance computed from the previous loop, and the voltage output loop counter, whose specific implementation and reset conditions are covered later as well. When a new data point is collected by the connector, it is sent to the blue "DATA" box (called subVI), where it gets added in the array and in lieu of the oldest data point following the algorithm from figure 3.18 c (described in more details in section 3.3.1). The "DATA" subVI outputs the new array, which is used in following parts of the code. Also, the new data point is sent at the top of the code, towards the "COMP" (short of comparison) subVI. However, before this occurs, a comparison function is used to compare the current counter value with the array size to see if the entire array has been filled. If it has not, the comparison (counter $\overset{?}{<}$ array size) returns a boolean true that sets the case structure to the true case, which only contains a boolean false statement to link to the local variable "Flip". In short, when this variable is set to true, it means a translocation event began and the voltage flipping sequence (check for the end of the translocation, wait a delay and flip the voltage) is engaged. When the array is full, the comparison returns a boolean false, which sets the case to the "COMP" subVI, whose algorithm is shown in figure 3.18 b. This serves to check if the point is an outlier by: 1) transforming the input

data into a single-precision float, 2) subtracting the mean from it, 3) squaring the result (to compare with variance instead of standard variation), 4) multiplying the variance to a threshold (set as threshold squared so that the outlier can be referred to as being *threshold* number of standard deviations off), and comparing the two results. The output is a boolean, which dictates the state of the "Flip" variable.

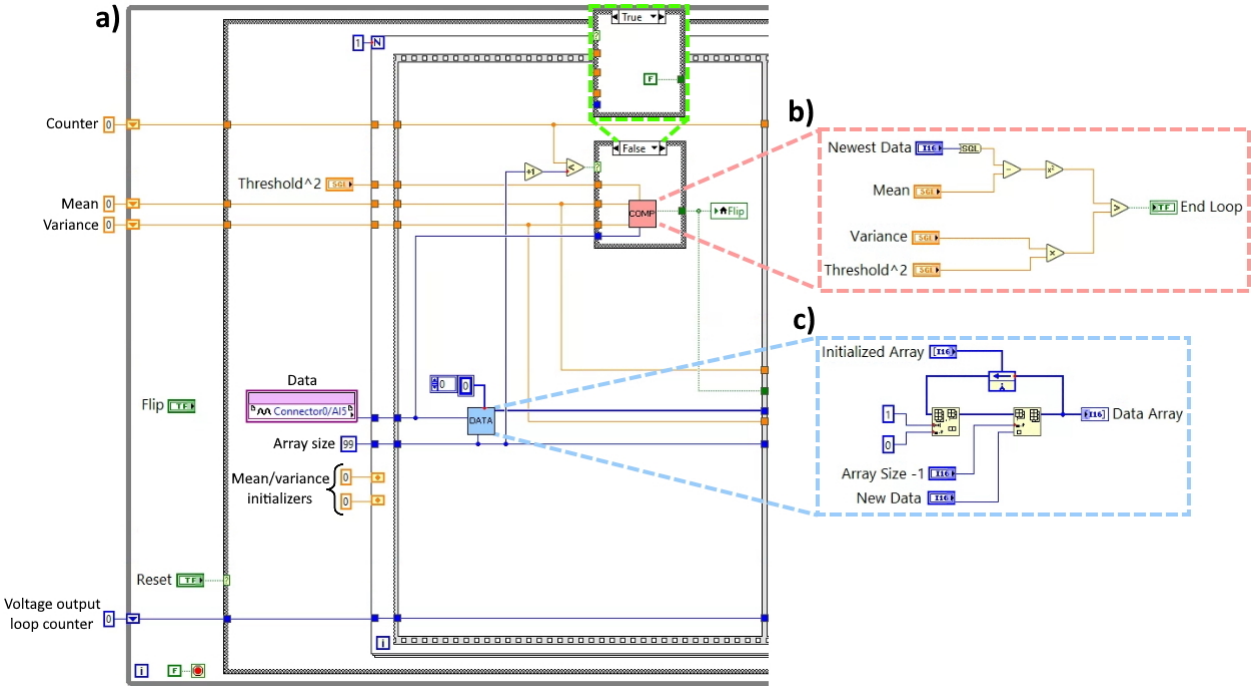

Figure 3.18: Array creation and comparison structures as implemented in the resensing code. (a) View of the left-most part of the code, which includes the shift registers' starting point and their initialized values, the definition of local variables "Flip" and "Reset", the data acquisition process and the array creation and comparison algorithms. The case structure in the code is set to the False case, and an image of the true case is provided above. **(b)** Comparison algorithm. Simple logic algorithm that compares the difference between the data and the mean (squared) with a fixed number of standard deviations (squared) and outputs a boolean constant. **(c)** Array creation algorithm. SubVI that contains the code creating the shifting array where the newest data is added and oldest data is removed during each loop.

The next segment of the flat structure (figure 3.19) is composed of the mean and variance computation codes. First, similarly to the previous part, a case structure encom-

passes everything, with a comparison function in front checking to see if the array is full or not. If it isn't full, the case is set to an empty case (with mean and variance set to 0), otherwise it is set to the case with the code inside. Inside of this, another case is placed this time controlled by the "Flip" variable. When "Flip" is set to false, the most recent event falls within the threshold, and so it is added to the array. The mean and variance of this new array must be computed, which is what happens in the false interior case structure. When "Flip" is true, this means a translocation event started, and thus the mean and the variance that were used to identify the event need to be stored. To do so, the true statement simply carries over the mean and variance from the previous segment and outputs them instead of newly computed values.

The third part of the flat sequence is where the code checks for the end of translocation events (figure 3.20). As usual, the code is inside of a case structure, which is controlled by the "Flip" variable. When there is no event, "Flip" is false, and so the output of the case structure is just a false boolean constant. When there is a translocation, the code enters into a while loop. This while loop includes another stacked sequence, where the first case recalls the threshold set via a local variable and acquires a new data point (which is converted to single-precision float). The second part does the same check as the comparison function, but in reverse. Each new data point is compared to the standard deviation and checked to see if they still fall outside the threshold (i.e. if the translocation is still happening). The output of the flat sequence is connected to the loop condition of the while loop, and as long as the translocation is not over and the comparison outputs a false constant, the loop keeps running. When a point lies within the threshold again, the loop ends and the true boolean constant is sent to the next part of the code. Note that a select function is used for the loop condition such that if the "Reset" LED is activated via the front panel, the output of the select function will be a true and the loop will end, which terminates the entire code.

Finally, the last part of the code is where the output voltage is set, as well as where the counter is computed (figure 3.21). It utilises a case structure that contains four cases.

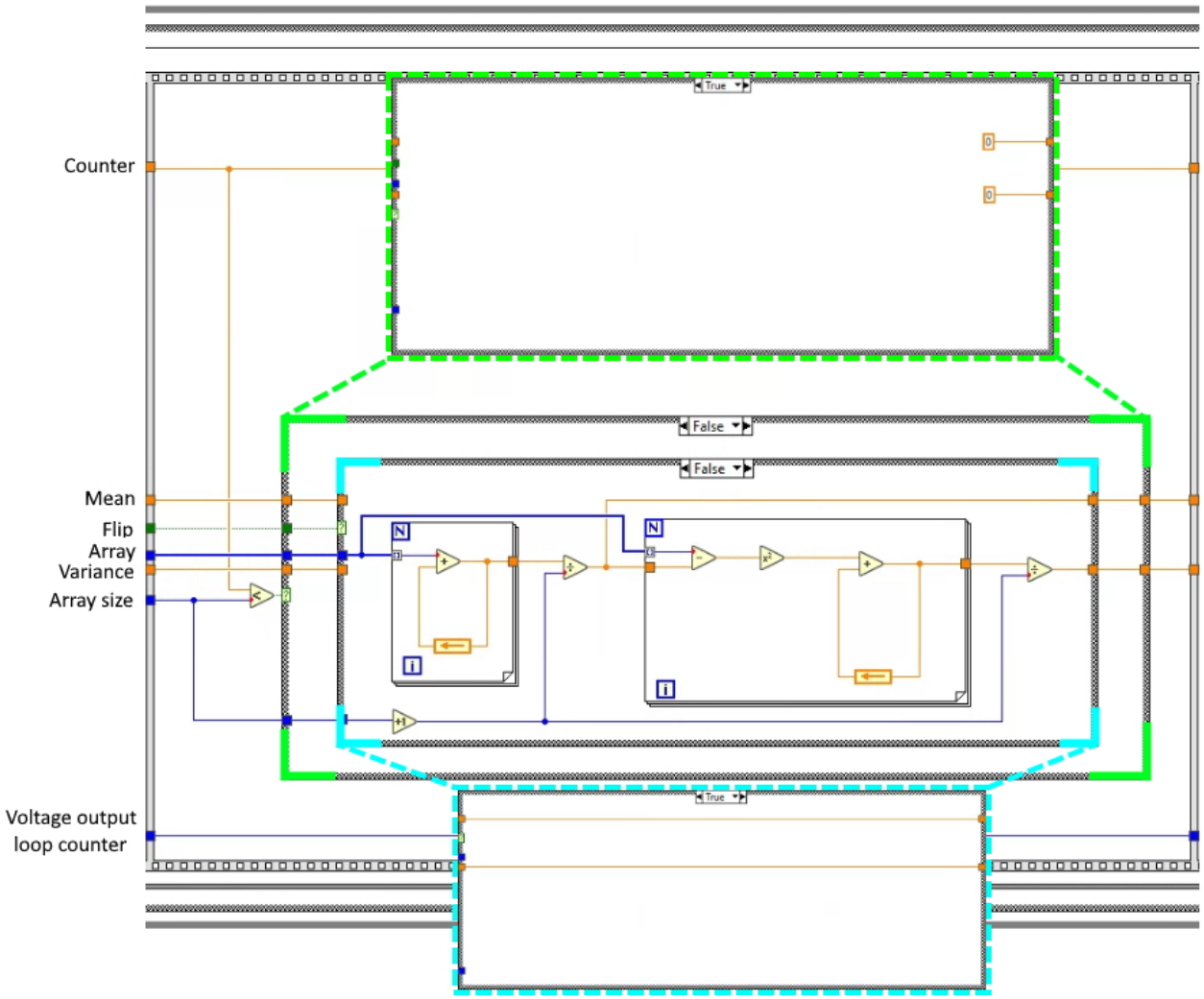

Figure 3.19: Mean and variance computation structures as implemented in the resensing code. Shown here are the two for loops required to compute the mean and the variance of the array as described in depths in section 3.3.1. A verification is first made to make sure the array is full before starting to compute the values. When "Flip" is set to false; that is, when there is no translocation occurring, the code runs as shown in the figure and a new mean and variance are computed. If "Flip" is true, the previously computed mean and variance are passed along to the next part of the code.

To decide which case to execute, the code relies on a mod counter set by a remainder function with a divisor of 4. The first case is simply outputting a fixed positive voltage at the pore, and a value of 0 is output from the case structure. This value is added to the voltage output loop counter, which starts at zero, so the case structure stays at the

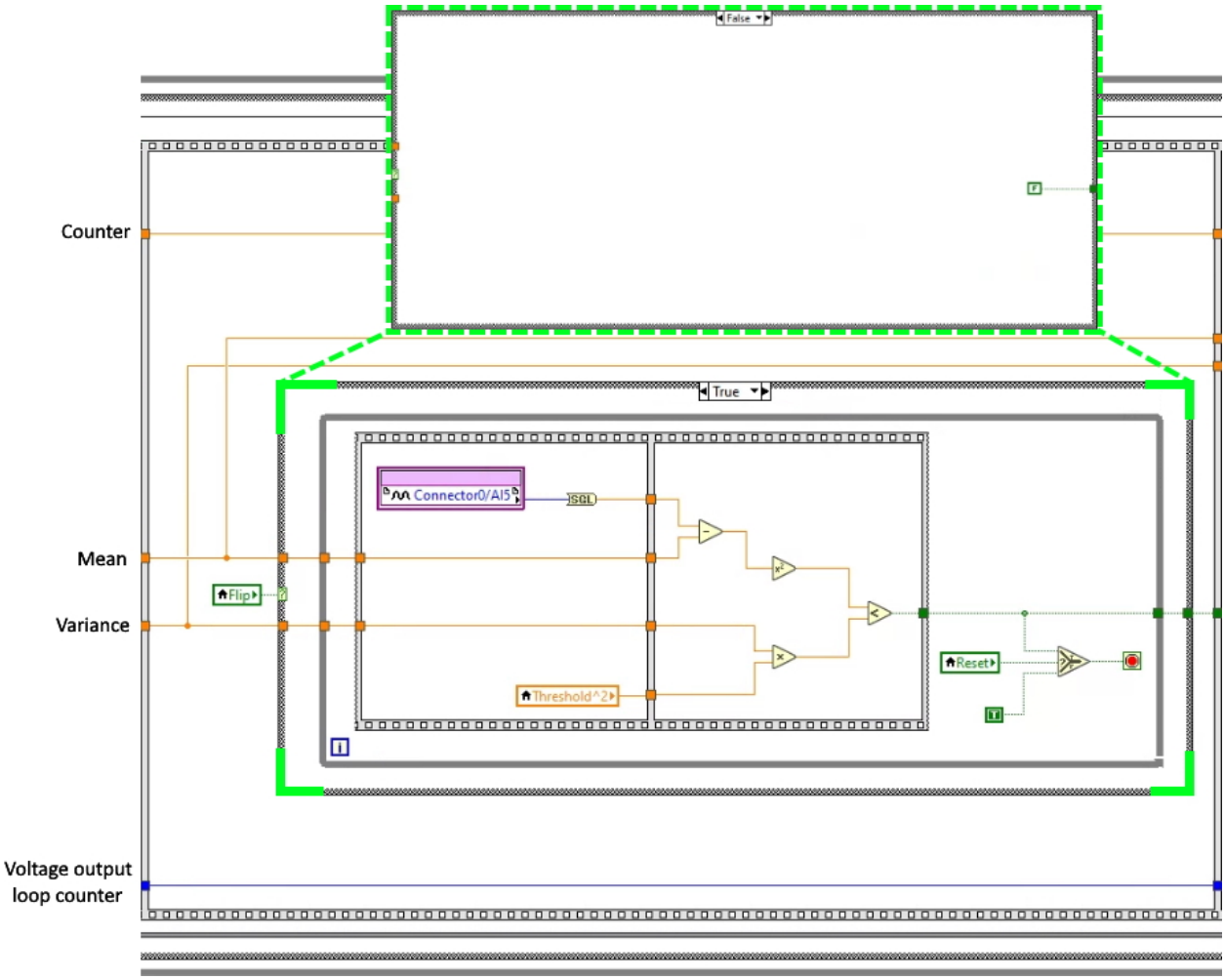

Figure 3.20: End of translocation check structure as implemented in the resensing code. This part of the code never runs unless a translocation is happening, as indicated by the fact that when "Flip" is false, the case structure is empty besides a false boolean constant. When a translocation happens, new data is acquired and compared with the threshold set previously. If the data is back within the threshold, this corresponds to the end of a translocation, and the loop ends, while transmitting a boolean true constant to the next part of the code. Also, a select function is used to reset the code if need be.

first case (case zero). However, when the output of the end of translocation check is set to true, this boolean value is converted to a binary 1 value, and is added to the loop counter. This changes the case to the second case, which has four steps: 1) wait a delay while outputting the same voltage to push the molecule away from the pore, 2) flip the voltage using a negate function, 3) wait another delay to avoid sampling the data during

the capacitance transient, and 4) output a 1 in the code such that for the very next loop, the case structure is changed to the third case. This case is the same as the first one, except that the voltage is negative. Once again, when a translocation ends, the case is swapped to case four, which is the same as the second one, except that all positive voltages are negative and vice-versa. After going through all four cases, the counter then reads 4, which is converted to a 0 using the remainder function, and the whole sequence begins again. The very last part of the flat structure includes only the counter, which starts at 0 and is incremented by 1 every loop, unless "Flip" is set to true, in which case the select function resets the value of the counter to 0. This is to make sure that once the voltage is flipped, the comparison code does not run until the array gets filled up again.

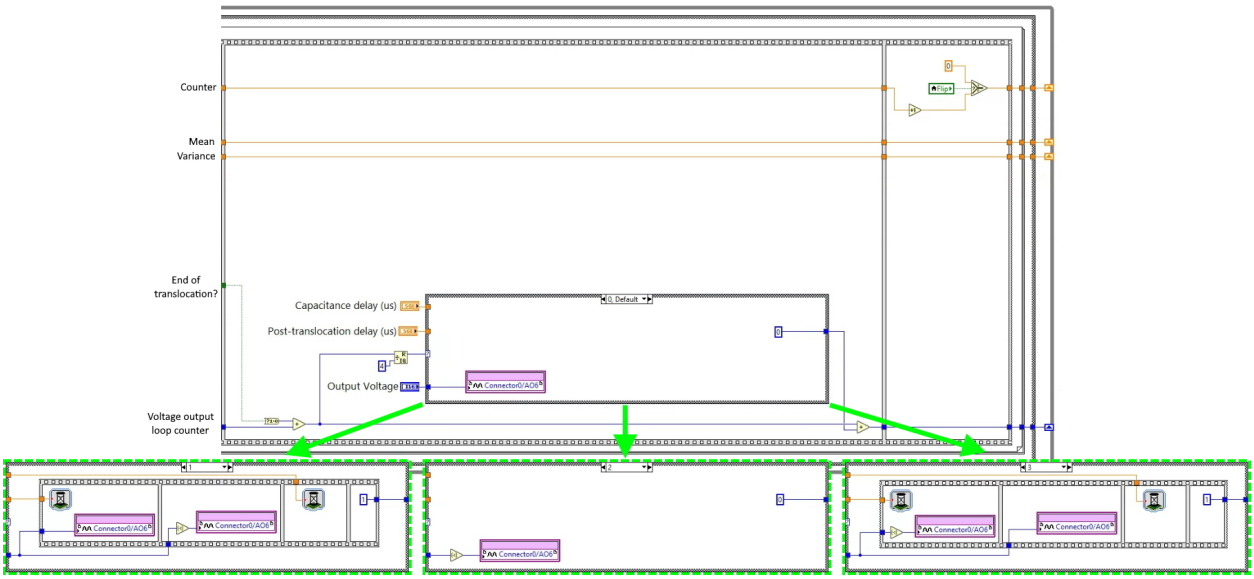

Figure 3.21: Voltage output and code counter as implemented in the resensing code. Shown in the middle and below is the code dictating the voltage output to the pore. When there is no translocation, either at the start or following a complete translocation, the code outputs a constant positive or negative voltage. Then, when the translocation is over, a delay is waited, then the voltage is flipped and then another small delay is waited. The counter at the top right increments by 1 every loop, unless a translocation has been identified, in which case the counter resets to 0.

Chapter 4

EXPERIMENTAL RESULTS

4.1 Conditioning

In this section, we will cover preliminary experiments aimed towards investigating the effects of conditioning on nanopore growth. As such, the most basic type of conditioning was used, which is a short voltage pulse of fixed voltage, repeated several times. Also, only the amplitude of the pulse was modified in order to keep the experiments as consistent as possible. From early experiments performed in our group and from the conditioning paper, it was determined that there exists a conditioning voltage barrier below which no amount of conditioning would make the pore grow. Additionally, from data presented in the paper, it is clear that with high enough voltages, the growth induced by conditioning will be exponential. As such, one could conceive the existence of two well-defined regimes (no effect and exponential growth). However, it also raises the questions as to how one regime transitions into the other, and what exactly the parameters that determine the voltage barrier observed are. We have therefore conducted a few experiments in hopes of gaining knowledge on the conditioning process, with the aim of later developing a robust understanding of this process using electrodynamics theory and computer simulations. Note that the following experimental results shown are based on a low sample size and could be the product of simple luck.

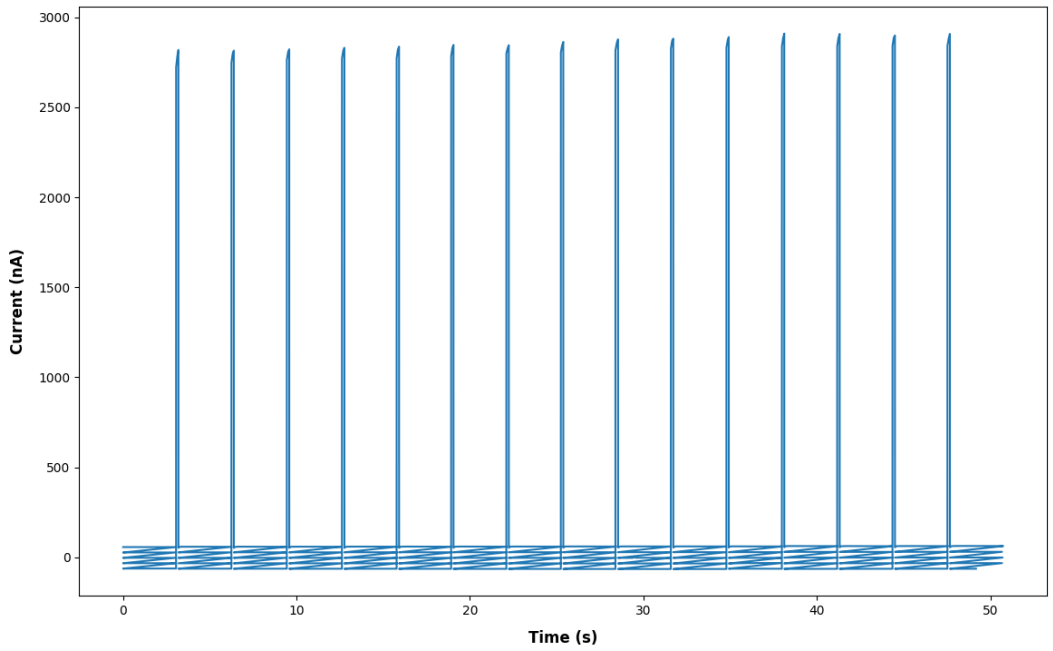

Figure 4.1: Current vs. time plot of low-voltage conditioning. Five points IV curves were performed between each pulse to carefully track the change in size between 4 V, 100 ms pulses, and no growth was observed.

First, we used a small range of voltages (usually between 2 and 4 V, and in the case of figure 4.1, 4 V was used) and a pulse time of 100 ms to confirm the lack of change in pore size over a long time. We have indeed found that regardless of the number of pulses applied, no growth was ever observed beyond the natural growth that all nanopores have in running buffer. We have also noticed that, in the case of certain pores, much high voltages (sometimes up to 8 V) could also cause no change in pore size, though we have yet to understand what might be the cause for that.

Another key feature that was noted in earlier experiments is that the pores seem to have a very rough starting geometry, which is indicated by a very unstable conductance curve, but this stability increases slowly over time. As such, we have experimented with a higher range of voltages (between 4 and 6 V) to see if stability could be affected, without modifying pore size by a large amount. We have used a nanopore that started with

a size of roughly 2.5 nm, and as shown in figure 4.2 a, its IV curve was fairly noisy and diverged from the linear fit expected in several locations. A stable IV is necessary to perform translocation experiments, as a noisy IV curve will correspond to a noisy pore, from which translocation events will be hard to detect. Assessing the stability of an IV curve is unfortunately not an exact process, as many factors (such as how well a buffer has been degased and the quality of DNA used) can affect noise during translocation experiments much more. Then, a fixed 6 V conditioning voltage was used for a few dozens of pulses of 100 ms in duration (figure 4.2 c), and the IV curve obtained afterwards (and plotted in figure 4.2 b) showed a huge improvement in terms of stability, while the pore still remained small (3 nm). Indeed, the correlation coefficient of the linear fit increased from 0.906 to 0.979 post conditioning, which indicates that the IV curves became much more linear. An important thing to note however is the lack of discernible pattern in the conditioning voltage, which was also observed in other similar experiments.

The next regime observed seemed to be very ambiguous at best, as the transition to and away from this regime was not very clear, and sometimes event non-existent. Data shown in figure 4.3 and in a couple more experiments points towards the potential existence of a conditioning regime where pore growth is linear. In this case, 8 V was used to grow the pore from 4.3 nm to 6.3 nm, which seemed to occur in a step-by-step fashion. However, since exponential curves are quite similar to linear curves at small values, it is possible that all growth is actually governed by exponential behaviors, and that the smallest possible voltage grows the pore in the (approximately) linear portion of the exponential curve.

Finally, like in the original conditioning paper, we have also observed a regime where growth was exponential, and seemed to begin at around 8 V. This regime seem to dominate over the linear one very early, as only 3% of pores grown using conditioning did not show an exponential growth. This would further support the belief that there could only be three different regimes instead of four. An example of a growth is shown in figure 4.4, where a 4.1 nm nanopore grew to 33.5 nm using 21 8 V pulses.

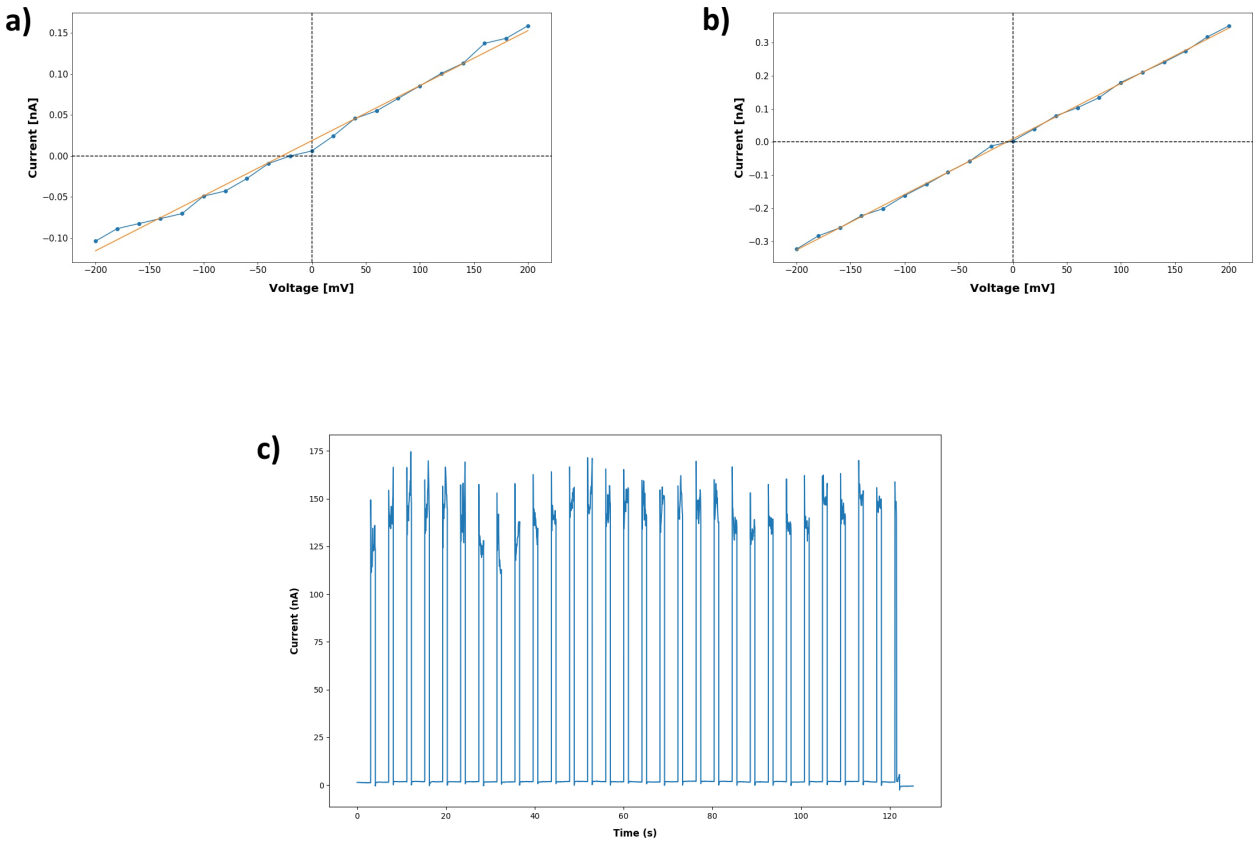

Figure 4.2: Increase in IV curve stability due to conditioning. **(a)** and **(b)** show an IV curve of the same pore before and after applying the conditioning voltage shown in **(c)**. The nanopore grew from 2.5 nm to 3 nm, which is in part due to natural growth, while its stability improved drastically post-conditioning (the correlation factor of the linear fit went from 0.906 for the first curve to 0.979 for the second). **(c)** Current vs. time of the conditioning voltage applied to the pore. There does not seem to be any discernible pattern in the current signal that could point towards an increase in stability.

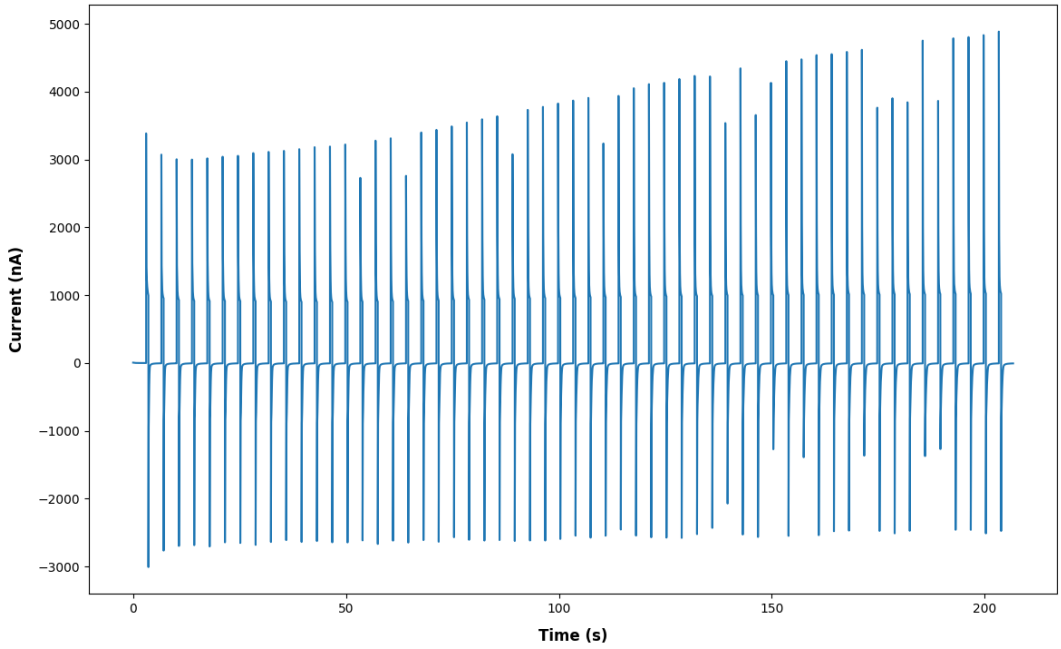

Figure 4.3: Current vs. time graph of conditioning showing linear growth. Conditioning pulses of 8 V for 100 ms were used to obtain this data. As the graph indicates, the increase in conductivity appears to be linear, which can be corroborated by the linear increase in size measured between the pulses. The pore was grown from 4.3 nm to 6.3 nm.

4.2 Resensing

In this section, we present the experimental data obtained from coupling the resensing system presented in section 3.3 to regular DNA translocation experiments in an attempt to observe molecules passing through the pore and back. Many parameters can be modified in the code, but most were kept constant to standardize the process. Among them, the array size was kept to 99, which was arbitrarily chosen but seemed to work well given the time a loop takes to execute and the typical translocation duration. Also, the voltage used to drive molecules through the pore was fixed at 120 mV. The threshold was often adjusted depending on the level of baseline noise that the pore had (increased for higher noise levels, decreased for lower noise) and was usually set at around 5-7 standard devi-

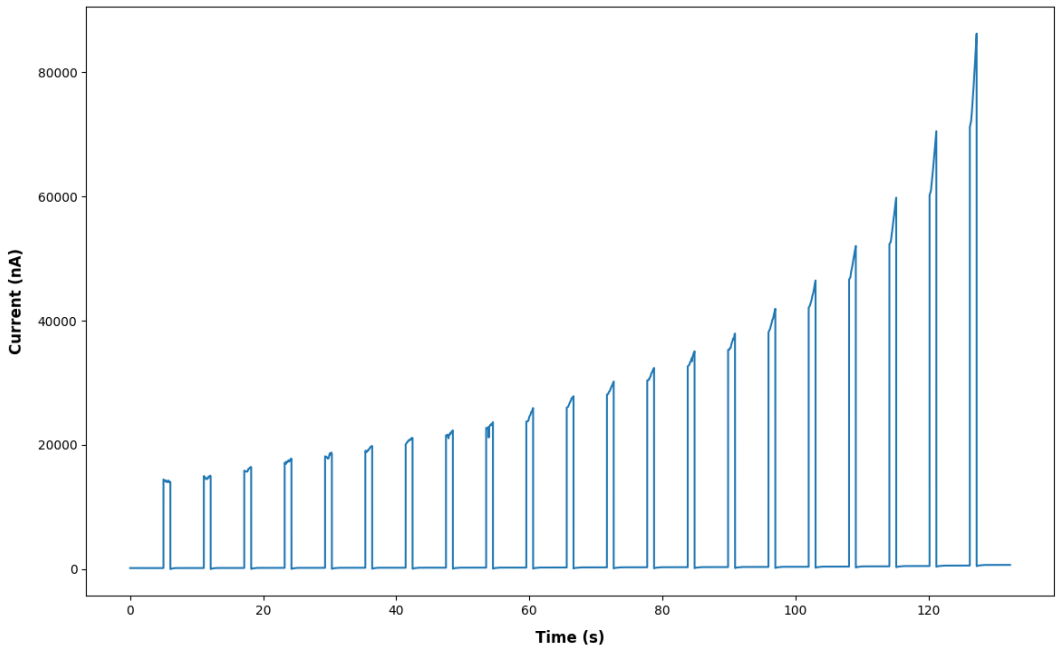

Figure 4.4: Current vs. time graph of a typical exponential growth. Pulses of 8 V were used to grow the pore from 4.1 nm to 33.5 nm in an exponential fashion. This behavior has been observed in the vast majority of pores grown using conditioning.

ations. Finally, the delay before the polarity of the driving voltage is reversed was also modified depending on the length of the molecule used. Longer molecules typically have a large capture radius, so higher times can be used to make sure the translocation back occurs far away from the capacitance transient.

Using λ-DNA (which contains 48 502 bases), the code was shown to successfully achieve resensing for several molecules, an example of which is shown in figure 4.5. In this case, the molecule was driven through a 13.5 nm nanopore in 2 M LiCl with 10 mM Tris and 1 mM EDTA buffer at 120 mV. A delay of 100 ms between detecting the end of a translocation event and flipping the voltage was used, which ensures that most of the molecules translocate back in the pore after the capacitance transient is done. As clearly indicated in the zoomed-in data of a translocation and a subsequent re-translocation (called post-translocation delay), both the duration and the amplitude of the current

blockades are virtually the same (with a time difference of 6 μs and an amplitude difference of 12 nA). Also, the time separating the two events is slightly higher than the post-translocation delay, which is expected, since once the voltage is flipped, the molecule still has some momentum in the opposite direction. With all these observations, we can safely conclude that the same molecule was indeed translocating through the pore both times. Note that the first event seems to have a high peak at the start which doesn't appear in the second event, but this could be due to a feature of the DNA either unfolding or getting ripped apart from the fast change of direction of the electric field.

Another interesting event that was observed is presented in figure 4.6. In this case, the same conditions as before were used, with the exception of the DNA studied (1 kilobase pairs DNA) and the post-translocation delay (50 ms). In this case, the current blockade has two distinct plateaus of different amplitudes with one being roughly half the size of the other. This corresponds to a DNA folded in half across most of its length, with a small unfolded tail at the end (figure 4.6 b). When flipping the voltage, if the molecule does not rotate or change geometry, one would expect the small unfolded end to translocate through the pore first, followed by the folded part, and so the smaller plateau would appear first. However, in this case, the large plateau happened first, meaning that the molecule either unfolded and refolded in the other direction (left path of figure 4.6 c), or it conserved its geometry but rotated by 180° (right path of figure 4.6 c). Despite not being able to conclude anything from a single event like this, this data demonstrate the viability for using the resensing system to investigate interesting phenomena relating to transport mechanisms. For instance, it would be interesting to use the system with non-uniformly charged proteins, to see if a rotation can be observed (such that the highest charged particle always translocates first).

An important thing to note is that resensing statistics of the experiments performed were quite poor. This is mostly due to issues regarding the quality of the DNA used, and the resulting issues when trying to set the code parameters correctly. Typical DNA solutions contain molecules that vary a lot in size (for example, a solution of 10 kilobase

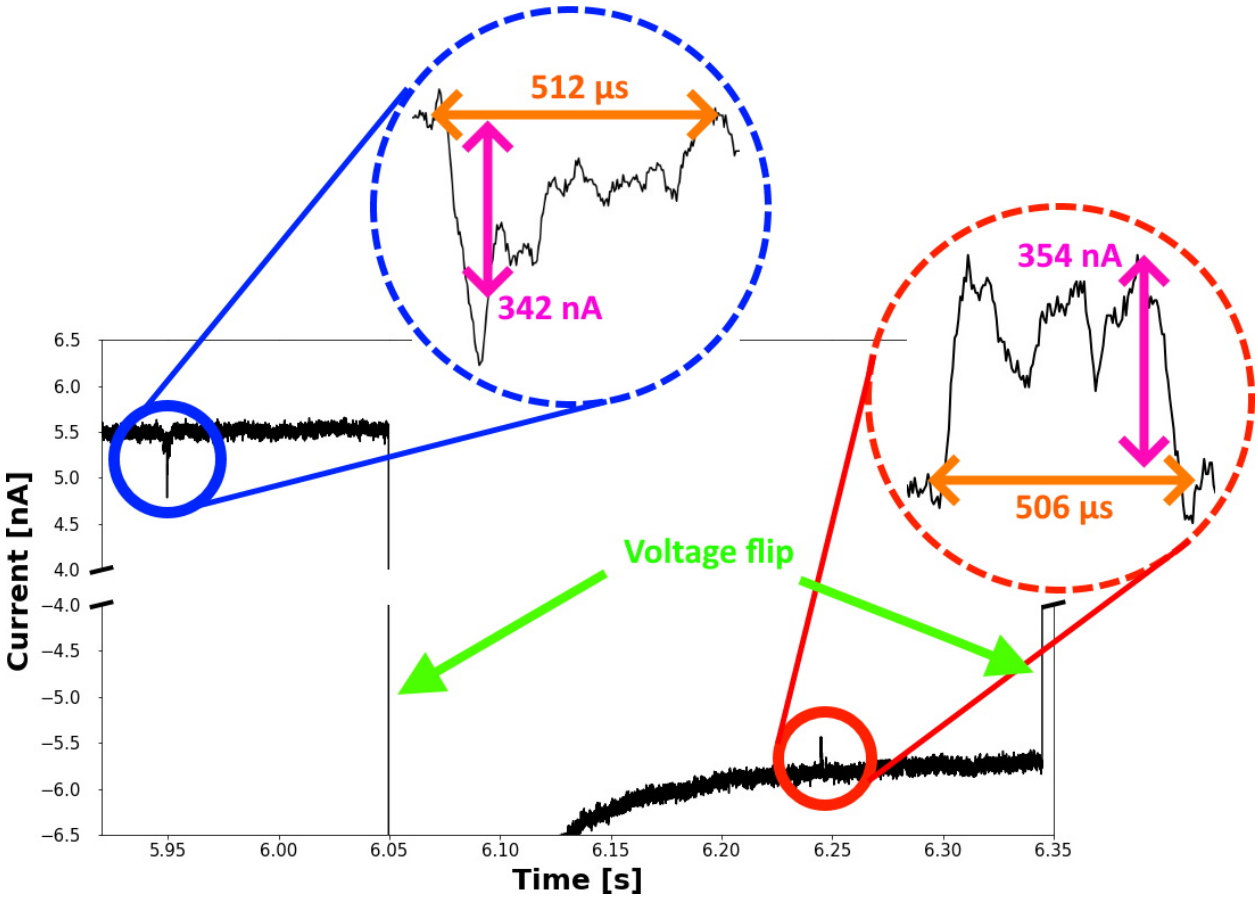

Figure 4.5: Successful resensing. Resensing data obtained in an experiment where λ-DNA was used in standard running buffer at 120 mV. As shown in the zoom-in data of the two blockade signals, both their duration and their amplitude are very similar, meaning that these are most likely signals for the same molecule. The concentration of DNA in solution used was set to a small amount (roughly 2 ng/µl) in order to avoid two molecules translocating at the same time. Note that the signal not shown in the negative currents is the capacitive response of the membrane, which could be compensated for, but if the translocation event were to occur during that time, its important features would be lost through the compensation. Across all initial experiments performed for this thesis (where no parameters were optimized), molecules were lost after 3 resensing at most. Using better parameters to detect translocation from noise and finding the optimal waiting time before flipping the voltage would certainly result in resensing the same molecule multiple times (though it is very time consuming).

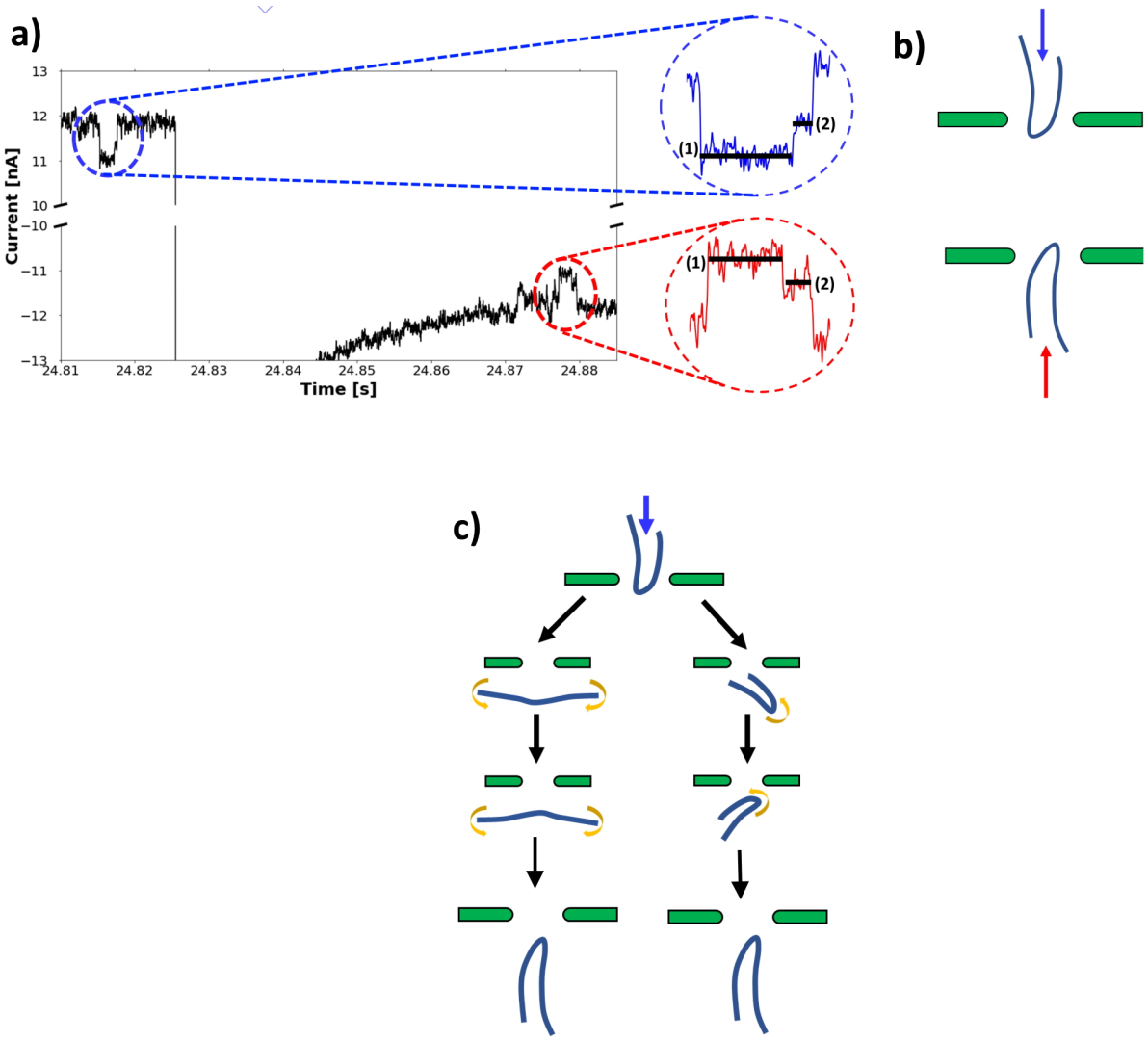

Figure 4.6: Resensing event highlighting interesting transport dynamics. **(a)** Resensing data of a 15 nm pore using 2 M LiCl, 10 mM Tris, 1 mM EDTA running buffer. A molecule folded in half with an unfolded tail shown in **(b)** translocates with the folded part first. This is illustrated by the zoomed-in data, where the translocation starts with a deeper, longer plateau, followed by a shorter and less deep one. Then, as the molecule translocates back into the pore, the folded part translocates first again. Without precisely knowing how this occurred, figure **(c)** shows the two possibilities that could have caused this, which is either the molecule unfolded, refolded again in the opposite direction and translocated back (left path) or it stayed folded but rotated by 180°.

pairs ladder DNA will have fragments ranging from 100 to 100,000 base pairs), which also affects the noise levels. Larger DNA tends to stick to the surface or walls of the pore and create significant fluctuations in the noise. As such, the threshold to differentiate noise and translocation events must be high enough, but also not too high otherwise the events will be missed. This often times results in only the bigger current blockade to be identified as translocation event, but those are mostly folded molecules, with a very fast blockade. This means that most resensing signals have a poor quality. To resolve this issue, the code will be modified to verify the length of translocation events and discard those deemed too short, as they do not provide any insightful data. This will be done by modifying the loop responsible for confirming the end of a translocation to add a condition where there must be a minimum amount of data that fall outside of the threshold (i.e. the translocation event is still happening).

Chapter 5

EXPERIMENTAL PROTOCOLS

5.1 Nanopore Fabrication

1. **Initial cleanup and preparation**

   (a) AFM chuck

      i. Scratch the bottom of the well and the biggest part of the side of the metal disk (removes potential crust of dried buffer and improves contact, only needs to be done for the first pore fabrication of the day)

      ii. Rinse the whole chuck with deionized (DI) water

      iii. Dry the top of chuck gently with a nitrogen gun (the bottom has sheets of mica that peel off easily with compressed air)

      iv. Dry the bottom of chuck with a wipe

      v. Fill the 2-20 μl pipette with 6 μl of either sodium perchrolate ($NaClO_4$) or degassed 2 molar lithium chloride (LiCl)

      vi. Place the pipette tip at the bottom of the well, overfill the chuck (while making sure no air is introduced) and pump out the buffer until it is leveled or slightly bulging up

   (b) Glass slide for plasma treatment

i. Clean a microscope glass slide with isopropanol (or IPA)

ii. Dry the IPA with a nitrogen gun or a wipe

2. **Chip preparation**

(a) Gently lift the corners of the chip to dislodge it from the sticky pad using plastic tweezers (avoids damaging the frame)

(b) Grab the chip from the sides with sharp tweezers positioned horizontally

(c) Bring the chip above the glass slide, then rotate the tweezers to flip the chip such that the membrane is facing down

(d) Gently drop the chip in the center of the glass slide

(e) Place the glass slide inside the chamber of the plasma tool such that the chip is centered and aligned with the glass markers

(f) Configure the plasma tool with the following settings: plasma time = 45, vacuum set point = 200.3, atmosphere vent = 00, gas stabilize = 25, vacuum alarm = 3:00, auto cycle-off = on

(g) Start the plasma treatment

(h) Once the chamber starts glowing purple, press the up arrow on the tool to see the chamber pressure and adjust the front valve (black knob) to keep the pressure between 200 and 300

(i) Once the treatment is over, take out the glass slide from the tool and flip it over on a wipe such that the membrane is facing up

(j) Pick up the chip and place it in the center of the well on the AFM chuck

3. **AFM setup**

(a) Software setup

i. Open the Nanoscope program (V613R1 icon on the top-left of the desktop)

ii. Open the "Nanopore Script" folder on the desktop

iii. Drag-and-drop "Backup.wks" in Nanoscope

iv. Select "Scan-Triple"

v. On the toolbar, click "Tools" then "Select scanner" and select "997SJ – 997J"

vi. Make sure the default scan settings are set properly (importantly, scan rate shouldn't exceed 1 Hz and force should be set to 0.150 V)

vii. Set the scan size to 1 μm and make sure the X and Y offsets are set to 0

viii. Begin the scan by pressing the green arrow ("Engage") in the toolbar

ix. Once the tip has made contact with the surface, make sure the scan looks fine and then increase the scan window to 50 μm

x. If the scan data has to be recorded, go to "File", the "Real time" and "Capture filename" to select where the capture should be recorded and the hit the "Ctrl+m" keys (simultaneously)

xi. Locate the membrane (make sure the center can be clearly seen through the scan; use the "Offset" button to displace the scan window if need be)

xii. Click the "Zoom" button, drag the zoom window at the center of the membrane, shrink the window size to <1 μm and click "Execute"

xiii. *Fabrication parameters (Microsoft Visual Studio program)*

   A. In the "Nanopore Script" folder, open the "single_nanopore_creation" folder and open the "single_nanopore_creation.vcproj" file

   B. Adjust the pulse voltage and time in the code as needed

   C. In the toolbar, select "Build" then click "Build solution" (at the bottom of the software in the code window, make sure there were no errors when building the code)

xiv. On the second computer on the left, open the "Picoscope" program

xv. Click the green "Play" button at the bottom left to start recording the breakdown data

xvi. Back on the main computer, in the Nanoscope program, click "Tools" on the toolbar, then "Run script" and select "single_nanopore_creation.dll" (do not double click the file unless the other computer is recording the data as it will begin the voltage pulse)

xvii. Press "Open"

xviii. On the computer on the left, verify that a breakdown occurred (the green signal will drop significantly if one occurs)

xix. If a breakdown didn't happen, increase the breakdown voltage or time and repeat steps xv-xviii

xx. If a breakdown happened, click the red arrow button ("Withdraw") on the toolbar at least twice to stop scanning and fully retract the tip

xxi. On the computer on the left, save the recorded data as ".psdata" and ".csv" files

(b) Pore fabrication

i. At the top of the setup, make sure the switch box has the "Mode" switch set to "FM/phase" and the "Tip or sample voltage" switch set to "Analog 2"

ii. On the control module below the switch box, make sure the switch labeled "Pulse" and "KPFM" is set to "Pulse" (the switch is located above the labels, not below)

iii. On the AFM, make sure the switch on the left is set to "AFM & LFM"

iv. Plug in the current probe in the connector at the bottom right

v. *Tip setup*

A. Unscrew the big screw at the back of the headstage and remove the tip holder

B. Flip the tip holder and lay it flat on a wipe

C. Press down on the whole holder to lift the gold clamp

D. Take the tip out of the box by holding it from the sides

E. Insert it under the gold clamp such that it is centered

F. Stop pressing down on the tip holder to secure the tip in place

G. Flip the tip holder over, place it back in the headstage and screw the big center screw

H. Place the headstage on the scanner

I. Using the two right-most screws at the top of the headstage, position the laser at the edge of the cantilever (when the laser is centered, the bar at the bottom of the second indicator on the AFM will be at around 7.5; play around with the tip position slightly to verify no other tip position increases the bar)

J. If the bar never becomes full, play with the lever at the back on the right to tilt the mirror until the laser is reflected in the center of the window

vi. Unscrew the two front screws below the scanner and pull the bottom-right switch (towards "Up") to lift the tip and avoid crashing it on the chip

vii. Take the headstage off the scanner

viii. Mount the chuck with the chip on the scanner and orient it such that the biggest part of the chuck metal disk is at the bottom left

ix. Place the headstage back on the scanner (make sure the tip doesn't touch the chip/chuck)

x. Secure the headstage by attaching the springs on the left and right on the small protruding metal bars

xi. Place the current probe on the metal disk

xii. Secure the headstage by attaching the springs on the left and right on the small protruding metal bars

xiii. Place the current probe on the metal disk

xiv. Lower the tip until its reflection can be seen on the chip through the front microscope and make sure the headstage is leveled in all directions

xv. Position the tip above the membrane using the screws at the bottom front and left of the headstage

xvi. Keep lowering the tip until it is just above the surface of the chip and adjust its position if needed

xvii. Adjust the left screw at the back of the headstage first then the right screw on the top until both numbers at the bottom of the AFM read between -0.05 and 0.05

xviii. Fabricate the pore using the procedure listed above

xix. Lift the tip using the screws, remove the current probe, unhook the springs and take off the headstage from the scanner

xx. Take the chuck off the scanner

xxi. Fill a dish or tube with IPA and place the chip inside

xxii. Cover with Parafilm if needed and label the sample

5.2 Conductance and Size Measurements

1. **Initial cleanup and preparation**

   (a) Electrodes

      i. Sand the wire electrodes until the silver is completely exposed using fine sand paper

      ii. Clean the electrodes with DI

      iii. Place the electrodes in a dish filled with bleach for at least 30 minutes and ensure that they are completely covered (by aluminium foil or in a box) so as to avoid light-induced corrosion

      iv. Clean the electrodes with DI and cover them

   (b) Buffer preparation

i. Remove the cap of the storage tube containing the buffer, cover the tube with Parafilm and punch holes with tweezers

ii. Place the tube in the vacuum chamber and place wipes around it

iii. Plug the pump to start vacuuming

iv. Wait until bubbles escape the buffer only once every few seconds, then unplug the pump and turn the valve on the chamber to depressurize it

(c) Translocation chuck

i. Rinse the chuck with IPA and DI

ii. Place the chuck in a dish filled with DI

iii. Place the dish in the sonicator bath and sonicate for 180 seconds

iv. Dry the chuck using a nitrogen gun

2. **Physical setup**

(a) Dip the chip in IPA

(b) Dry the chip on a wipe, making sure both sides are completely dry

(c) Clean an area of a chip-storing plastic box using tape (to avoid contamination)

(d) Place the chip membrane side facing up on the cleaned area

(e) Clean both sides of an o-ring using tape by sticking it to the o-ring and peeling it a few times

(f) Carefully place the o-ring on the chip, making sure the membrane is aligned with the center of the o-ring and the o-ring makes contact with the chip all around (to seal everywhere)

(g) Gently press down on the o-ring using tweezers to lightly bond the o-ring

(h) Add 2 μl of buffer in the center hole of the o-ring (to wet the membrane) and make sure no liquid is on the o-ring itself

(i) Flip the chip over and place it in the center of one half of the translocation chuck

(j) Clean another o-ring using tape

(k) Place the o-ring on the chip, making sure the two o-rings are aligned together

(l) Add 2 µl of buffer in the center hole of the o-ring

(m) Screw the both parts of the chuck together

(n) Add 75 µl of buffer in both reservoirs of the chuck

(o) Degas the chuck in a weak vacuum chamber for 30 minutes

(p) Place the chuck in the metal box of the setup used to run experiments

(q) Connect an electrode to the headstage component and another to the ground wire

(r) Insert the electrodes in both reservoirs from the top

(s) Close the metal box

(t) Turn on the Axopatch patch-clamp system and the digitizer

(u) Adjust the pipette offset of the patch-clamp using the left-most knob until it reads 0.000 nA

3. **Software steps**

(a) <u>IV curve measurement</u>

i. Open the "Clampex" program

ii. In the toolbar, click on "File" then "Set data file name..." and select where the recorded files should be stored

iii. In the toolbar, click on "Acquire" then "Open protocol" and select the "IV CURVE.pro" file

iv. Below the toolbar, click on the red circle icon ("Record") to run and record an IV curve

(b) Conductance calculation

i. Open the "Clampfit" program

ii. Click the "Open file" icon below the toolbar on the left and select the IV curve file

iii. Place the black line ("Cursor") numbered 1 at 1.5 s on the x-axis and the one numbered 2 at roughly 2.9 s

iv. Click on the IV icon ("IV plot") below the toolbar and press "Ok"

v. Click on the F(x) icon ("Fit") button and press "Ok" to perform a linear fit of the IV curve

vi. Click on the "Fitting results" button to obtain the slope of the IV (which corresponds to the conductance)

vii. Go to `www.xcapaldi.com/pore-calculator.html`, enter the buffer conductivity, membrane thickness and conductance to obtain the pore size

5.3 Conditioning

1. **Physical setup**

(a) Connect the electrodes to the two alligator clips in the metal box, making sure the clips do not touch

(b) Place the chuck in the box and insert the electrodes in both reservoirs of the chuck from the top

(c) Close the box

(d) Turn on the "Keithley" device

2. **Software setup**

(a) Open the "conditioning" folder on the desktop

(b) Open the "Grow to dimension with IV.py" file

(c) Enter the buffer conductivity, membrane thickness and directory folder

(d) Set the amplitude and duration of the conditioning voltage pulse under "Conditioning/growth voltage (V)" and "Pulse time (ms)" respectively

(e) Set the amount of pulses between each IV curve and the waiting time between each pulse under "Pulse number" and "Down time (ms)" respectively

(f) Set the desired finale pore diameter under "Target pore diameter (nm)"

(g) Click "Queue" to start the conditioning process

(h) If needed, press "Abort" to abort the process

(i) *To start a new process following an aborted process, click the "Queue" button again, which will become "Resume" after the first click*

5.4 Translocation Experiments

1. **Physical setup**

   (a) Take the DNA tube out of the freezer and put it in the tube heater

   (b) Thaw the DNA at 37 °C for 3 minutes

   (c) Fill a pipette with 75 µl of DNA solution

   (d) Fill (or flush out the existing buffer) one reservoir of the chuck with the DNA solution

   (e) Place the chuck in the box such that the reservoir with the DNA is opposite of the headstage

   (f) Insert the electrodes in both reservoirs of the chuck from the top

   (g) Close the box

2. **Software setup**

(a) Data recording

i. Open the "Clampex" program

ii. In the toolbar, click on "File" then "Set data file name..." and select where the recorded files should be stored

iii. In the toolbar, click on "Acquire" then "Open protocol" and select "Translocation test protocol.pro"

iv. Set the translocation voltage (usually 120 mV to 200 mV) below CMD0 on the left

v. Below the toolbar, click on the red circle icon ("Record" to run and record the translocation data

(b) Data analysis

i. Open the "Clamfit"

ii. Click the "Open file" icon below the toolbar and select the translocation data file

iii. Set a filter to the data if necessary

5.5 Resensing/FPGA Setup

1. **Physical setup**

(a) Connect the FPGA Ai5 (connectors 60/26) cable to Channel 1 Primary of the patch-clamp and Analog input 1 of the digitizer

(b) Connect the FPGA AO6 (connectors 49/15) cable to Channel 1 Command of the patch-clamp and Analog input 0 of the digitizer

(c) Place the chuck in the metal box

(d) Plug the electrodes inside the box and insert them in both reservoirs of the chuck from the top, making sure the electrode connected to the headstage is not in the reservoir with DNA

(e) Close the box

2. **Software setup**

(a) FPGA code

i. Open "NI LabVIEW 2020"

ii. Select "Pore Recapture Project.lvproj" then click "SP Resensing"

iii. Make sure the FPGA is set to "FPGA target" by right-clicking "FPGA target (RIO0, PCIe-7951R)" then "Select execution mode" and "FPGA target"

iv. Enter the desired translocation voltage under "Output voltage"

v. Enter the desired delay before switching the direction of the voltage under "Post translocation delay (microseconds)"

vi. Enter the desired trigger threshold (standard deviation squared) under "Threshold" and delay before acquiring data after changing the voltage polarity under "Capacitance delay"

vii. Click the white arrow icon ("Run") to run the code

viii. Press the "Reset" green button to initiate the code, and press it anytime the code needs to be reset to its original state

(b) Data recording

i. Open the "Clampex" program

ii. In the toolbar, click on "File" then "Set data file name..." and select where the recorded files should be stored

iii. In the toolbar, click on "Acquire" then "Open protocol" and select "FPGA translocation.pro"

iv. Below the toolbar, click on the red circle icon ("Record") to run and record the translocation data

5.6 Miscellaneous

- Sterilizing pipette tips

  1. Fill the reservoir of the autoclave with DI until it reaches the safety valve
  2. Flip the power switch of the autoclave to turn it on
  3. Turn the bottom knob at the front to "Fill water" to fill the chamber with water
  4. Turn the knob again to "Sterilize" when water reaches the line at the front of the chamber
  5. Place a piece of sterilization tape on the box of tips
  6. Load the box in the chamber and close the door of the autoclave
  7. Set the temperature of the autoclave to 250 °F/121 °C using the knob in the middle
  8. Set the timer to 60 minutes using the knob at the top
  9. Once the timer is over, turn the bottom knob to "Exh + Dry"
  10. Wait several minutes for the chamber to completely vent, then open the door and take out the box of tips

- DNA solution

  1. Take the tube of concentrated DNA out of the freezer
  2. Thaw the DNA in the tube heater at 37 °C for 3 minutes
  3. Compute the dilution factor to obtain the desired concentration for a volume of roughly 1-2 ml
  4. Mix a larger volume of the required quantities of DNA and buffer in a 2 ml tube
  5. Aliquot 80 ml of solution in several smaller-sized tubes

6. Store the tubes that will be used within the next few days in the refrigerator and the rest in the freezer
7. Store the concentrated DNA in the freezer

- AFM chuck

1. Cut a square of thick PDMS material roughly 2 by 2 cm
2. Punch a hole in the center using the hole puncher
3. Glue the PDMS square onto a circular metal disk using UV glue
4. Cure with a UV light for 15 s
5. Glue a sheet of mica at the bottom of the chuck using UV glue
6. Cure with a UV light for 15 s

- AFM current probe

1. Cut a small rectangle from a sheet of copper
2. Strip both ends of a shielded wire
3. Solder the wire to one end of the rectangle
4. Solder the metal pin connectors to the other end of the wire, making sure there are no shorts
5. Glue a small, circular magnet on the other end of the copper rectangle using UV glue
6. Cure with a UV light for 15 s

- O-rings

1. Cut a piece of PDMS
2. Peel the plastic cover on one side (as to expose the PDMS)
3. Punch circles in the PDMS sheet (with the PDMS side up) using the hole-puncher (for the outer ring)

4. Place the PDMS sheet on a cutting board, place the 2 mm sized hole puncher in the center of the circles and hit it with the hammer (for the inner ring)
5. Peel the remaining plastic cover and separate the o-rings from the PDMS sheet

- Electrodes

1. Strip both ends of a regular insulated copper wire around 10 cm long
2. Cut a 3 cm long piece of silver wire
3. Solder the silver wire to one end of the copper wire
4. Cover the solder with a heat shrink tube
5. Use the heat gun to shrink the tube
6. Solder the other end of the copper wire with a metal pin
7. Cover the solder with a heat shrink tube and use the heat gun to shrink it

Chapter 6

CONCLUSION

In this study, we have developed a good understanding of the fundamentals of nanopore-based research. When generating an electric field across a pore, charged ions and macromolecules translocate through, in turn generating a specific signal based on the experimental parameters used, as well as the molecules themselves. We have also seen that biological nanopores are a staple in genetics applications through their ability to perform sequencing, but we have also laid the case for the use of solid-state nanopores. Their practicality and ease of access were the main factors allowing us to perform most of the work presented here, in part due to the development of a pore fabrication technique combining the precision of an atomic force microscope with the high-throughput capabilities of the dielectric breakdown method.

We have also shown how pores can be modified through the action of a high electric field. An overview of the setup used for conditioning was provided, as well as a few important codes capable of altering the pores in different ways. With this, we have shown that it is possible to carefully track growth in nanopores using growth curves, where the size of pores is tracked following the application of a voltage pulse several times. Furthermore, by obtaining growth (or lack thereof) curves at several voltages, we have theorized the possibility for different growth regimes to exist, which might vary based on the type of pore used or its initial conditions like its geometry or its size. Further work

of subsequent analysis and modelling might potentially provide insightful knowledge on how nanopores get modified in the presence of an electric field, and how different fabrication techniques influence specific pore characteristics.

Finally, we have covered the basics required to create codes using LabVIEW, and more specifically, how to execute them on an FPGA device. With key implementations reviewed, we covered the specific requirements that a resensing code must have, which is based on a local peak detection algorithm. The code was used in real experiments to successfully bounce macromolecules back-and-forth across the pore, with the parameters controlling the flow of molecules being dynamically modified during the experiment to improve the rate of recaptures. With this, some key events were highlighted where molecules seemed to undergo either folding/unfolding or rotation, and we conjectured the possibility to use such a resensing setup to probe the effect of an electric field on the translocation of non-uniformly charged molecules like certain proteins.

Bibliography

[1] M. Wanunu, "Nanopores: A journey towards dna sequencing," *Physics of Life Reviews*, vol. 9, p. 125–158, Jun 2012.

[2] J. J. Kasianowicz, J. W. Robertson, E. R. Chan, J. E. Reiner, and V. M. Stanford, "Nanoscopic porous sensors," *Annual Review of Analytical Chemistry*, vol. 1, p. 737–766, Jul 2008.

[3] C. Dekker, "Solid-state nanopores," *Nature Nanotechnology*, vol. 2, p. 209–215, Mar 2007.

[4] B. M. Venkatesan and R. Bashir, "Nanopore sensors for nucleic acid analysis," *Nature Nanotechnology*, vol. 6, p. 615–624, Oct 2011.

[5] D. Branton, D. W. Deamer, A. Marziali, H. Bayley, S. A. Benner, T. Butler, M. Di Ventra, S. Garaj, A. Hibbs, X. Huang, S. B. Jovanovich, P. S. Krstic, S. Lindsay, X. S. Ling, C. H. Mastrangelo, A. Meller, J. S. Oliver, Y. V. Pershin, J. M. Ramsey, R. Riehn, G. V. Soni, V. Tabard-Cossa, M. Wanunu, M. Wiggin, and J. A. Schloss, "The potential and challenges of nanopore sequencing," *Nature Biotechnology*, vol. 26, p. 1146–1153, Oct 2008.

[6] B. Sakmann and E. Neher, *Single-Channel Recording*. Boston, MA: Springer US, 1995.

[7] O. Otto and U. F. Keyser, *DNA Translocation*, p. 31–58. Elsevier, 2013.

[8] S. Howorka and Z. Siwy, "Nanopore analytics: sensing of single molecules," *Chemical Society Reviews*, vol. 38, no. 8, p. 2360, 2009.

[9] B. Katz and R. Miledi, "The statistical nature of the acetylcholine potential and its molecular components," *The Journal of Physiology*, vol. 224, p. 665–699, Aug 1972.

[10] D. W. Deamer and M. Akeson, "Nanopores and nucleic acids: prospects for ultrarapid sequencing," *Trends in Biotechnology*, vol. 18, p. 147–151, Apr 2000.

[11] J. Kasianowicz, E. Brandin, D. Branton, and D. Deamer, "Characterization of individual polynucleotide molecules using a membrane channel," *Proceedings of the National Academy of Sciences*, vol. 93, p. 13770–13773, Nov 1996.

[12] I. M. Derrington, T. Z. Butler, M. D. Collins, E. Manrao, M. Pavlenok, M. Niederweis, and J. H. Gundlach, "Nanopore dna sequencing with mspa," *Proceedings of the National Academy of Sciences*, vol. 107, p. 16060–16065, Sep 2010.

[13] E. A. Manrao, I. M. Derrington, A. H. Laszlo, K. W. Langford, M. K. Hopper, N. Gillgren, M. Pavlenok, M. Niederweis, and J. H. Gundlach, "Reading dna at single-nucleotide resolution with a mutant mspa nanopore and phi29 dna polymerase," *Nature Biotechnology*, vol. 30, p. 349–353, Apr 2012.

[14] J. Li, D. Stein, C. McMullan, D. Branton, M. J. Aziz, and J. A. Golovchenko, "Ion-beam sculpting at nanometre length scales," *Nature*, vol. 412, p. 166–169, Jul 2001.

[15] A. J. Storm, J. H. Chen, X. S. Ling, H. W. Zandbergen, and C. Dekker, "Fabrication of solid-state nanopores with single-nanometre precision," *Nature Materials*, vol. 2, p. 537–540, Aug 2003.

[16] H. Kwok, K. Briggs, and V. Tabard-Cossa, "Nanopore fabrication by controlled dielectric breakdown," *PLoS ONE*, vol. 9, p. e92880, Mar 2014.

[17] G. F. Schneider, S. W. Kowalczyk, V. E. Calado, G. Pandraud, H. W. Zandbergen, L. M. K. Vandersypen, and C. Dekker, "Dna translocation through graphene nanopores," *Nano Letters*, vol. 10, p. 3163–3167, Aug 2010.

[18] C. A. Merchant, K. Healy, M. Wanunu, V. Ray, N. Peterman, J. Bartel, M. D. Fischbein, K. Venta, Z. Luo, A. T. C. Johnson, and M. Drndić, "Dna translocation through graphene nanopores," *Nano Letters*, vol. 10, p. 2915–2921, Aug 2010.

[19] B. N. Miles, A. P. Ivanov, K. A. Wilson, F. Doğan, D. Japrung, and J. B. Edel, "Single molecule sensing with solid-state nanopores: novel materials, methods, and applications," *Chem. Soc. Rev.*, vol. 42, no. 1, p. 15–28, 2013.

[20] E. Beamish, H. Kwok, V. Tabard-Cossa, and M. Godin, "Fine-tuning the size and minimizing the noise of solid-state nanopores," *Journal of Visualized Experiments*, p. 51081, Oct 2013.

[21] M. Gershow and J. A. Golovchenko, "Recapturing and trapping single molecules with a solid-state nanopore," *Nature Nanotechnology*, vol. 2, p. 775–779, Dec 2007.

[22] D. Stein, "Molecular ping-pong," *Nature Nanotechnology*, vol. 2, p. 741–742, Dec 2007.

[23] J. K. Rosenstein, M. Wanunu, C. A. Merchant, M. Drndic, and K. L. Shepard, "Integrated nanopore sensing platform with sub-microsecond temporal resolution," *Nature Methods*, vol. 9, p. 487–492, May 2012.

[24] X. Liu, Y. Zhang, R. Nagel, W. Reisner, and W. B. Dunbar, "Controlling dna tug-of-war in a dual nanopore device," *Small*, vol. 15, p. 1901704, Jul 2019.

[25] X. Liu, P. Zimny, Y. Zhang, A. Rana, R. Nagel, W. Reisner, and W. B. Dunbar, "Flossing dna in a dual nanopore device," *Small*, vol. 16, p. 1905379, Jan 2020.

[26] R. Brown, "Xxvii. a brief account of microscopical observations made in the months of june, july and august 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies," *The Philosophical Magazine*, vol. 4, p. 161–173, Sep 1828.

[27] A. Einstein, "Über die von der molekularkinetischen theorie der wärme geforderte bewegung von in ruhenden flüssigkeiten suspendierten teilchen," *Annalen der Physik*, vol. 322, no. 8, p. 549–560, 1905.

[28] R. P. Buck, "Kinetics of bulk and interfacial ionic motion: microscopic bases and limits for the nernst—planck equation applied to membrane systems," *Journal of Membrane Science*, vol. 17, p. 1–62, Jan 1984.

[29] T. Albrecht, T. Gibb, and P. Nuttall, *Ion Transport in Nanopores*, p. 1–30. Elsevier, 2013.

[30] S. W. Kowalczyk, A. Y. Grosberg, Y. Rabin, and C. Dekker, "Modeling the conductance and dna blockade of solid-state nanopores," *Nanotechnology*, vol. 22, p. 315101, Aug 2011.

[31] P. Horowitz and W. Hill, *The Art of Electronics*. USA: Cambridge University Press, 3rd ed., 2015.

[32] J. E. Hall, "Access resistance of a small circular pore.," *Journal of General Physiology*, vol. 66, p. 531–532, Oct 1975.

[33] W. R. Smythe, *Static and dynamic electricity*. International series in pure and applied physics, New York: McGraw-Hill, 3. ed ed., 1968.

[34] J. D. Jackson, *Classical electrodynamics*. New York: Wiley, 3rd ed ed., 1999.

[35] V. Tabard-Cossa, D. Trivedi, M. Wiggin, N. N. Jetha, and A. Marziali, "Noise analysis and reduction in solid-state nanopores," *Nanotechnology*, vol. 18, p. 305505, Aug 2007.

[36] R. M. M. Smeets, U. F. Keyser, N. H. Dekker, and C. Dekker, "Noise in solid-state nanopores," *Proceedings of the National Academy of Sciences*, vol. 105, p. 417–421, Jan 2008.

[37] R. M. M. Smeets, N. H. Dekker, and C. Dekker, "Low-frequency noise in solid-state nanopores," *Nanotechnology*, vol. 20, p. 095501, Mar 2009.

[38] E. Beamish, H. Kwok, V. Tabard-Cossa, and M. Godin, "Precise control of the size and noise of solid-state nanopores using high electric fields," *Nanotechnology*, vol. 23, p. 405301, Oct 2012.

[39] A. Rhoads and K. F. Au, "Pacbio sequencing and its applications," *Genomics, Proteomics & Bioinformatics*, vol. 13, p. 278–289, Oct 2015.

[40] M. Jain, H. E. Olsen, B. Paten, and M. Akeson, "The oxford nanopore minion: delivery of nanopore sequencing to the genomics community," *Genome Biology*, vol. 17, p. 239, Dec 2016.

[41] J. Clarke, H.-C. Wu, L. Jayasinghe, A. Patel, S. Reid, and H. Bayley, "Continuous base identification for single-molecule nanopore dna sequencing," *Nature Nanotechnology*, vol. 4, p. 265–270, Apr 2009.

[42] S. Lindsay, "The promises and challenges of solid-state sequencing," *Nature Nanotechnology*, vol. 11, p. 109–111, Feb 2016.

[43] M. D. Fischbein and M. Drndić, "Electron beam nanosculpting of suspended graphene sheets," *Applied Physics Letters*, vol. 93, p. 113107, Sep 2008.

[44] J. Nilsson, J. Lee, T. Ratto, and S. Létant, "Localized functionalization of single nanopores," *Advanced Materials*, vol. 18, p. 427–431, Feb 2006.

[45] J. Gierak, A. Madouri, A. L. Biance, E. Bourhis, G. Patriarche, C. Ulysse, D. Lucot, X. Lafosse, L. Auvray, L. Bruchhaus, and R. Jede, "Sub-5nm fib direct patterning of nanodevices," *Microelectronic Engineering*, vol. 84, p. 779–783, May 2007.

[46] J. Yang, D. C. Ferranti, L. A. Stern, C. A. Sanford, J. Huang, Z. Ren, L.-C. Qin, and A. R. Hall, "Rapid and precise scanning helium ion microscope milling of solid-state nanopores for biomolecule detection," *Nanotechnology*, vol. 22, p. 285310, Jul 2011.

[47] D. Emmrich, A. Beyer, A. Nadzeyka, S. Bauerdick, J. C. Meyer, J. Kotakoski, and A. Gölzhäuser, "Nanopore fabrication and characterization by helium ion microscopy," *Applied Physics Letters*, vol. 108, p. 163103, Apr 2016.

[48] Y. Zhang, Y. Miyahara, N. Derriche, W. Yang, K. Yazda, X. Capaldi, Z. Liu, P. Grutter, and W. Reisner, "Nanopore formation via tip-controlled local breakdown using an atomic force microscope," *Small Methods*, vol. 3, p. 1900147, Jul 2019.

[49] T. St-Denis, K. Yazda, X. Capaldi, J. Bustamante, M. Safari, Y. Miyahara, Y. Zhang, P. Grutter, and W. Reisner, "An apparatus based on an atomic force microscope for implementing tip-controlled local breakdown," *Review of Scientific Instruments*, vol. 90, p. 123703, Dec 2019.

[50] C. Buchner, "Pymeasure." `https://github.com/pymeasure/pymeasure.git`, 2015.

[51] D. Fologea, J. Uplinger, B. Thomas, D. S. McNabb, and J. Li, "Slowing dna translocation in a solid-state nanopore," *Nano Letters*, vol. 5, p. 1734–1737, Sep 2005.

[52] Z. Yuan, Y. Liu, M. Dai, X. Yi, and C. Wang, "Controlling dna translocation through solid-state nanopores," *Nanoscale Research Letters*, vol. 15, p. 80, Dec 2020.

[53] K. Yokota, M. Tsutsui, and M. Taniguchi, "Electrode-embedded nanopores for label-free single-molecule sequencing by electric currents," *RSC Advances*, vol. 4, p. 15886–15899, Mar 2014.

[54] U. F. Keyser, B. N. Koeleman, S. van Dorp, D. Krapf, R. M. M. Smeets, S. G. Lemay, N. H. Dekker, and C. Dekker, "Direct force measurements on dna in a solid-state nanopore," *Nature Physics*, vol. 2, p. 473–477, Jul 2006.