POLITECNICO DI TORINO

Master of Science in Electronics Engineering Micro and Nanosystems

Master Thesis

Design and Standardization of a MolFCN Based Cell

From a Single Standard Cell to a Fully Automated Molecular Logic Toolchain

Supervisors
Prof. Gianluca Piccinini
Dr. Yuri Ardesi
Dr. Federico Ravera

Candidate
Ten. Eleonora Girardi
Student ID: s316794

December 2025

Compiled with LaTeX on November 27, 2025.

Abstract

Nowadays making tiny device and simulating to reach upgrades it has pushed Molecular Field-Coupled Nanocomputing - called MolFCN - into focus; here bits aren't stored with usual currents like in CMOS but through molecular charge alignment, flipped by close-range electric pushes. In MolFCN, nearby units interact via electric push-pull forces; timed background fields gently tilt energy levels to set, move, or hold those alignments. Because charges don't have to travel far through wires, energy waste drops sharply, wiring gets easier, plus devices can shrink way further. Instead of typical logic gates, decisions come from group influence and flipping actions; data moves along multi-step pathways that guide polarization shifts while resisting glitches from heat or tiny flaws.

In this setup, compounds able to switch between two stable charge states work well at normal temps. Of these, bisferrocene stands out - thanks to its ability to shift between mixed-valence forms and maintain tight internal links, which builds a reliable electric dipole tied straight to MolFCN polarization. The study uses these traits to build a basic unit - a tiny structure controlled by electrodes that sets, holds, and detects molecular polarization - shaped and made with materials picked to boost coupling, cut unwanted effects, and keep enough energy separation when exposed to real-world electric fields.

In this case study has been set up a MATLAB-driven Graphical User Interface that builds complete 3D device shapes, then spits out input files for Synopsys Sentaurus where SDE handles structure and mesh, SDevice deals with electrical and thermal behavior, while SVisual takes care of results visualization. From a base cell layout - one with a Cut-Y version to test spacing and wiring limits -it has been expanded the system into practical building blocks: a three-phase bus adapted for N-phase use, along with a standard majority logic gate.

Simulation outcomes match MolFCN predictions for the chosen test setup, showing proper field coupling along with timed signal movement. We looked at electric fields, current density patterns, how layer structure affects performance, also heat behavior - highlighting usable operation ranges plus dependence on shape. Taken together, this work delivers: (i) a flexible, automation-friendly process linking CAD models to TCAD simulations for building MolFCN devices; (ii) real-world evaluation of a bis-ferrocene-based logic cell, including its Cut-Y version, the N-Phase bus and the Majority Voter gate; besides scalable upgrades for N-phase wiring and majority voting circuits - offering solid groundwork for lab testing and structured research into nanoscale data transfer.

"Music can change the world because it can change people"

-Paul Hewson

Acknowledgements

I would like to express my gratitude to my thesis advisor, Prof. Gianluca Piccinini, for letting me dive into his research field, this chance shaped the whole idea behind my thesis. Because of his guidance, this project took form, linked to bigger ideas in molecular and nanoscale computing. I'm just as thankful to Dr. Yuri Ardesi along with Dr. Federico Ravera, each brought sharp know-how, steady support, didn't vanish when needed. Input from them, starting back during first planning talks right through sorting out simulated outcomes, made all the difference shaping a rough thought into something solid that actually holds together. A big thank you to the QNANO Molecular Tech Team, being part of their science community really shaped this project. Folks close to me, family, friends, have stuck by without fail through good and rough patches alike. Whether things got tough or just plain busy, they stayed around, keeping spirits up when it mattered most.

Contents

List of Tables

List of Figures

List of acronyms and abbreviations

AC	Atomic Charge
CMOS	Complementary Metal-Oxide-Semiconductor
ECLK	Electric Clocking Field
EF	Electric Field
EP	Electrostatic Potential
FCN	Functionalized Charge Networks
HDL	Hydrogen Depassivation Lithography
HOMO	Highest Occupied Molecular Orbital
HP	High Performance
ITRS	International Technology Roadmap for Semiconductors
LP	Low Power
LUMO	Lowest Unoccupied Molecular Orbital
MolFCN	Molecular Field-Coupled Nanocomputing
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MQCA	Molecular Quantum Cellular Automata
MV	Majority Voter
QCA	Quantum-dot Cellular Automata
SAM	Self-Assembled Monolayers
SDE	Sentaurus Structure Editor
SDevice	Sentaurus Device
SL-3P	Single-Line 3-Phase
STM	Scanning Tunneling Microscopy
SVisual	Sentaurus Visual
TCD	Total Current Density
VACT	Vin-to-Aggregated Charge Transcharacteristic

Chapter 1

Introduction

The tech progress over the last half century lined up pretty much with what was predicted earlier, though some twists came along the way. Still, most changes moved in sync with early forecasts despite surprises popping up now and then due to Moore's observation [17]; this idea suggests that the count of transistors inside a chip tend to double about every 24 months (fig. 1.1).

That idea, above all, it's worked like a starting point for an understanding between three main groups: coders, circuit creators, or chip makers. Still, there's another way to say it Moore's Law highlights a significant physical barrier faced by the Silicon Industry, the continual shrinking the tiniest part inside a chip, the stretch electricity travels through, by One-third each trio of years [17]. Gordon Moore once said in a chat that the tiny transistor's dimensions, specifically the smallest As size gets close to atomic levels, it hits a basic limit [17].

The next big issue that comes from making transistors smaller is linked due to energy use. Over the last ten years, chip elements along with speed saw steady growth across every tech version. Still, because of limits in energy output, one of these factors started slowing down its rise. So they chose to limit how often something happens, recently stuck at just a few billion cycles per second .

Looking at how things have shifted lately, plus what's driving the changes since the update took hold, over the past twenty years, finding new ways to boost typical results, driven new ideas start to show up. In the context of Beyond CMOS, the concept of Functionalized Charge Networks (FCN) introduces a big change in how gadgets are used, instead of just being basic tools gadgets that manage power flow, they're seen as "organized charge holders". The gadgets are set up on purpose so they interact by linking, making movement possible yet handling data [18] [19] [20].

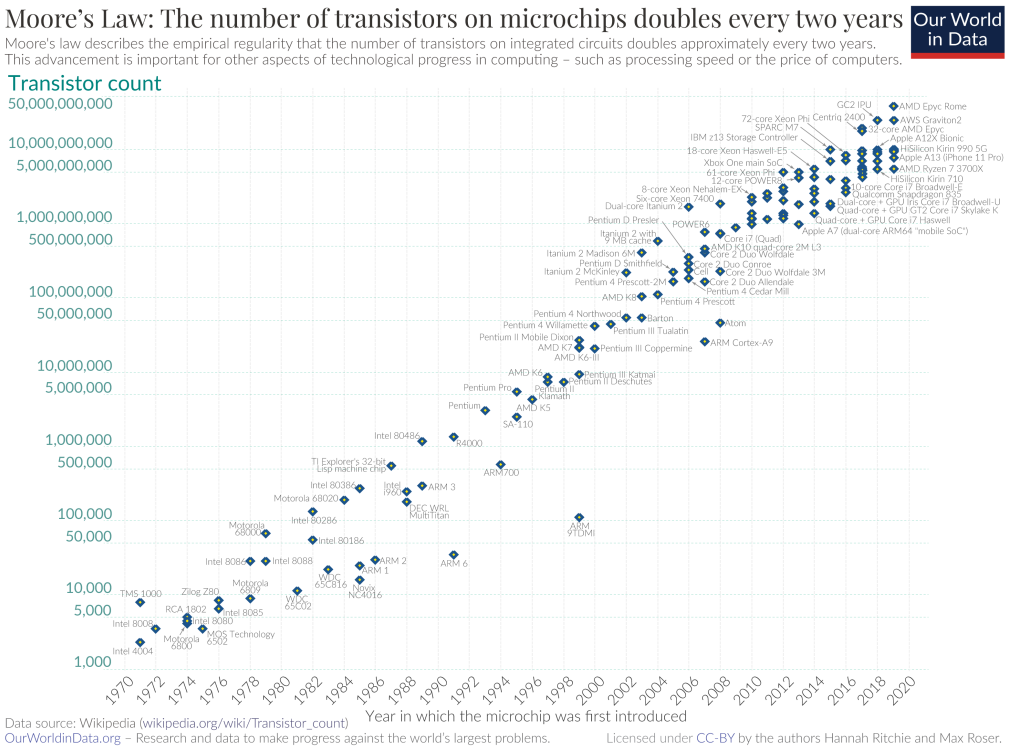


Figure 1.1: Plot of transistor counts for microprocessors against dates of introduction, nearly doubling every two years [17]

A key breakthrough in Molecular Field-Coupled Nanocomputing uses tiny quantum dots, like single silicon atoms, for basic computing tasks. Because they're carefully built, these dots can guide one electron at a time while using almost no energy, unlike older CMOS systems. The main ways things work involve are two:

- *Storing data using electric charge*: info is held where the charge sits inside the system, not by how it moves through wires.
- *Confinement Vessels for Charge*: 3D trapped quantum dots hold single charges without leaking them. These spots have been used before in devices like single-charge switches, light emitters based on dots, sensors that catch light, or setups meant for processing quantum info.

The FCN setup works with different types of quantum dots, each one brings its own perks. Regular quantum dots are tiny bits of material that have been used for years to handle computing without wires. Instead, they rely on fields that interact up close. Molecular Quantum Dots are tiny built-up clusters acting like quantum dots, offering a different path through chemical design for making FCN gadgets, built using precise molecule arrangements instead of traditional methods. Magnetic Quantum Dots instead, are one type where dots link through magnetic fields instead of electric ones, opening fresh possibilities and uses.
In short, the Molecular FCN idea flips traditional silicon electronics on its head. Instead of relying only on silicon, it uses different kinds of quantum dots to arrange and control electric charge right down to individual atoms. This could massively boost how much computing power fits in tiny spaces while using way less energy. It's paving new roads for next-gen chips that go far beyond today's standard tech.

1.1 Quantum-dot Cellular Automata.

In line with QCA ideas, the basic unit is a square holding six tiny dots, where two shifting charges, like electrons or holes, move across through tunneling [26]. Figure 1.2 gives a cleaned-up view of a perfect QCA cell: dots sit at corners and stretch along both diagonals, acting as spots to store on/off data; meanwhile, the inner pair, set halfway along opposing edges, stay set aside for inactive "NULL" mode (fig. 1.3).

Depending on where the two moving charges sit, the cell takes one of three clear forms. Logic "1" or "0" shows up when both charges settle into spots on the same diagonal, thanks to push from their electric force, they end up in opposite corners. The "NULL" form happens if those charges get pushed toward the middle dots by an outside voltage, which works like a timing pulse. Even though "NULL" doesn't hold real data, it's key for smooth, low-energy shifts in QCA setups [27].

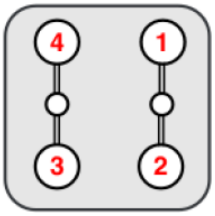


Figure 1.2: Standard Logic dots indexing in a QCA cell. [12]

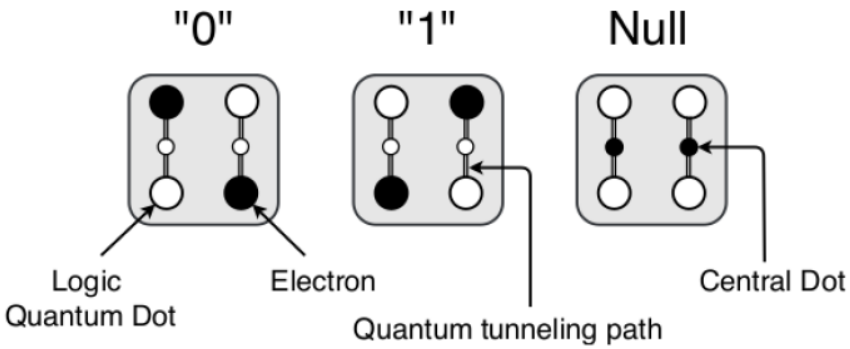


Figure 1.3: QCA cell structure and logic state encoding.[12]

Numerous test molecules were built to try out tiny QCA systems. A well-known case is the double ferrocene type seen in Fig. 1.4 [28]. These two ferrocene parts act like small switches holding on/off info. Meanwhile, a middle carbazole piece holds the neutral or "no signal" setup. Instead of linking randomly, they use an alkyl arm ending in –SH, so it sticks neatly onto gold surfaces by forming SAMs [29]. Even though that center bit doesn't conduct electricity well, the full molecule still reacts strongly to outside electric signals; those shifts can steer where the topmost electron sits, so you can set it to mean “0”, “1”, or “NULL” (fig. 1.3).

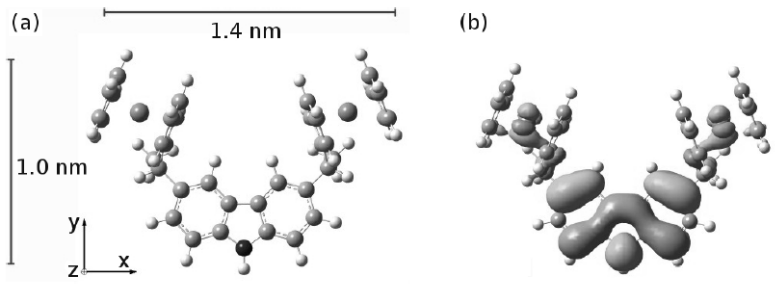
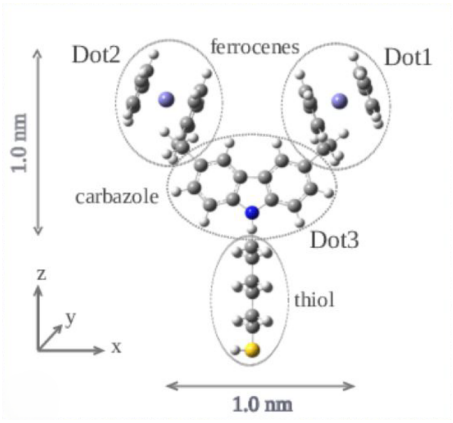
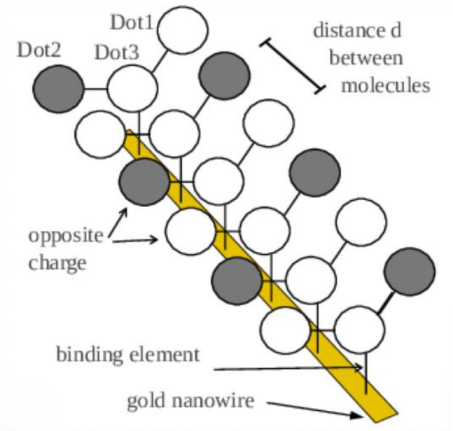


Figure 1.4: Bisferrocene Molecule [33].

A single bis-ferrocene unit has just three dots,so it acts like a QCA half-cell (fig. 1.5). To get a full MQCA cell, you link two of these up closely (see Fig. 1.4). In this project, it is assumpted that they're lined up perfectly along a flat nanowire. But in real tests, bumps on the surface or crowding during SAM setup might shift their positions [30]. Information moves in QCA through electric push-pull forces; if molecules sit crooked, that connection gets weaker and less reliable.



(a) Molecule structure and properties [33].



(b) Molecules distribution on a gold nanowire [33].

Figure 1.5: Both images show the bis-ferrocene molecular system used in molecular QCA studies [12].

1.1.1 Principle of Information Propagation

In field-coupled nanocomputing, data moves because tiny QCA units are placed just the right distance apart. Nearby cells have their quantum dots spaced evenly, not only within each unit but also between them. The first cell in the line works like a starter, set by something outside (an Input), an electric push or a built-in trigger, to show either a "1" or a "0". Because of electric pull between parts, the free charges in the next cell shift to copy the diagonal setup of the one before.

To understand the mechanism of information transfer in QCA, it is essential to analyze the electrostatic coupling between two neighboring cells (see fig. 1.6, 1.7. When a driver cell (A) and a target cell (B) are placed in close proximity, the fixed charge configuration (polarization) of the driver induces the mobile charges in the target cell to rearrange such that both cells adopt the same polarization. This aligned configuration corresponds to the global electrostatic energy

minimum of the two-cell system [12].

The strength of this *cell* ⇔ *cell* coupling is quantified by the so called **kink energy** E_{kink}, which represents the additional electrostatic energy penalty incurred when two adjacent cells have opposite polarizations (fig.1.6).

For simplified four-dot QCA cells (each containing two mobile electrons that can occupy any of the four corner dots), the kink energy is defined as

$$E_{\text{kink}} = E_{\text{anti-aligned}}, E_{\text{aligned}} \ , \tag{1.1}$$

where $E_{\text{anti-aligned}}$ is the electrostatic energy when the two cells have opposite polarizations, and E_{aligned} is the energy when they have the same polarization.

In the standard two-cell Hartree approximation [12, 30], this difference simplifies (for identically sized cells at typical nearest-neighbor distances) to

$$E_{\text{kink}} \approx \frac{1}{4\pi\epsilon_0\epsilon_r} \cdot \frac{q^2}{d} \cdot K \ , \tag{1.2}$$

where

- $q = |e|$ is the elementary charge,
- d is the center-to-center distance between the two cells,
- ϵ_r is the relative permittivity of the medium,
- K is a geometric factor (typically $K \approx 0.2$–0.3 depending on exact dot positions).

More rigorously, the kink energy is obtained by summing the Coulomb contributions over the dominant charge-site pairs:

$$E_{\text{kink}} = \sum_{i,j} \frac{1}{4\pi\epsilon_0\epsilon_r} \frac{q_i q_j}{r_{ij}}\bigg|_{\text{anti}}, \sum_{i,j} \frac{1}{4\pi\epsilon_0\epsilon_r} \frac{q_i q_j}{r_{ij}}\bigg|_{\text{aligned}} \ . \tag{1.3}$$

The resulting positive E_{kink} energetically favors alignment of the target cell with the driver.

For reliable operation at room temperature, the thermal energy $k_B T$ must be significantly smaller than E_{kink} (typically $E_{\text{kink}} \gg 5k_B T$) to suppress thermal excitations into erroneous anti-aligned states. This requirement largely dictates the minimum feasible cell size and spacing in any practical QCA implementation.

In relation to this, one after another, this pattern moves forward through the row, much like falling blocks tipping their neighbors. The setup works through fields only, no actual charge moves from cell to cell. Rather, every unit reacts right away to the electric push coming from nearby units that have already shifted. The most stable layout, the one where electric pushes are weakest, happens when all units line up just like the starting one [12].

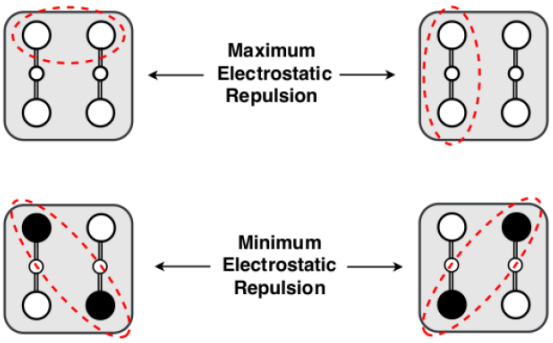


Figure 1.6: Maximum and minimum electrostatic repulsion conditions in QCA cell [12].

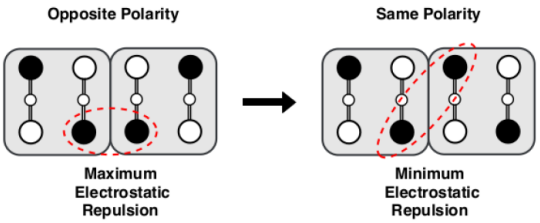


Figure 1.7: State transfer between two adjacent cells [12].

This lack of charge movement between cells is what makes QCA use so little power. Instead of drawing energy from flowing currents or loading up wires like regular CMOS circuits do, QCA works differently. In a QCA line, energy is only used when electrons jump inside a cell to flip its state diagonally. That internal shift needs very little power, about $10^{(-20)}J$ for tiny quantum dots. Compared to the roughly $10^{(-18)}J$ burned each time a smallest possible transistor switches, it's around a hundred times less [31].

Figure 1.8 shows how it works: once the driver cell settles into either on or off mode, every next cell follows along naturally, syncing up to hit the lowest energy point across the system; this keeps signals moving steadily forward using almost no power, even though electrons don't actually travel from one unit to another [31, 12].

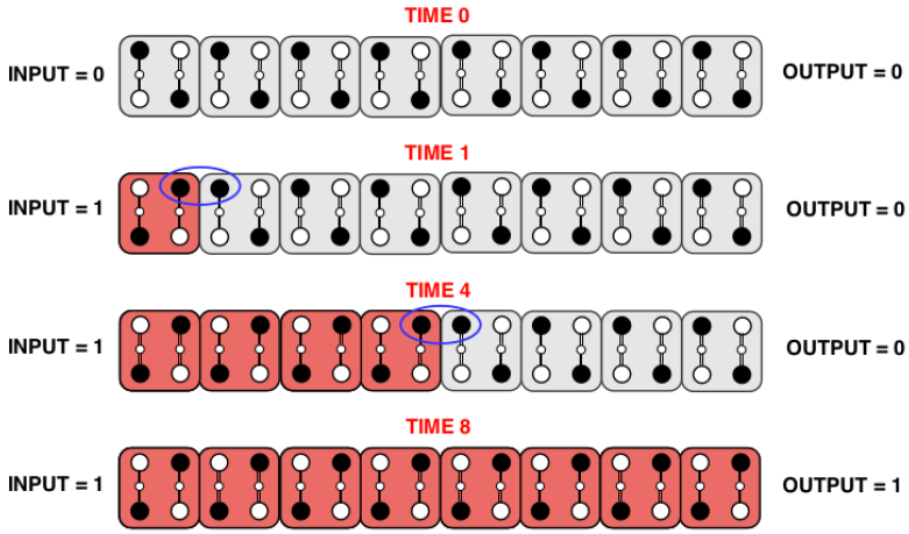
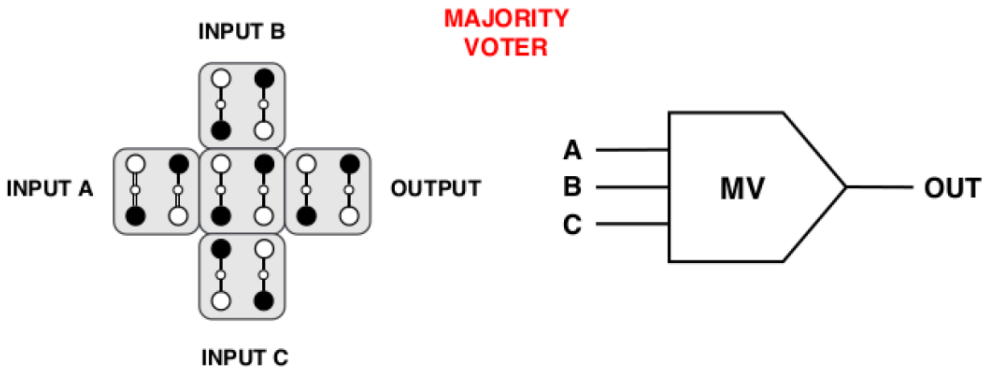


Figure 1.8: Propagation of information in an FCN Wire [12].

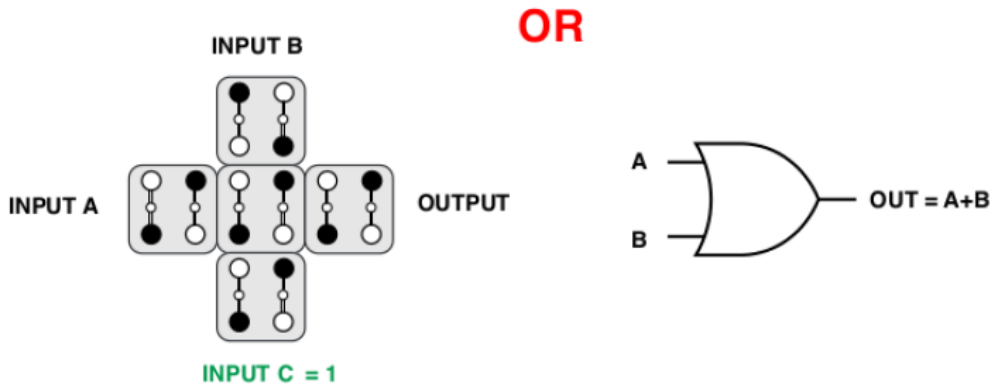
The highest molecular level holding electrons, the HOMO,is where valence electrons sit before getting pulled away during oxidation; this spot holds the easiest-to-remove ones. On the flip side, the LUMO sits just above as the first empty slot in a neutral molecule, ready to grab an extra electron when reduction happens. That space between them,called the HOMO–LUMO gap, or $E_g = E_{\text{LUMO}} - E_{\text{HOMO}}$shapes how a molecule handles electricity and light. When the gap's big (usually more than 3–4 eV), molecules tend to stay stable, block visible light, and resist conducting current. But if it's narrow (less than 2 eV), they react faster, soak up visible photons, and move charges better once energized or doped. For tiny circuits like QCAs or molecular-scale electronics, having a broad enough gap keeps heat from shoving electrons across at room temp, so signals labeled "0", "1", or NULL don't blur together while running [12].

1.1.2 Basic Logic Layout

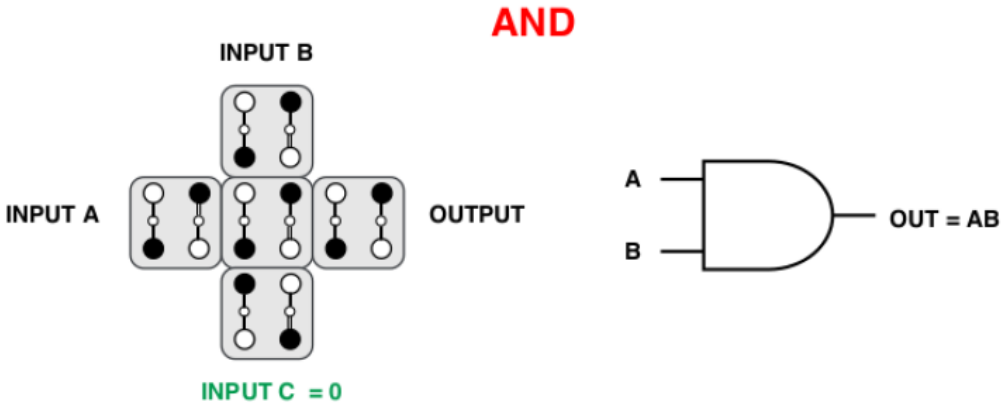
According to the previous section, it is clear that the shape of the layout decides what happens, seeing that signals hide in charge positions across units rather than travelling as currents. The core piece, the strongest one, is the majority voter configuration, pictured in Figure 1.9a. This setup's made up of five cells set in a cross shape. Three are inputs, often marked in pink on diagrams. In the middle sits the device cell doing the real job. Then there's the last part: an output cell. The three input cells affect the middle one through electric forces. Because of this push-pull effect, the center cell naturally takes on the same charge as most of the inputs, so if two or more inputs are "1," its value becomes "1"; if not, it turns to "0." Positioned on one arm of the cross shape, the output cell mirrors whatever state the core holds, showing the final decision clearly [12]. Fixing one of the three inputs to a steady signallike setting it to "1" when using AND, or "0" if using ORthe majority gate acts like a regular two-input AND or OR. That shift lets it serve as a building block, so you can build any logical operation from it instead of needing separate parts.



(a) FCN Majority Voter (MV) [12].



(b) FCN OR based on Majority Voter [12].



(c) FCN AND based on Majority Voter [12].

Figure 1.9: MV implementation (fig. 1.9a). AND and OR implementation from the Majority Voter configuration by setting input C = 1 (fig. 1.9b) and C=0 (fig. 1.9c). The thruth tbale is shown in 1.1.

Table 1.1: Truth table of the three-input majority gate used in QCA

C	B	A	OUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

1.1.3 Clocking System

Coulomb repulsion sets up binary data in one QCA cell while also pushing signals through chains of connected cells. Still, in extended lines made from many linked units, electric-only transfer might failsometimes getting stuck in semi-stable setups where flipped polarizations create "kinks," breaking correct logic output [32]. These hiccups need fixing; without a timing control method, steady forward movement of info wouldn't work reliably.In the basic QCA setup, the clock doesn't hold data but shifts how easily electrons move between dots inside a cell. Instead it tweaks those movement paths, by lifting or dropping them, to decide what the cell does: either stay fixed in one position (on mode), settle into a shared setting based on nearby cells (working/idle phase), or get reset completely to neutral (reset/change stage) [12]. With the standard six-dot design, neutrality happens when the timing signal pushes both free charges into extra center dots, wiping out old values so the unit can pick up fresh signals from adjacent units. So real-world QCA wires split into chunks, called clock zones, as shown in Fig.1.10. Usually, there are four sections, each powered by a different timing signal (0°, 90°, 180°, 270°). Right at any time, side-by-side segments sit in opposite modes, one might be on and locked, the next sits idle waiting for data, another processes it, and so on. This slow energy shift pushes a voltage ripple forward through the line, moves info one way only, smooths out glitches, boosts reliability over long stretches, yet keeps power use super low like the tech promises.

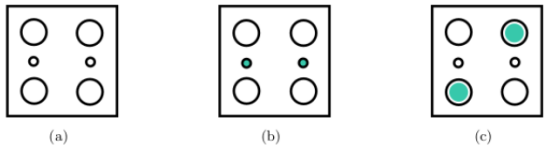


Figure 1.10: Schematic of enriched QCA cells: (a) QCA basic cell with the six quantum dots represented, (b) QCA basic cell in the Null state or Reset State, (c) QCA basic cell in one of the two stable states [12].

By applying a clocking field along the wire, information propagation becomes fully controlled and unidirectional. The standard QCA clocking scheme employs four successive phases, each corresponding to a specific modulation of the inter-dot potential barriers within the cells:

- **Switch phase**: the potential barriers are gradually raised, allowing the cell to adiabatically transition from the NULL state to one of the two binary logic states under the influence of the electrostatic field produced by the previous (already polarized) zone.

- **Hold phase**: the barriers are kept high, locking the mobile charges in the corner dots corresponding to the logic state; the cell maintains its polarization and acts as a driver for the following zone.
- **Release phase**: the barriers are progressively lowered, permitting the charges to move adiabatically back toward the central (null) dots, thereby erasing the previous logic information.
- **Relax phase**: the barriers remain low, keeping the cell fully depolarized in the NULL state with charges confined to the central dots and ready to receive new information during the next switch phase.

These four phases are combined to form a complete quasi-adiabatic clock cycle, which is typically phase-shifted by 90° between consecutive clocking zones along a wire. A schematic of the resulting four-zone pipeline and the associated clock signals is shown in Fig. 1.10 and Fig. 1.11 [12].

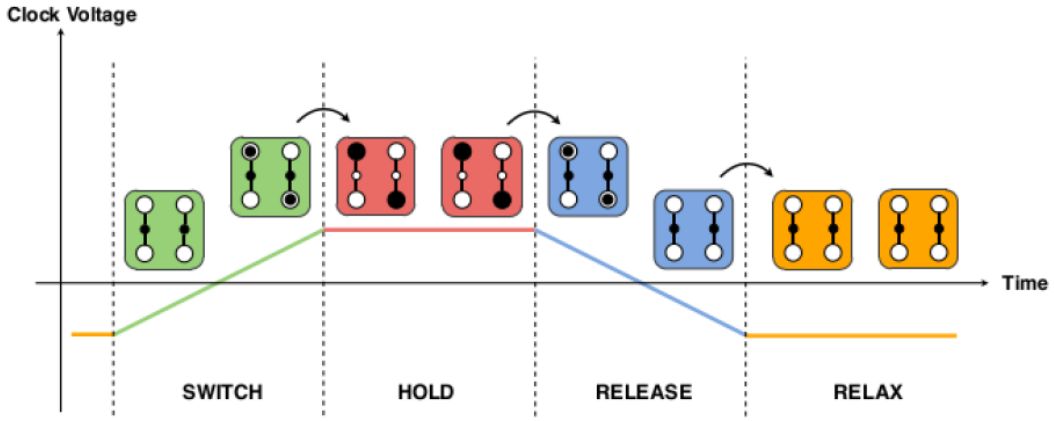


Figure 1.11: SHRR CLK Phase [12].

Implementation of this clocking scheme requires cells with six quantum dots (four corner dots for the logic states and two additional central dots for the NULL state), as illustrated in fig. 1.12. The presence of the central dots enables the explicit NULL configuration that is essential for power-efficient, error-free adiabatic switching and long-range signal propagation.

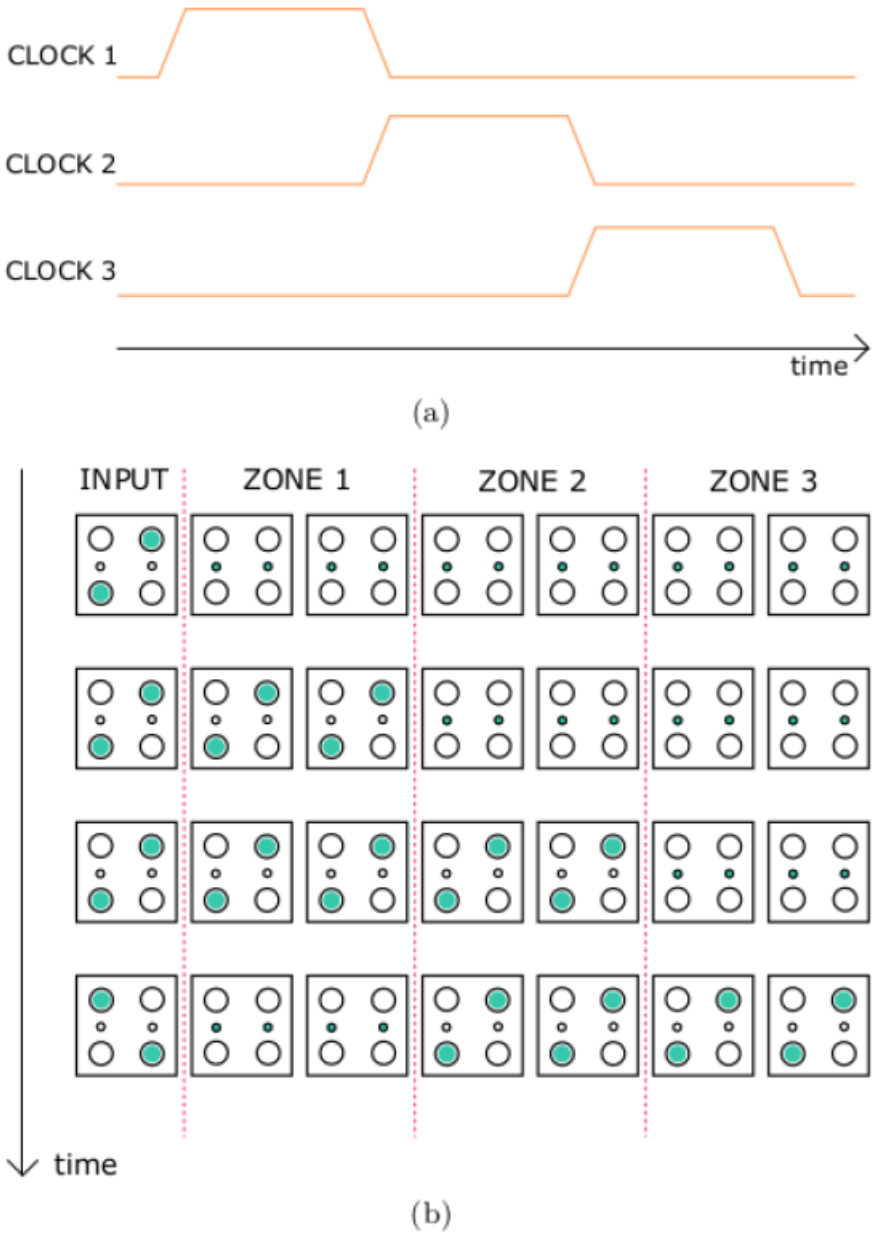


Figure 1.12: Schematic representation of the information propagation as a pipeline: (a) trapezoidal clock signals overlapped to ensure correct propagation, (b) example of the first time steps in a wire. [12]

1.2 Molecular Field Coupling Clock

While plenty of research focuses on theory and tiny components in field-coupled nanocomputing, very few look at real-world hurdles when building it from molecules. Most methods use specially shaped surfaces that act like rails or blocks where molecules stick neatly into place, usually through self-assembled monolayers (SAM). Yet active parts for molecular QCA are just 1–2 nm wide, meaning connections and guides must match that scale, far smaller than today's best chip-making tools can manage. Even assuming we could shrink those processes further, placing elements precisely within less than a nanometer near tight turns stays out of reach using current fabrication approaches. Besides this, flawless self-assembled layers need perfectly smooth and uniform surfaces across broad regions; otherwise flaws in the guide structures lead to shifted molecules or mismatched zones, weakening how well neighboring units influence each other electrically.
As reported in [12], in tiny quantum setups, energy levels in dots spread out because of time-energy uncertainty, this shift changes how packed the states are while altering chances for electrons to move through; the pattern ends up shaped like a smooth curve around the starting energy point. A look at MQCA setups in fig. 1.13 shows a groove built to hold molecules over a hidden wire. At the groove's sides, electrodes use electric fields, shaping molecule behavior while syncing operations. Instead of direct links, forces guide changes through spaced interactions. These side elements manage timing by shifting local conditions near the trench. A new idea called a "guiding wire" shows upit's a tiny structure helping position molecules accurately so data moves steadily. Making these gadgets needs super sharp cuts in the material, often using silicon bases with hidden metal sheets to build the necessary electrical parts.

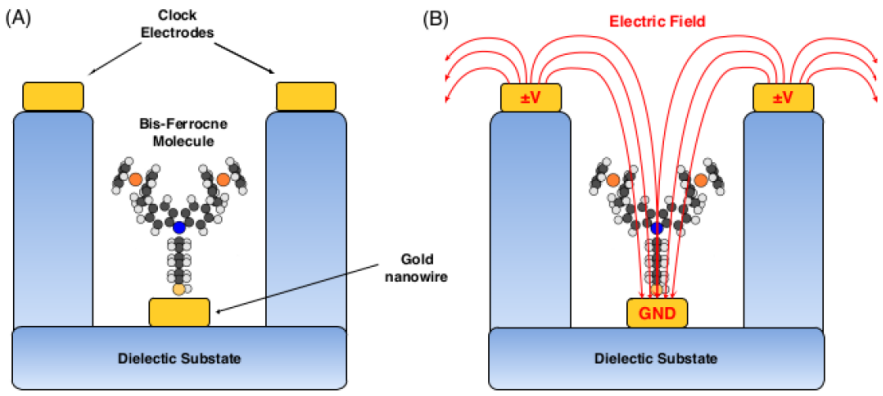


Figure 1.13: (A) Scheme of physical implementation of MQCA clock system (B) Almost vertical Clock field generated by the designed clock system [12].

Applying a voltage between the electrodes on the top and at the bottom of the trench, a vertical electric field is generated over the molecule. The expected vertical field depends on the aspect ratio of the trench [12]. In real systems where nearby electrodes mess with one another because their fields blend together, the aim's to keep them separate, building up signals helps keep data safe. Timing the wave patterns together works out better then there's the problem with areas taking up the same spot. Another problem comes from how much space is needed between control electrodes, shown clearly in Figure 1.14. The ideal gap is around 1 nm, which matches how close molecules sit side by side; that way each one can be targeted accurately. But making stable gaps this tiny isn't possible yet with today's tools. When we use bigger spaces instead (tens of nanometers) some molecules end up too far from any electrode pair and seem

impossible to reach. Still, because electric fields from nearby electrodes spill into shared areas, the issue softens: even distant molecules feel enough field strength to switch states properly when timed right.

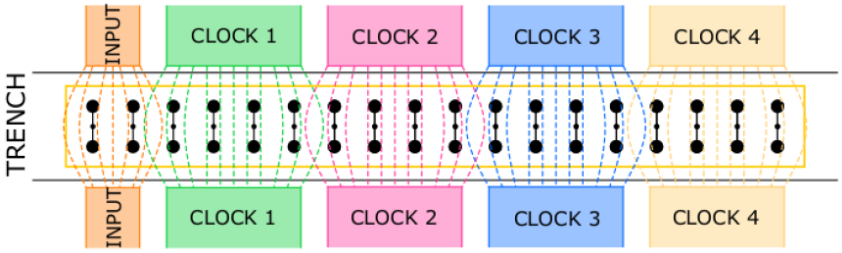


Figure 1.14: Clocked molecular FCN nanowire implementation.

According to this, to generate adiabatic propagation of information in the MQCA device, for example, an MQCA Wire is divided into three clock zones. That's because it uses three separate contacts; using three, rather than four, helps reach a setup without zones. Following the clock areas, three golden electrode clocks per side are laid out, while inside every one of them electrodes get a shifting trapezoid signal at once, yet each one runs out of sync (fig. 1.15).

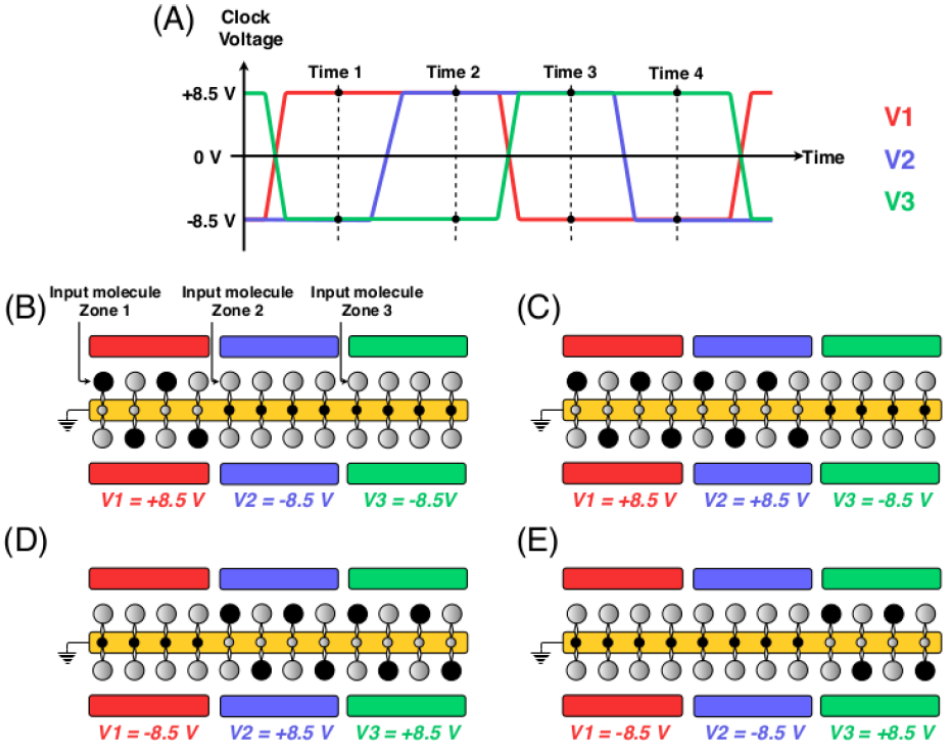


Figure 1.15: Schematic example of information propagation in MQCA Wire

Although some field-coupled nanocomputing (FCN) technologies have already been demonstrated experimentally, for instance, aluminum-based metal-dot QCA cells operating at cryogenic temperatures and nanomagnetic logic (NML) wires in which clear signal propagation has been

observed, a fully functional molecular prototype has yet to be realized. The principal obstacles remain the need for sub-nanometer patterning precision and the difficulty of reliably detecting the minute charge configuration inside individual molecules.
Nevertheless, several molecules tailored specifically for molecular FCN, such as bis-ferrocene and diferrocenyl-carborane derivatives, have been successfully synthesized. Substantial progress has also been made on the fabrication front, with techniques including hydrogen depassivation lithography (HDL), nanoshaving, scanning tunneling microscopy (STM) patterning, electron-beam lithography, and DNA-origami-based nanopatterning all showing considerable promise for the assembly of molecular-scale circuits. Complementary advances in producing ultrathin nanowires now provide viable electrodes and clocking lines [12]. A particularly encouraging breakthrough is IBM's demonstration of direct charge imaging on single molecules using Kelvin-probe force microscopy. Taken together, these developments have brought a working molecular FCN device significantly closer to experimental reality. SAMs have become a key technique for depositing molecules to create thin organic films. A SAM is simply a highly ordered layer of molecules that forms all by itself when suitable molecules are brought into contact with a surface. Each molecule has a "head" group that strongly sticks to the substrate and a tail that points outward. The most common head groups are thiols (which links very well with gold and other metals), silanes (great for glass or silicon oxides), and phosphonates. Choosing the right head group ensures the molecules attach firmly and pack neatly into a uniform film. This simple yet powerful method is widely used in molecular electronics, sensors, and nanotechnology [29].
Picking the right molecule depends on what you're building it on and what job it needs to do. A well-known group often used is alkanethiolstake ferrocenyl-hexanethiol, which ends in a sulfur link but carries a ferrocene part on the far side. This one's common in tiny circuits since its ferrocene bit can switch back and forth between states, making it useful for things like nano-scale switches or devices that sense chemical changes.

1.3 Fabrication Layer

To build accurate molecule setups, like circuits, logic gates, or tiny wires, a smooth, even surface just won't work anymore. The base material needs custom shaping or specific chemical tweaks so molecules settle exactly where they need to be, lined up right for working nano-scale computing systems. A good method starts by making a thin gold guide line on the surface, its width, height, and span carefully set. Then, using an earlier technique, a special layer of tiny organized molecules is placed just on that gold strip. This way, active bits stick exactly where they should go, shaping the intended QCA layout. The way the molecules line up in the end depends on a few main things: first, how the gold surface's atomic structure guides where sulfur-based groups stick best; next, the angle those sulfur anchors take when they settle into place while forming the monolayer; also, the large size of the bis-ferrocene parts gets in the way, restricting how much the molecules can twist and pushing them toward particular positions instead of random ones.
Scientists are looking into trickier shapes (like bends and T-junctions) to make working logic switches and full circuit setups. In a bent wire shaped like an L, getting the signal through the turn depends mostly on aligning the gold crystal just right, since that affects how the molecules angle and fit together at the curve. Some molecule positions might be required to keep connections between cells solid and stop the signal from weakening. A single molecule's size is around 1 nm, so the best wire width would match that. Still, thicker wires (say, tens of nanometers) work just fine in real setups. That's because many molecular rows side by side can act like one strong QCA path.In situations where achieving high resolution for corners with gold patterning is challenging, nonideal conditions may lead to additional molecules being attached at corners.
A popular way to control molecular QCA setups uses hidden wires placed inside grooves cut into an insulator. These grooves are made with precise size, both wide and deep enough for accuracy.

Then, a narrow strip of gold, carefully shaped, is laid down at the base. This metal line acts like a path, directing how the molecules line up in neat rows. Clocking works when a voltage shift ΔV is applied through electrodes placed at each end of the trench, or along its sides. That creates an electric field with lines moving straight down the trench's length. Switching the sign of ΔV flips the field's direction, either way, so molecules resting above the gold line get pushed into a target logic setting or reset to NULL, each time under full control. Because of this setup, with wires hidden inside trenches, alignment stays sharp while also offering a solid, expandable path for sending clock signals.

1.4 Thesis Structure

The thesis is built step by step on purpose, matching how real research usually unfolds when simulating tiny devices, yet it also works like a learning path for newcomers trying to grasp the mixed-up world of molecular computing at nano level.

Chapter 1 sets up the real-world groundwork needed to make sense of everything else. Introduction chapter is important since if it is not clear why standard CMOS tech can't shrink much further, how field-driven methods could slash energy use by huge margins, or what makes molecule-based systems the peak goal for QCA, then later design efforts feel pointless. So instead, this part ties together ideas from chip physics, quantum-level chemistry, tiny-scale manufacturing hurdles, along with core info theory, then leads into where the rest of the paper's headed.

Chapter 2 doesn't just summarize past research, it sets the stage. One part shows where current science stands, pointing out what's already been tested, like metal-dot or nanomagnetic QCA systems, while also underlining missing pieces, especially molecular designs. Then there's another angle, introducing Sentaurus TCAD alongside a self-built MATLAB automation setup, tools that together act as this project's virtual lab. This isn't about listing software; it digs into why an off-the-shelf chip simulator, made for conventional silicon tech, works here even at the molecular level. Plus, it reveals how tailored scripts fix its shortcomings when handling features smaller than 5 nm.

Chapter 3's where the real tech work kicks in, built step by step so anyone can follow along. Instead of skipping details, it spells out everything: exact shapes, doping setups if needed, how the grid's set up, which physics models run, even the full SDE script. That way, someone using Sentaurus could rebuild it from scratch. After that, trickier designs come next, like Cut-Y cells or multi-phase grids, majority voter, T wire, and L wires, but without rehashing basics. Each one just focuses on what's different: tight corners messing with signals, managing phase edges, interference when parts get packed close. This cuts down repeated info while showing how well the setup scales. It also proves how easily pieces fit together across bigger systems.

Chapter 4 shifts things up, turning scattered gadget tests into an actual design hub. The MATLAB tool presented, turns a standard TCAD package into a smooth QCA circuit builder. With it, users sketch any 2D shape, then get automatic 3D buildup plus exportable simulation files. That means newcomers can jump straight into testing advanced setups, like full adders, storage units, maybe tiny CPUs, no need to wrestle Sentaurus code from zero.

Chapter 5 sticks close to the numbers, showing hard metrics, like polarization accuracy, kink energy, power use over time, and needed clock signal strengthwhile also pointing out quirks that came up, such as grid-related glitches, trouble stabilizing under strong fields, or shortfalls in how physics was modeled. Being open about both results and hiccups helps keep things trustworthy.

Lastly, **Chapter 6** ties things together by linking the fine-grained simulation outcomes back to the big-picture goals set at the start. It checks how much this study moves us closer to working molecular QCA tech. The chapter also suggests clear follow-upson one hand lab work like testing real bis-ferrocene monolayers; on the other, deeper modeling that includes environmental effects and atomic jiggles within molecules.

In short, this setup isn't randomit's built on purpose to blend solid science with repeatable results, clear teaching value, while also pushing tools aheadnot just dreaming up ideas. That mix matters most because molecular field-coupled computing is full of potential, yet still new and unproven, needing more than theory to actually work in real life.

Chapter 2

State Of Art

The Molecular Field-Coupled Nanocomputing idea stores ones and zeros using how charges are arranged on single molecules, as seen in the previous chapter 1, while signals move only by electric push-pull forces between neighboring units. Instead of moving current, it relies on static fields, making it way more energy-efficient than regular electronics. Even though scientists have made headway designing these systems in theory and testing them virtually, nobody's built a working model so far.
Nowadays projects tackles this key issue by bringing together finite-element modeling and the SCERPA tool into one simulation approach [33], so it can closely examine both quantum effects at the molecular level and electrical behavior in full devices. This research starts with single-molecule wires placed inside nano-trench setups, carefully testing how consistent signal transmission depends on specific shape features of the trench. Then, it moves on to arrangements with multiple lines, using those results to figure out core design rules for working molFCN models. In the end, the nano-trench setup proves to be a solid and realistic way to handle the input process. Overall, these findings create a solid setup for simulating devices while considering real-world build challenges, this pushes closer to actually making molecular computing gadgets in lab settings.

2.1 Theoretical Background

MolFCN is a method using molecules to act like QCA by holding electrical charges in specific spots as already clarify. Instead of traditional signals, it uses where the charge sits inside a single molecule. Focus lands on the bis-ferrocene molecule, which has three active zones called Dot1, Dot2, and Dot3, these are where the charge settles based on how strong and which way a vertical electric clocking field (ECLK) is pointing. That ECLK isn't just background noise; it acts like a timing pulse, guiding data movement step-by-step across MolFCN setups. Molecular cells under identical ECLK fields get sorted into clocking areas and this allows signals to move straight through molecular wires while also supporting basic logic functions, especially the Majority Voter (MV). Instead of using "and," you could say this setup works thanks to either signal flow or gate operation. Fix one MV input at a steady high or low level, then it acts like an AND or OR gate without extra parts.

Compared to metal or magnetic versions, MolFCN works way better, it runs at normal temps, could switch signals super fast, hundreds of gigahertz, and uses almost no power per molecule. Even though it sounds very stimulating as a subject to focus on, any prototype that actually works has been built yet. To close the current divide between theory and lab results, this study offers a full simulation setup merging FEM-based electrostatics with SCERPA [35]. Instead of treating scales separately, it handles molecular-level chemistry alongside large-scale device performance at

once, leading to much better predictions when building working MolFCN systems. Simulations were set up for MolFCN gadgets, using MoSQuiTo, a tool made for quantum-dot cellular automata work in Turin. This setup connects smoothly with Sentaurus, helping model electric behavior at the circuit level through detailed math methods. The MoSQuiTo approach involves three key steps:

- From scratch, computer-based chemistry guesswork on a possible MolFCN structure.
- Molecular-level simulations followed by pulling out key details.
- Complete circuit modeled through SCERPA software.[35]

A key part of molecular modelling is the "Vin-to-Aggregated Charge Transcharacteristic" (VACT), this links input voltage (Vin) to total atomic charge (AC) on the molecule under a steady clocking field ECLK. Then, that VACT pattern goes into SCERPA so it can estimate how circuits act along with signal movement [35] [33]. The current research looks into the oxidized bis-ferrocene structure, using a refined SCERPA method that calculates the local Vin effect on each unit by including electric forces from nearby units up to a set range (IR). Tests start with a basic three-part wire setup then move step-by-step toward MV logic gates and complex multi-wire layouts. Also, a capacitance-based approach helps gauge energy loss during switching, while changes are tested across insulating types, groove sizes, and traits of the molecules. This combined approach gives realistic performance estimates for MolFCN circuits that account for manufacturing limits, so it helps close the divide between simulation designs and actual lab results.

2.2 Nano-trench Device Description

The main setup studied [33] is a tiny trench design made to fit and manage molFCN systems. Shown in Figs. 2.3, 2.1 and 2.2, plus different versions (ConfA–ConfH), it includes these parts: a small groove cut into a material, either $low-k$ or $high-k$, with height around 3 to 17 nanometers, while its width measures between 3 and 4 nm. A thin strip of gold sits at the base of a groove, its width about 2 to 3 nanometers. Along its middle line, bis-ferrocene units stick down using sulfur-gold bonds; these form either a single row or multiple side-by-side lines spaced roughly a nanometer apart from center to center. Pairs of top electrodes, each around 2 to 6 nanometers wide, ten nanometers long, hang above the groove, placed evenly along both inner edges. A space with little to no material sits between the molecule layer and upper contacts, purposely leaving out any solid insulating stuff in that area.

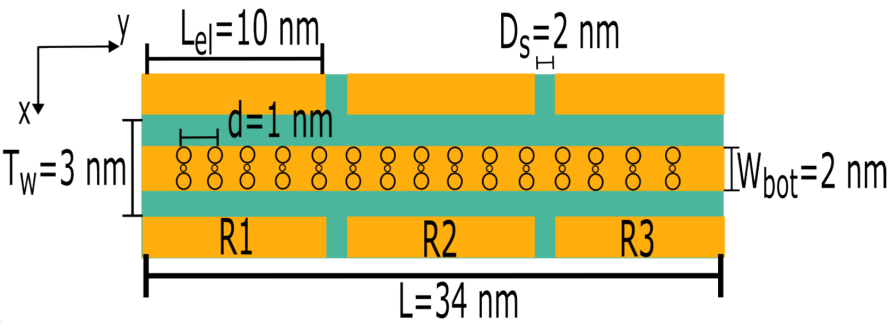


Figure 2.1: Looking down at the setup with the three wires. The tiny particle line up side by sidekeeping molecules spaced evenly at distance “d” across the base conductor. The geometrical settings match ConfA [33].

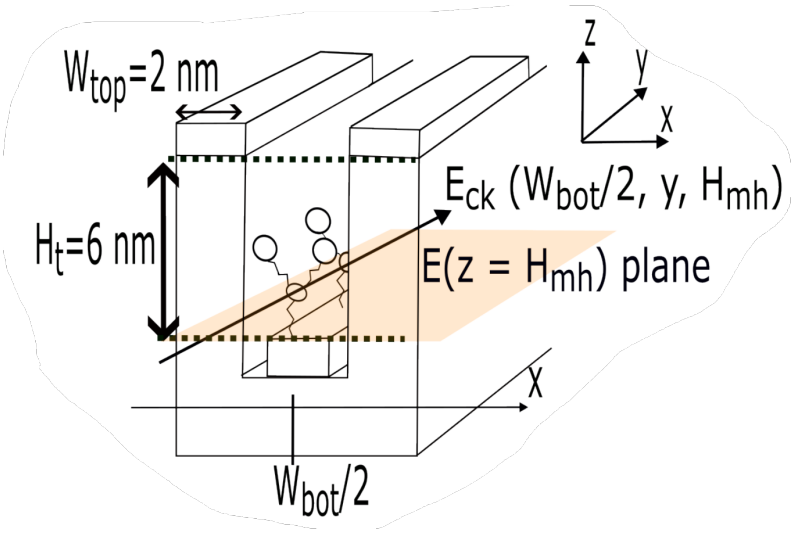


Figure 2.2: Side view sketch of the nano-trench but Eck removal. [33]

Clock signals work by sending small voltages (from $\pm 3V$ up to $\pm 12V$) through the upper electrodes (V_{el}). This creates an electric field mostly pointing straight, along the z-axis, that pushes charges one way or another depending on its strength and direction. When the field's positive, it flips bits into Hold mode: electrons move to Dot1 or Dot2, storing data. But if the field turns negative, things shift; the charge gets stuck in Dot3, wiping out what was there before. Meanwhile, the lower contact stays tied to zero volts.

Because the electrodes are really close, just under 4 nanometers apart in working setups, the needed electric fields (around 1 to 2 volts per nanometer) show up with just a small voltage push; sideways field bits stay tiny near the molecule's centerline. So this setup gives:

- exact clock sections you can control separately down the line,
- works naturally with advanced nanomaking tools like e-beam patterning, nano-stamping, shrinkable layers, or metal coating via ALD,
- a practical, expandable writing setup using special side-mounted drive wires chances are you can use extra lines to stay safe if something fails.

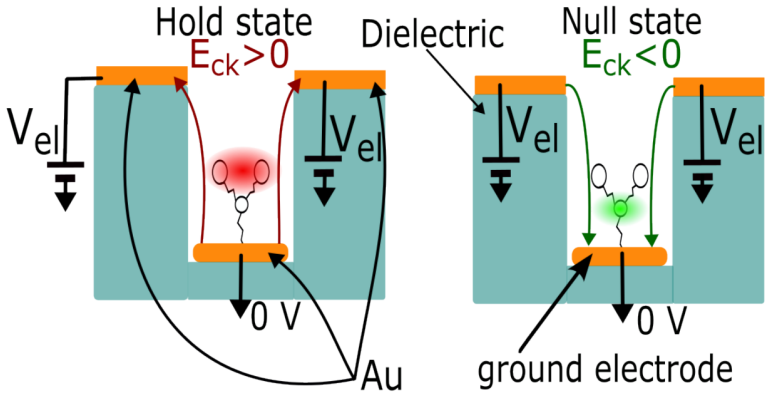


Figure 2.3: The tiny trench idea put forward using method from [11], opposite charges on the plates push particles in null but hold positions, in that order.

This tiny trench setup, along with an upgraded SCERPA+FEM simulation method, forms the first device idea aware of manufacturing limits, linking atomic-level science to real-world molFCN circuits in a solid way [33]

2.2.1 Impact of geometry on information propagation

The main focus here [33] is straightforward: what role do the shape of the nano-trench and where you put the electrodes play in how well data moves through a molFCN wire? Researchers checked out eight different setups, labeled ConfA to ConfH (table 2.1, tweaking one value or more in each while holding the electrode size steady at 10 nm.

Conf.	D_s (nm)	T_w (nm)	W_{bot} (nm)	H_t (nm)	W_{top} (nm)	L_{el} (nm)
ConfA	2	3	2	6	2	10
ConfB	4	3	2	6	2	10
ConfB1	6	3	2	6	2	10
ConfC	2	4	3	6	2	10
ConfD	2	4	3	7	2	10
ConfE	2	4	3	7	6	10
ConfF	2	4	3	12	6	10
ConfG	2	4	3	17	6	10
ConfH	2	3	2	3	2	10

Table 2.1: Geometrical Parameters Of The Simulated Configurations as analysed in [33]

Boosting the space between electrodes D_s, going from ConfA to ConfB and then ConfB1, really hurts how well things work. Bigger distances create intense stray electric fields in spots between neighboring clock sections, messing up the Null state big time. In those messy zones, molecules flip unpredictably, causing errors that travel backward and scramble data stored earlier in prior segments. These findings show plainly: D_s should stay under roughly 4 nm if you want the clock to fully control all the molecules. Slight changes in trench width T_w (ConfA vs ConfC) lead to almost identical clock-field patterns and smooth signal flow, showing that single-line molFCN wires handle minor side-to-side size shifts pretty well. The way trench height H_t connects to top-electrode width W_{top} creates a tricky balance. When trenches go deeper,

meaning higher H_t, near ConfG, the clock field Eclk weakens no matter the phase, making Null and Hold modes shaky unless voltage shoots up, which then pushes power use higher. On the flip side, shallow trenches, with low H_t like in ConfH, produce strong fields using just small bipolar voltages ($\pm 3V$), supporting steady signal movement and cutting energy needs, though this demands tighter precision during manufacturing.
Expanding the upper electrodes, raising W_{top} from 2 nm up to 6 nm, shifting from ConfD to ConfE, boosts the null-state reset in matching clock regions because of increased surface area on the electrode; this improves signal spread while also reducing how tight the sideways precision needs to be for the clock wiring above. Taken together, these tests point to a clear priority in design: how far apart the electrodes are plus how deep the trench is mainly decide if things work right, while the trench's width isn't so picky. A good nano-trench uses little depth, letting it run on less voltage and power, keeps electrode spaces tiny, also includes somewhat wider tops, to help hold that off-state solid. This setup meets tight electronic needs for molecule-based QCA but still fits what today's tech can actually build.

2.3 Two-line wire analysis

It is shifted from a wire having one row of molecules to another with two side-by-side rows packed into the same tiny groove, called a "two-line wire." The thought here is that using more than one line might it loosen a few build limits on the trench, though maybe just tweak one or two, or perhaps ease up slightly where it's dug build simple molFCN setups, like inverters, to better handle faults or interference using alternatives such as workarounds or shielding. They mimic the groove between two wire lines, then measure the clock field (E_{ck}) right down the middle of every molecule row, keeping molecular height steady. With help from four timing stages (based on voltages (V_{el}) listed in Table 2.2) plus SCERPA, it has been tracked how data moves forward. What pops up are hiccups in flow: the pulse strays from the clean path earlier noticed in one-row wires.

Region	Electrode Voltage V_{el}			
	P1	P2	P3	P4
R1	−6 V	−6 V	+8 V	+8 V
R2	+8 V	−6 V	−6 V	+8 V
R3	+8 V	+8 V	−6 V	−6 V

Table 2.2: Voltages applied to regions R1–R3, referred to the structure shown in Fig. 2.1.

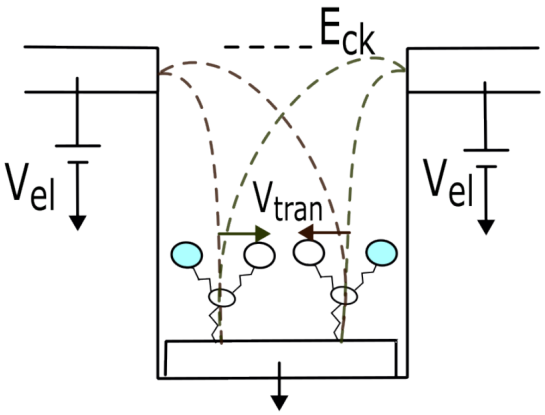


Figure 2.4: Schematic view of how V_{tran} is generated, leading to charge confinement on the external dots.

The issue comes down to shape. Since the molecules aren't perfectly centered in the groove, the electric field tilts instead of staying straight up and down. Side-to-side parts pop up, building a sideways push (V_{tran}). That side force shoves charges toward the wire's outer spots, sneaking an unintended voltage (V_{in}) into areas meant to stay off. Because of this, the timing signal fails to clear the molecular rows completely, so data moves unpredictably when two lines run together using this layout. Shifting from one wire [34] to testing a tiny clock-driven setup is important to checking if signals stay steady when linked sections pass info onward. Instead of stacking parts directly, they are connected through timed triggers across eight zones (R1, R2, up to R8), each running on separate timing pulses, pushing chemical messages step-by-step down a narrow groove. Every zone gets its own voltage settings based on timing, while simulations track how the system behaves with both high ("1") and low ("0") starting inputs [11] The test runs reveal, using this groove shape and power pattern, the data moves smoothly from spot to spot in sync with the timing signal, keeping its polarity intact while avoiding fuzzy middle states. As it shifts forward step by step, no glitches pop up where they shouldn't. Each segment passes the active zone on time, thanks to how the pulse signal lines up with outside nudges. There's no messy overlap messing things up in inactive areas. Basically, the timing setup stays in charge of each molecule's alignment across the whole layout. This aspect checks the design rules for a single wire by testing them in a small circuit. So it shows that the tuned trench shape and voltage work well not just in one wire, but when several clock zones connect. That means with the right size and power settings, molFCN setups can move data reliably using clocks across circuits, not only in basic cases. The results back up the idea that real circuit performance is possible.

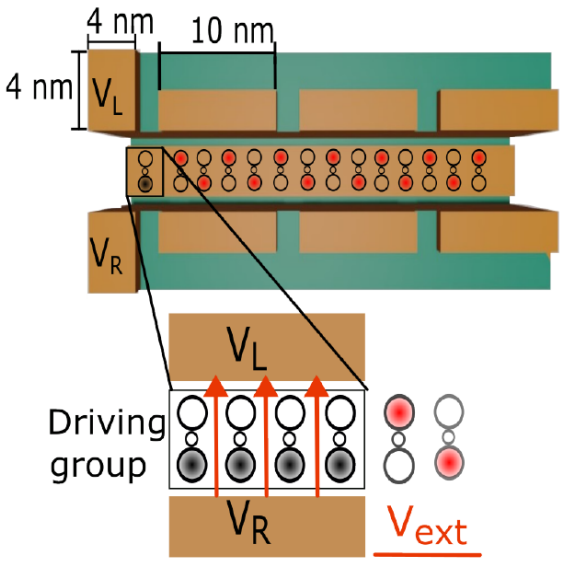


Figure 2.5: Nano-trench structure with embedded driving electrodes and schematic of the driving mechanism [33].

2.4 Single line three-phase wire and MV

This section underline two core parts of MolFCN: a one-line three-phase wire (SL-3P) but also a one-line majority voter (MV) switch. The SL-3P sits inside an Al_2O_3 nano-trench, just big enough for a single chain of molecules yet still keeps voltage needs down. Instead of side contacts, gold electrodes go on top and bottom, spaced by only 2 nm to help sharp clock shifts across zones as shown in figure 2.6. Clock signals run at $\pm 3V$, delivering around $\pm 1V/nm$ through the molecule, exactly what's needed so bis-ferrocene units can reliably switch. The resulting clock signal E_{CLK} gets pulled out then sent into SCERPA runs. Those reveal logic '0' alongside '1' move properly across the SL-3P, just a minor drop at edge zones, yet no mistakes pop up, showing this setup works fine as a basic wire for MolFCN.

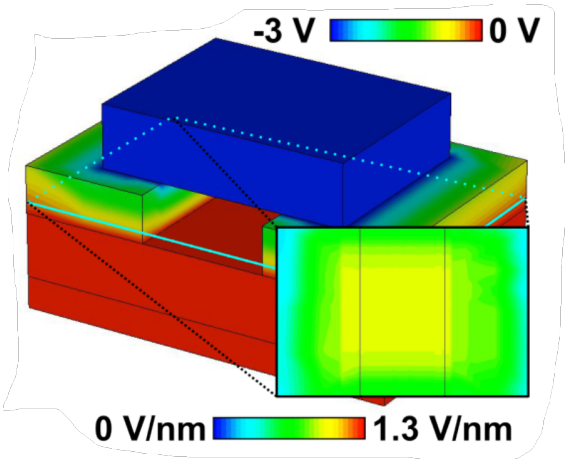


Figure 2.6: A solid upper electrode that spans the whole nano-groove helps prevent voltage dips, this setup spreads the electric force evenly, which shows up plainly in the matching light-output image.

To check if the method works broadly, researchers tried it on a one-wire MV gate. Instead of multiple paths, each input uses a single two-phase line, these bring three logic signals to a core processing unit managed by four control points; meanwhile, an extra two-phase section at the end helps with timing flow. They looked at how the clock signal behaves along the middle wire across three stages: feeding data in, computing the majority result inside the center zone, then pushing the outcome out. Here, holding the neutral state needs more voltage (+6 V), just so the central MV element stays steady without drifting into messy molecule shapes. Tests using one example setup (IN1 = 0, IN2 = 1, IN3 = 0) confirm the pulses move properly through the arms, get handled in the hub, and appear right at the exit. A ripple shows up on the narrow input path - yet everything still works fine. All in all, the test outcomes suggest this tiny groove plus timing method handles basic links as well as working logic switches without issues.

Power Consuption

The intrinsically low power consumption of molecular field-coupled nanocomputing stems from the complete absence of charge-current flow during information transfer [6, 7]. However, the power dissipated by the clocking mechanism—widely regarded as the dominant contribution—has received only limited attention to date [9].

To obtain a first-order estimate, it has been adopted the capacitive model depicted in Fig. 1.13B and 2.4 together with Eq.2.1, considering a single clock zone whose top electrode covers an area of $10nm^2$.

$$P_{\text{diss}} = C_{\text{reg}} f V_{\text{switch}}^2 \tag{2.1}$$

The region capacitance C_{reg} is calculated as the parallel combination of the sidewall capacitances of each nanotrench, with values extracted from Sentaurus Device simulations. Three dielectric materials are evaluated:

- $SiO_2(\epsilon_r = 3.9)$, which yields the lowest capacitance, and the high-k dielectrics
- $Al_2O_3(\epsilon_r = 9)$ and $HfO_2(\epsilon_r = 25)$, both well-established for atomic-layer deposition (ALD) in advanced semiconductor processes [22, 23, 25]

Since the effective relative permittivity and dielectric strength depend strongly on the deposition conditions and underlying substrate, the figures employed here serve solely to reveal

qualitative trends in clock-related power dissipation for the proposed MolFCN device. The initial look at power use in the MolFCN timing setup shows a full majority-voter unit likely uses about six times more energy than one clock area which gives a rough but handy tech reference point. The regional capacitance C_{reg} sits around the attofarad level, it's shaped mostly by $\epsilon_r = 3.9$ and trench height h, though w plays just a small role. Sure enough, boosting ϵ_r ramps up capacit, and that means more power loss; meanwhile, deeper trenches cut down both capacity and energy use, but you get a softer clock signal E_{CLK} when voltage stays fixed. With the system running steady at 3 GHz and trenches set to 3 nm wide, energy use in a single clock area climbs from just a few nanowatts up into microwatts once voltage hits 6 V instead of 1 V. Using high-k insulators boosts this loss quite a bit; however, raising the trench height helps cancel it out, but if h gets too big, molecule behavior might suffer [24, 36]. When you hold the shape and voltage steady, power rises steadily with frequency, right up to 10 GHz. With SiO_2, heat stays low, under $200nW$ no matter how fast it runs; that's around $1.2\mu W$ total for a full voter setup. On the flip side, HfO_2 in narrow $3nm$ gaps blows past $200nW$ just at 1 GHz, though bumping its height to $9nm$ brings results close to what Al_2O_3 gives at only $3nm$. Dropping the switch voltage down to 1 V, every material, even across different shapes, keeps energy loss safely beneath $200nW$. On the whole, findings show a fine balance between insulator choice, groove size, signal strength, and speed, though it's not hard to handle. Because voltage needs depend on which molecule you pick, designers should focus on molecules needing less electric push so they can truly benefit from the tech's natural energy savings. This studies sets out clear steps for improving materials and shapes while building a base for later models that include shifting capacitances, wire effects, and complex circuits, to allow side-by-side tests against standard and new computing methods. Plus, the approach easily applies to other nano-scale field-driven systems, not just this trench setup.

2.5 Current Landscape and Open Challenges in Molecular Field-Coupled Nanocomputing

This studies [33] looks into simulating molecular field-coupled computing at the nano level, focusing especially on building tiny setups that can properly hold redox-active molecules, while also checking, how much energy the clock system uses. Instead of just linking parts, it tests how well they work together under real conditions; each piece must respond correctly when triggered. Structure shapes matter because they guide molecule behavior, so design tweaks happen step by step. Testing power isn't an afterthought, it's built into early trials using basic measurements. Rather than assuming efficiency, actual draws are watched during operation, helping spot waste before scaling up. A fresh nanotrench layout called DevB 2.7 is introduced, tuned to fit bis-ferrocene units while enabling reliable three-phase signal control.

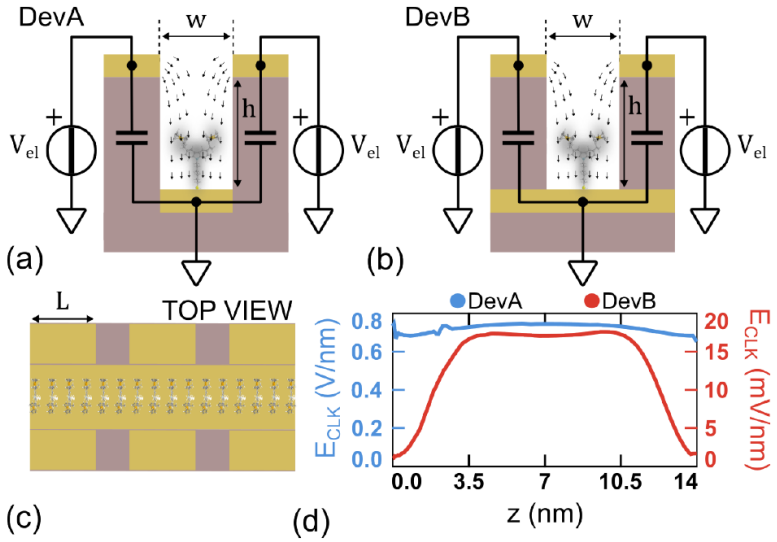


Figure 2.7: (a) The nanotrench gadget mentioned in [11], also checked out in [33]; (b) Tiny trench using a gadget this study suggests. Under the grooves sits the gold coating helps build the device while boosting the ECLK's upward part; (c) Looking down on the DevB three-phase wire, where molecules are fixed at the middle part of the lower contact. (d) Side electric bits (ECLK,y) about $1.33nm$ apart from a trench that's 3 nm wide, this fits both DevA, also DevB middle where V_{el} equals $3V$.

With Sentaurus Device paired with SCERPA, consistent data movement shows up in straight three-phase lines as well as majority-voter (MV) circuits, thanks to manageable clock fields near $\pm 1V/nm$. Compared to the older $DevA$ version, this updated $DevB$ (fig. 2.7(b)) setup cuts down stray sideways electric effects, labeled $E_{CLK,y}$ (fig. 2.7 (d)), which slashes cross signals and noise. In comparison, setups with multiple lines show a clear dip in clocking field intensity, while the middle rows develop stronger sideways elements, suggesting better designs are needed if we're packing things tighter. A basic two-plate setup mimicking the $DevB$ clock area helps guess energy use. Findings show key links between groove shape, insulating stuff, signal strength, also how fast it runs, pointing to needing materials that react well even with weak signals so the system keeps its natural low-energy edge. These results push MolFCN much nearer to real-world use, back up major design decisions with data, while also setting a clear path forward for fine-tuning and later building tiny molecular circuits in lab tests.

Chapter 3

Design of Standard Cell and Derivates Structures for Molecular Field-Coupled Computing

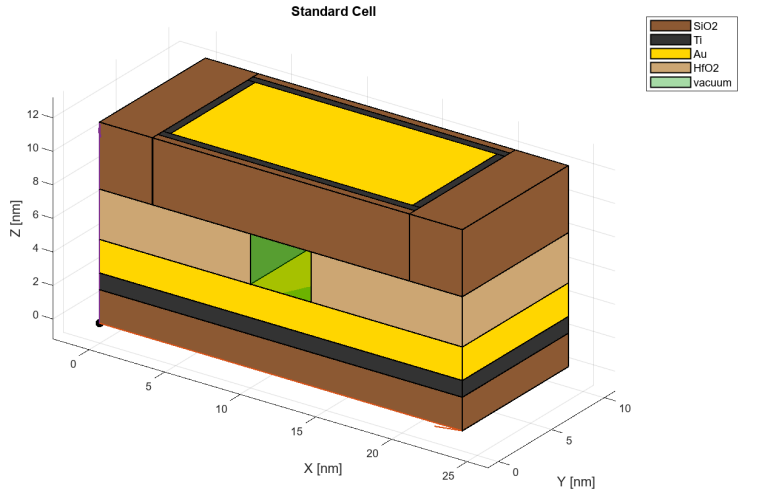
This work proposes a nanometer-scale structure across, with a space perfectly shaped to hold a specific molecule. The idea is to control how the molecule interacts with electric fields directly, rather than relying on electricity constantly flowing through it. Basically, by carefully shaping the electrical environment around the molecule, its internal charges become signals. This approach aligns with efforts to move beyond traditional computer chips, exploring ways to process information through interactions instead of just moving electrons.
Instead of relying on chance, the cell's layered construction allows us to carefully control its electrical conditions thus influencing how signals move within. It uses silicon dioxide, titanium, gold, and hafnium oxide as key components. Silicon oxide provides structure and electrical isolation; titanium binds components together and defines features; gold ensures reliable connections and resists corrosion; while hafnium dioxide offers excellent insulation or increased capacitance when needed. This layered design works with standard nanomanufacturing techniques applying materials, patterning with light, etching unwanted parts, and removing temporary layers making adjustments easier than complete redesigns.
Start with a solid base, a well-defined standard cell. Once its shape and function are fixed, building more complex structures becomes straightforward; begin by connecting elements with buses, then add logical units like majority voters, T wire and L wire. This method manages complexity effectively. Additionally, it enables clear testing of cell stability, signal transmission between connections, and gate function and resistance to interference at the gate level.
These structures has been designed to using specialized software tools SDE, SDEVICE, and svisual-arranging everything in a streamlined process. Key dimensions, distances, and layer thicknesses has been modified to observe reactions, always maintaining critical limits. This balance of flexibility and defined boundaries was intentional; it accelerates moving from an idea to a practical, functional design.
This work provides a basic, adaptable component suitable for building more complex molecular computers. The upcoming sections detail its shape and chosen materials, and explain how simulations were conducted.

3.1 Standard Cell Implementation

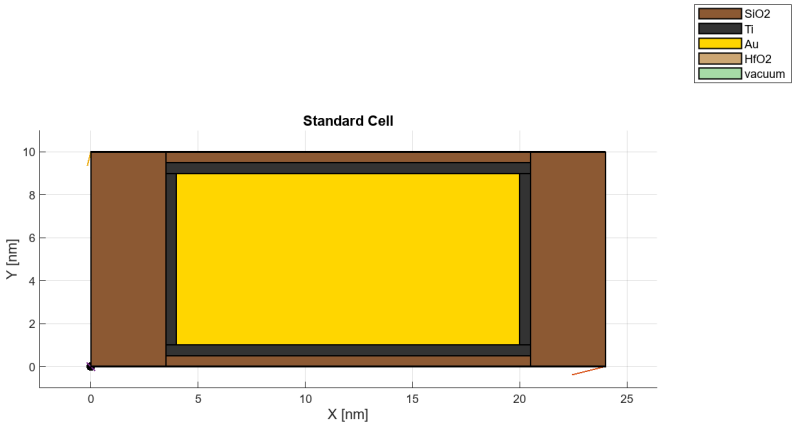
In the subsequent sections of this project, geometric design of the standard cell is implemented, through code lines for Sentaurus Structure Editor, and, thereafter, how the physical properties were assigned to the created structure using code lines for Sentaurus SDevice.

3.1.1 Standard Cell Geometrical Design in Sentaurus Structure Editor

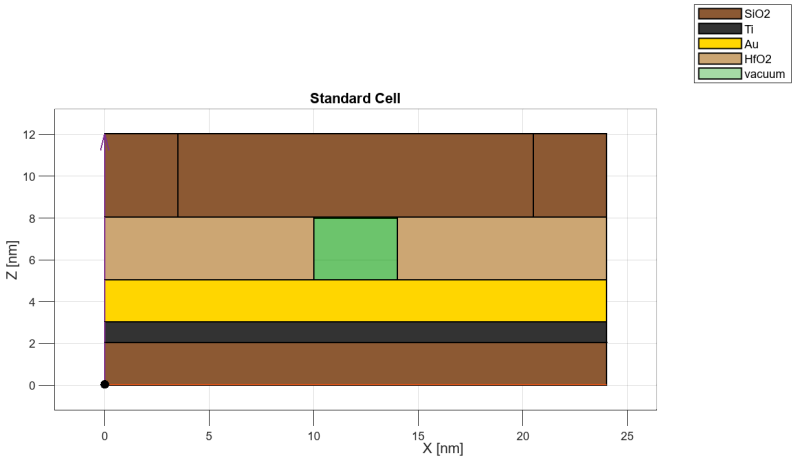
The standard cell started out as a tough, ready-to-use piece meant to act like the go-to basic part. It measures $24nm$ by $10nm$ across, standing $12nm$ tall altogether. From bottom to top, it stacks up five main levels: first comes a $2nm$ SiO_2 base, followed by a $1nm$ Ti film that helps things stick, then a $2nm$ Au layer for conducting current, topped off with a $3nm$ empty gap made for holding specific molecules. That open zone runs through the full depth, stretches $4nm$ wide, and is built on purpose with twin paths side by side, each one guiding its own line of molecules. To keep the empty space locked exactly where it should be, hafnium oxide (HfO_2) works as a built-in spacer, spreading sideways across the whole component edge when needed - holding the gap steady while giving it shape. Up above, right in the middle, there's a tiny gold lid measuring $16 \cdot 8nm$ on the surface and standing $4nm$ tall; this part sits inside a slim layer of titanium, then gets boxed in by SiO_2 at the same height so everything stays flat.
The design's size choices make it easy to build, keep steady, and fit the bis-ferrocene part well. With a $3nm$ gap between layers and $4nm$ wide dual tracks, there's enough room for two lines of molecules packed neatly - without weakening electric control or causing surface interference. Instead of just bonding poorly, titanium layers help stick things together and block mixing at edges; meanwhile gold stays unreactive while conducting signals efficiently. Rather than using regular insulators, hafnium oxide acts as a dense spacer that shapes empty zones sharply yet keeps electrical separation intact. Alongside this, silica around the frame smooths everything into an even, flat finish. Altogether, the setup aims for a reliable building block - one that fits easily into larger designs or modeling pipelines. In 3.1 it is shown the visual idea project trough a 3D render code implemented in MatLab.



(a) StdCell on XYZ axis.



(b) StdCell on XY plane.



(c) StdCell on XZ plane.

Figure 3.1: 3D project design overview of the Standard Cell

I put together the device layout following the given blueprint using automated scripts along

with the Sentaurus Structure Editor's visual interface. Scripts make each build consistent and adjustable through inputs. Meanwhile, the graphical tool helps speed up changes and spot shape problems sooner. Rules remain strict: precise layer thicknesses, accurate horizontal spans, zero overlapping parts.
In the SDE setup, each layer has been defined and the empty gap right from the start using labeled settings for width, depth, and position.

Layer Nº	**Material**	**Heigh [nm]**
6 & 7	$Au + Ti + SiO_2$	4
5	$Vacuum$	3
4	HfO_2	3 (7)
3	Au	2
2	Ti	1
1	SiO_2"	2

Table 3.1: Standard cell geometrical parameters.

Description of the line's code follows according to [15]. The block 3.1 locks down core size and stack settings for an SDE cell - measurements in microns, so nanoscale values adjusted. Main part lays out the flat layout (x: BottomDielectricWidth, y: CellLength), layer depths like SiO_2, Ti, Au, along with the open gap space (width TrenchMiddleWidth, height TrenchWallHeight), while HfO_2 side width comes from TrenchWallWidth. Stub entries ViaThickness and PhasesNumber stay free for later use when vias or multi-stage setups get added. The top part defines the maximum gold cap dimensions (TopGoldWidthX, TopGoldHeightY, TopGoldThicknessZ), along with how thick the titanium edge is (TitaniumRimThickness). These settings work together to ensure consistent fabrication, help test responsiveness across ranges, adjust according to mesh density yet still keep layers from overlapping and maintain correct positioning.

Listing 3.1: SDE: main and top-cap parameters [15]

```
;; ------------------------------
;; MAIN PARAMETER
;; ------------------------------
(sde:define-parameter "BottomDielectricWidth" 0.024000)
(sde:define-parameter "TrenchMiddleWidth" 0.004000)
(sde:define-parameter "TrenchWallWidth" (/ (- BottomDielectricWidth TrenchMiddleWidth) 2))
(sde:define-parameter "BottomDielectricThickness" 0.002000)
(sde:define-parameter "CellLength" 0.010000)
(sde:define-parameter "TrenchWallHeight" 0.003000)
(sde:define-parameter "AdhesionLayerThickness" 0.001000)
(sde:define-parameter "GoldLayerThickness" 0.002000)
(sde:define-parameter "ViaThickness" 0.001000)
(sde:define-parameter "PhasesNumber" 1)

;; ------------------------------
;; TOP GOLD + CORNER
;; ------------------------------
(sde:define-parameter "TopGoldWidthX" 0.016000)
(sde:define-parameter "TopGoldHeightY" 0.008000)
(sde:define-parameter "TopGoldThicknessZ" 0.004000)
(sde:define-parameter "TitaniumRimThickness" 0.000500)
```

The part 3.2 calculates positions and layer heights using main settings so pieces fit close without overlapping. From left to right, x_2ndWallStart shows where the empty gap ends and

the right HfO2_22 wall begins. In vertical order, endpoint totals build up at each boundary: Ti's peak (z_BottomAdhesionLayerEnd), mid-layer gold's top (z_MiddleGoldEnd), height of HfO2_22 walls or air space (z_WallEnd), plus highest gold cover's tip (z_2ndGoldLayerEnd). Central point on surface (CenterX, CenterY) sets balanced layout. Volume limits for upper gold (x_Au_min/max, y_Au_min/max) along with titanium edge (x_Rim_, y_Rim_) come from stretching gold area by border width, giving flexible, precise setup for outer shapes.

Listing 3.2: SDE: secondary/derived parameters [15]

```
;; ------------------------------
;; SECONDARY PARAMETER
;; ------------------------------
(sde:define-parameter "x_2ndWallStart" (+ TrenchWallWidth TrenchMiddleWidth))
(sde:define-parameter "z_BottomAdhesionLayerEnd" (+ BottomDielectricThickness
    AdhesionLayerThickness))
(sde:define-parameter "z_MiddleGoldEnd" (+ BottomDielectricThickness AdhesionLayerThickness
    GoldLayerThickness))
(sde:define-parameter "z_WallEnd" (+ z_MiddleGoldEnd TrenchWallHeight))
(sde:define-parameter "z_2ndGoldLayerEnd" (+ z_WallEnd TopGoldThicknessZ))
(sde:define-parameter "CenterX" (/ BottomDielectricWidth 2.0))
(sde:define-parameter "CenterY" (/ CellLength 2.0))

;; Bounding boxes
(sde:define-parameter "x_Au_min" (- CenterX (/ TopGoldWidthX 2.0)))
(sde:define-parameter "x_Au_max" (+ CenterX (/ TopGoldWidthX 2.0)))
(sde:define-parameter "y_Au_min" (- CenterY (/ TopGoldHeightY 2.0)))
(sde:define-parameter "y_Au_max" (+ CenterY (/ TopGoldHeightY 2.0)))
(sde:define-parameter "x_Rim_min" (- x_Au_min TitaniumRimThickness))
(sde:define-parameter "x_Rim_max" (+ x_Au_max TitaniumRimThickness))
(sde:define-parameter "y_Rim_min" (- y_Au_min TitaniumRimThickness))
(sde:define-parameter "y_Rim_max" (+ y_Au_max TitaniumRimThickness))
```

This chunk 3.2 sets up the base layers by stacking three flat pieces right on top of each other, stretching across the whole area sideways (0≤ x≤ BottomDielectricWidth, 0≤ y≤ CellLength). To start off, a SiO_2 bed runs from z=0 upward to BottomDielectricThickness. Right above that comes a thin coat of titanium it goes from where the glass part ends up to z_BottomAdhesionLayerEnd. On top of the metal glue sits a middle slab of gold; this one's tagged BOTTOMCONTACT so it can hook up contacts later and stretches from z_BottomAdhesionLayerEnd to z_MiddleGoldEnd. Stacking using total height markers keeps everything snug, no spaces or double bits, while feeding precise positions into future tweaks and grid splits.

Listing 3.3: SDE: bottom structure (SiO_2, Ti, mid-Au) [15]

```
;; ============================================================
;; BOTTOM STRUCTURE
;; ============================================================
(sdegeo:create-cuboid (position 0 0 0)
 (position BottomDielectricWidth CellLength BottomDielectricThickness)
 "SiO2")

(sdegeo:create-cuboid (position 0 0 BottomDielectricThickness)
 (position BottomDielectricWidth CellLength z_BottomAdhesionLayerEnd)
 "Titanium")

(define BOTTOMCONTACT
 (sdegeo:create-cuboid (position 0 0 z_BottomAdhesionLayerEnd)
 (position BottomDielectricWidth CellLength z_MiddleGoldEnd)
 "Gold"))
```

This part 3.4 makes a 4 nm gap in vacuum from z = z_MiddleGoldEnd up to z = z_WallEnd, while stuffing the sides with HfO_2 on left and right. Thanks to TrenchWallWidth along with

x_2ndWallStart, things line up side-by-side just right across the whole y-length tight fit, no extra space or clashes.

Listing 3.4: SDE: central structure (vacuum + HfO2 walls) [15]

```
;; ============================================================
;; CENTRAL STRUCTURE
;; ============================================================
(sdegeo:create-cuboid (position TrenchWallWidth 0 z_MiddleGoldEnd)
 (position x_2ndWallStart CellLength z_WallEnd)
 "Vacuum")

(define BODY_LEFT
 (sdegeo:create-cuboid (position 0 0 z_MiddleGoldEnd)
 (position TrenchWallWidth CellLength z_WallEnd)
 "HfO2"))

(define BODY_RIGHT
 (sdegeo:create-cuboid (position x_2ndWallStart 0 z_MiddleGoldEnd)
 (position BottomDielectricWidth CellLength z_WallEnd)
 "HfO2"))
```

The block 3.5 fits the upper gold lid into the set area $[x_A u_m in, x_A u_m ax] \times [y_A u_m in, y_A u_m ax]$, stretching vertically from z_WallEnd to z_2ndGoldLayerEnd. Surrounding that lid, four shapes made of titanium form a border; instead of staying separate, they're joined into one solid piece through sdegeo:bool-unite. Since every boundary is driven by inputs, alignment stays centered, touching edges line up perfectly, while any clash with nearby SiO_2 inserts gets prevented.

Listing 3.5: SDE: top structure (top Au cap + Ti rim) [15]

```
;; ============================================================
;; TOP STRUCTURE
;; ============================================================
(define TOPCONTACT
 (sdegeo:create-cuboid (position x_Au_min y_Au_min z_WallEnd)
 (position x_Au_max y_Au_max z_2ndGoldLayerEnd)
 "Gold"))

;; Ti Corners (4 sides)
(define RIM_LEFT (sdegeo:create-cuboid (position x_Rim_min y_Rim_min z_WallEnd)
 (position x_Au_min y_Rim_max z_2ndGoldLayerEnd)
 "Titanium"))
(define RIM_RIGHT (sdegeo:create-cuboid (position x_Au_max y_Rim_min z_WallEnd)
 (position x_Rim_max y_Rim_max z_2ndGoldLayerEnd)
 "Titanium"))
(define RIM_BOTTOM (sdegeo:create-cuboid (position x_Rim_min y_Rim_min z_WallEnd)
 (position x_Rim_max y_Au_min z_2ndGoldLayerEnd)
 "Titanium"))
(define RIM_TOP (sdegeo:create-cuboid (position x_Rim_min y_Au_max z_WallEnd)
 (position x_Rim_max y_Rim_max z_2ndGoldLayerEnd)
 "Titanium"))

(sdegeo:bool-unite (list RIM_LEFT RIM_RIGHT RIM_BOTTOM RIM_TOP))
```

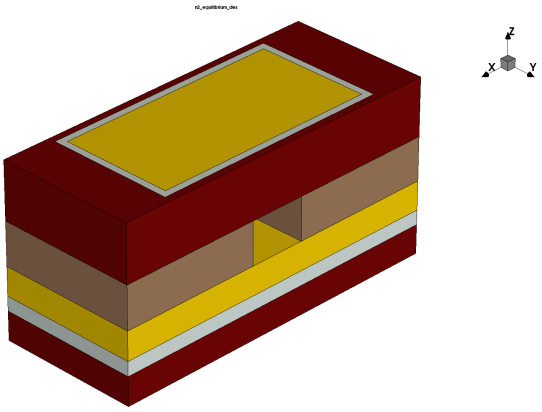
This part 3.6 creates two groups of contacts lower and upper gold and hooks them up separately. After that, it draws a precision zone lined up with the trench to sharpen the grid focus. Base grid dimensions get set first, followed by slotting in the fine-tuned area right where needed. Rules tied to surfaces are applied strictly at Gold meeting Vacuum and Gold touching Titanium. Finally, the grid gets assembled, setting up the layout so later device simulations can move forward

Listing 3.6: SDE: contacts, mesh refinement, and build [15]

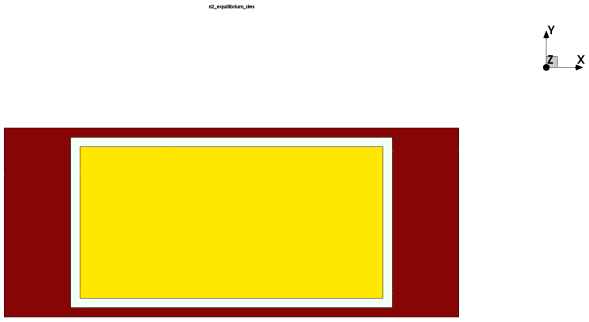
```
;; ============================================================
;; CONTACTS
;; ============================================================

(sdegeo:define-contact-set "BottomContact" 4 (color:rgb 1 0 0) "##")
(sdegeo:set-current-contact-set "BottomContact")
(sdegeo:set-contact BOTTOMCONTACT "BottomContact")

(sdegeo:define-contact-set "TopContact" 4 (color:rgb 0 0 1) "##")
(sdegeo:set-current-contact-set "TopContact")
(sdegeo:set-contact TOPCONTACT "TopContact");;

;; ============================================================
;; MESH / REFINEMENT
;; ============================================================
(sdedr:define-refeval-window "InnerTrenchMesh" "Cuboid"
 (position TrenchWallWidth 0 z_BottomAdhesionLayerEnd)
 (position x_2ndWallStart CellLength z_2ndGoldLayerEnd))

(sdedr:define-refinement-size "TrenchRefinement" 0.001 0.005 0.001 0.001 0.005 0.001)
(sdedr:define-refinement-placement "RefinementPlacement_1" "TrenchRefinement"
 (list "window" "InnerTrenchMesh"))
(sdedr:define-refinement-function "TrenchRefinement" "MaxLenInt" "Gold" "Vacuum" 0.0001 "DoubleSide
     ")
(sdedr:define-refinement-function "TrenchRefinement" "MaxLenInt" "Gold" "Titanium" 0.0001)

;; ============================================================
;; BUILD + SAVE STRUCTURE
;; ============================================================
(sde:build-mesh "n@node@")
```

The entire code lines are shown in the appendix section 1

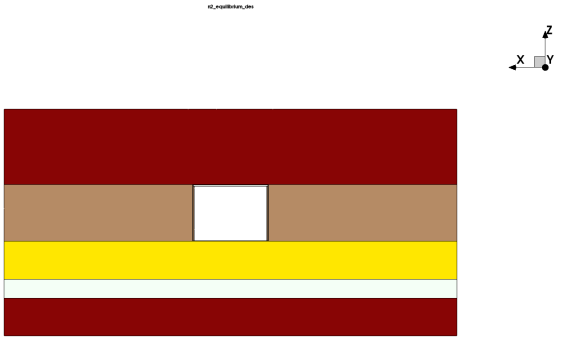
The resulting structure in fig.3.2 from Senturus Structure Editor SDE rendering.



(a) SDE: StdCell on XYZ axis.



(b) SDE: StdCell on XY plane.



(c) SDE: StdCell on XZ plane.

Figure 3.2: SDE: 3D project design overview of the Standard Cell

3.1.2 Standard Cell Physics implementation in Sentaurus SDevice and visualization in Svisual

This SDevice 3.7 setup connects the earlier shape and grid to a steady heat-electric insulation analysis, while sorting out what results to grab plus how to solve it.Each code line is referred to it's relative manual [16]. The File section hooks up the grid data (n1_msh.tdr), picks where plots and current logs go, then pulls in a settings file for live tweaks.

Listing 3.7: SDevice: Files involved [16]

```
File {
 Grid = "n1_msh.tdr"
 Plot = "n@node@_device_des.tdr"
 Current = "n@node@_device_des.plt"
 Parameter = "sdevice.par"
}
```

Contacts 3.8 pop into place using labels like BottomContact, TopContact both starting at zero volts to line up exactly with SDE's named touchpoints, making sure limits tie clearly to physical surfaces.

Listing 3.8: SDevice: Contacts [16]

```

Electrode {
 { Name = "BottomContact" Voltage = 0 }
 { Name = "TopContact" Voltage = 0 }
}
```

Material behavior 3.9 depends on area tags: HfO_2 alongside SiO_2 act like leaky insulators through CondInsulator, allowing displacement flow plus slight current seepage minus any semiconductor-style charge movement. Heat settings connect at both ends by Thermode markers set to 300 K including tiny interface resistances, wrapping up the combined electrical-thermal setup.

Listing 3.9: SDevice: Materials [16]

```
# ----------------------------
# PHYSICS
# ----------------------------
# MODELS FOR MATERIALS
Physics (Material="HfO2") {
 CondInsulator
}

Physics (Material="SiO2") {
 CondInsulator
}
```

The Plot 3.10 block asks for electrostatic and thermal fields, dielectric values, space charge, band structure data, plus conduction and displacement current densities, alongside the full current vector.

Listing 3.10: SDevice: Thermal aspect and plot commands [16]

```
# THERMIC CONTACT
Thermode {
 { Name = "BottomContact" Temperature = 300 SurfaceResistance = 1e-5 }
 { Name = "TopContact" Temperature = 300 SurfaceResistance = 1e-5 }
}
# ----------------------------
# PLOT
# ----------------------------
Plot {
 Potential
 ElectricField
 DielectricConstant
 Temperature

 # DIELECTRIC CURRENTS
 ConductionCurrentDensity/Vector
 DisplacementCurrentDensity/Vector

 # TOTAL CURRENT
 TotalCurrent/Vector

 # OTHERS
 SpaceCharge
 Potential Doping
 BandGap ElectronAffinity
 ConductionBandEnergy ValenceBandEnergy
}
```

Math options 3.11 turn on relative error handling together with extrapolation tricks. The solving steps start with a basic Poisson preview solution, shift into a fully linked balance state covering Poisson, electrons, holes, heat, and CondInsulator parts, wrap up with a near steady voltage climb pushing TopContact to 10 volts using smart time jumps, tossing out interim visuals during the voltage run.

Listing 3.11: SDevice: Mathematic solver [16]

```
Math {
 RelErrControl
 Extrapolate
}

# ----------------------------
# SOLVE
# ----------------------------
Solve
{
Coupled (Iterations= 100 LineSearchDamping= 1e-8) {Poisson}
Coupled{ Poisson Electron Hole Temperature Contact CondInsulator }
Plot(FilePrefix="n@node@_equilibrium")
quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
Goal {name= "TopContact" voltage = 10}
plot { range=(0, 1) intervals= 1}

){coupled { Poisson Electron Hole Temperature CondInsulator }}}
```

The entire code lines are shown in the appendix section 3

3.1.3 Parametric Standard-Cell Modeling: MATLAB GUI Design and Sentaurus Integration

This section documents a customizable tool in MATLAB designed to create tiny building blocks for nanoscale devices. It comes with simple controls for arranging these pieces into patterns either grids or plus shapes useful for decision-making circuits. This design focuses on creating reliable, adjustable shapes, able to modifying all parameters values. Consequently, designs transfer smoothly to Sentaurus software for detailed electrical testing.
Parametrization isn't just about neat shapes; it clarifies what a design should do its core ideas and proportions separate from specific dimensions. It also lets designers quickly test different options by tweaking only a few values, like choosing between versions or altering cell spacing. Moreover, every setting in the initial design directly corresponds to its final form via clear connections and conversions. User interfaces simplify things even more by keeping edits organized, instantly updating visuals, then creating reliable images for records.
This creates a reliable process: choose settings, build shapes using MATLAB, then generate meshes and assign materials within the simulation software. Matlab Codes were built firstly from the file *StandardCell.m*, and then implemented with a Graphical User Interface (GUI) in the relative Matlab file *stdcell_gui.m*, shown in fig 3.3

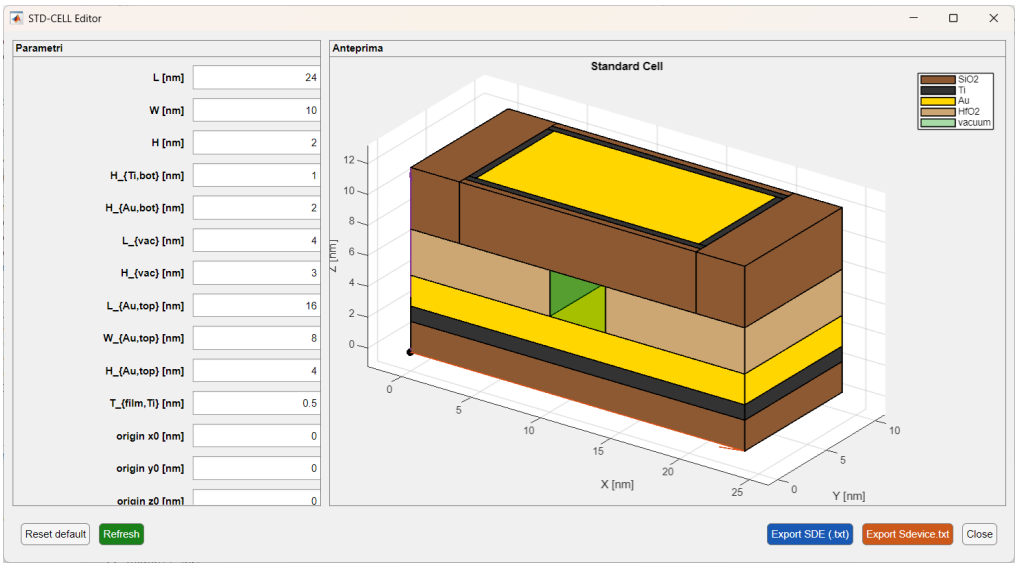


Figure 3.3: Matlab GUI for Standard Cell

All the value are customizable by the user from the Matlab GUI. Anyway, in the table 3.2 are shown all the relation between parameters, implemented inside the Matlab code in order to mantain always the designed idea of the standard cell.

Parameter	Symbol	Relation	Meaning of the relation
Overall Length	L	Arbitrary	Only reasonable values are those with $L \in \mathbb{R}_{\geq 0}$.
Overall Width	W	Arbitrary	Only reasonable values are those with $W \in \mathbb{R}_{\geq 0}$.
Overall Height	H	Arbitrary	Only reasonable values are those with $H \in \mathbb{R}_{\geq 0}$.
Height of bottom Titanium	$H_{\mathrm{Ti,bot}}$	Arbitrary	Only reasonable values are those with $H_{\mathrm{Ti,bot}} \in \mathbb{R}_{\geq 0}$.
Height of bottom Gold	$H_{\mathrm{Au,bot}}$	Arbitrary	Only reasonable values are those with $H_{\mathrm{Au,bot}} \in \mathbb{R}_{\geq 0}$.
Length of Vacuum	L_{vac}	$L_{\mathrm{vac}}^{\max} = L$	The maximum allowed value for L_{vac} equals the overall length L.
Height of Vacuum	H_{vac}	Arbitrary	Only reasonable values are those with $H_{\mathrm{vac}} \in \mathbb{R}_{\geq 0}$. The HfO_2 height adapts to H_{vac}.
Length of top Gold	$L_{\mathrm{Au,top}}$	$L_{\mathrm{Au,top}}^{\max} = L - 2T_{\mathrm{film}}$	The maximum $L_{\mathrm{Au,top}}$ is the overall length L minus twice the titanium-film thickness.
Width of top Gold	$W_{\mathrm{Au,top}}$	$W_{\mathrm{Au,top}}^{\max} = W - 2T_{\mathrm{film}}$	The maximum $W_{\mathrm{Au,top}}$ is the overall width W minus twice the titanium-film thickness.
Height of Top Gold	$H_{\mathrm{Au,top}}$	Arbitrary	Only reasonable values are those with $H_{\mathrm{Au,top}} \in \mathbb{R}_{\geq 0}$.
Thickness of Titanium film	$T_{\mathrm{film,Ti}}$	$T_{\mathrm{film,Ti}_{max}} = W \cdot L$	Extandable till $W \cdot L$
X origin	X0	Arbitrary	Value to change structure x origin
Y origin	Y0	Arbitrary	Value to change structure y origin
Z origin	Z0	Arbitrary	Value to change structure z origin

Table 3.2: Relation between parameters of the Standard cell

3.1.4 Standard Cell Cut-Y version: Geometrical Design in Sentaurus Structure Editor

Starting from the Standard Cell, a version called Cut-Y came about this one's by insert a cut plane along the XZ surface, lined up with the titanium layer, but just on one side. Because of that change, it breaks symmetry in what used to be an even design. Instead of mirroring everything, it was shaped and tested Cut-Y so building bigger setups would take less effort. With this setup, there's no need to repeat the SiO_2 layer where units meet. That tweak makes simulations quicker and neater.
Getting rid of extra SiO_2 overlap on Y-edges cuts down unnecessary insulator chunks fewer material borders mean less push for tighter meshing. Fewer tight spaces plus matching edges usually mean better mesh bigger smallest parts, less junk shapes as well as improved solve stability, cutting down on loop steps. All the values remains the same as the Standard cell 3.1. Here follows in the fig 3.4 the extra geometrical implementation to the Standard Cell

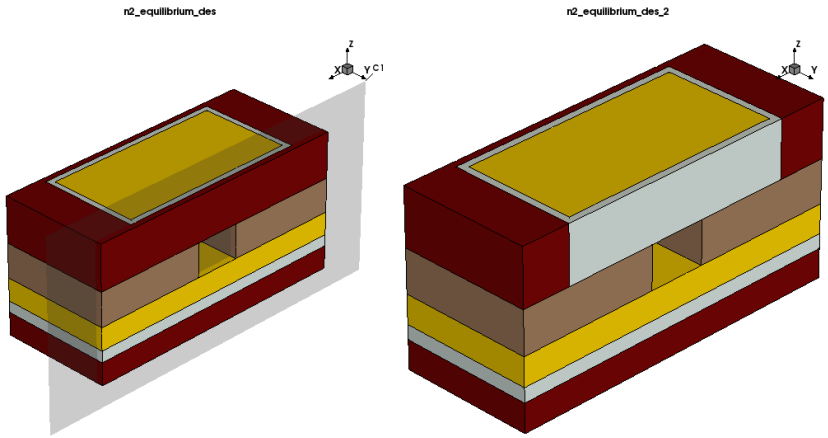


Figure 3.4: Cut-Y Standard cell version implementation trough the cut plane XZ.

Since the geometrical description of the Cut-Y version, here follow just the main variation to SDE code, in relation to the fact that the structure is basically the same of 3.1 with XZ cut implementation.

The first variation in the code is the implementation of YMaxCut = y_Rim_max 3.12and its substitute for CellLength in all bottom and central 3D primitives 3.13. Consequently, the cell is truncated along the +Y direction precisely flush with the upper edge of the titanium rim. In 3.13 the red codes are the one referred to the Standard Cell, the green one referred to Cut-Y version.

Listing 3.12: SDE: XZ plane [15]

```
;; ZX PLANE CUT DEFINITION (Cut-Y)
(sde:define-parameter "YMaxCut" y_Rim_max)
```

Listing 3.13: SDE: Differences in codes line of the Cut-Y version from Standard cell one (CellLength to YMaxCut in BOTTOM and CENTRAL parts)[15]

```
====== BOTTOM STRUCTURE ============================

- (sdegeo:create-cuboid (position 0 0 0)
-  (position BottomDielectricWidth CellLength BottomDielectricThickness) "SiO2")
+ (sdegeo:create-cuboid (position 0 0 0)
+  (position BottomDielectricWidth YMaxCut BottomDielectricThickness) "SiO2")

- (sdegeo:create-cuboid (position 0 0 BottomDielectricThickness)
-  (position BottomDielectricWidth CellLength z_BottomAdhesionLayerEnd) "Titanium")
+ (sdegeo:create-cuboid (position 0 0 BottomDielectricThickness)
+  (position BottomDielectricWidth YMaxCut z_BottomAdhesionLayerEnd) "Titanium")

- (define BOTTOMCONTACT
-  (sdegeo:create-cuboid (position 0 0 z_BottomAdhesionLayerEnd)
-   (position BottomDielectricWidth CellLength z_MiddleGoldEnd) "Gold"))
+ (define BOTTOMCONTACT
+  (sdegeo:create-cuboid (position 0 0 z_BottomAdhesionLayerEnd)
+   (position BottomDielectricWidth YMaxCut z_MiddleGoldEnd) "Gold"))

===== CENTRAL STRUCTURE ============================

- (sdegeo:create-cuboid (position TrenchWallWidth 0 z_MiddleGoldEnd)
```

```
-  (position x_2ndWallStart CellLength z_WallEnd) "Vacuum")
+ (sdegeo:create-cuboid (position TrenchWallWidth 0 z_MiddleGoldEnd)
+  (position x_2ndWallStart YMaxCut z_WallEnd) "Vacuum")

- (define BODY_LEFT
-  (sdegeo:create-cuboid (position 0 0 z_MiddleGoldEnd)
-   (position TrenchWallWidth CellLength z_WallEnd) "HfO2"))
+ (define BODY_LEFT
+  (sdegeo:create-cuboid (position 0 0 z_MiddleGoldEnd)
+   (position TrenchWallWidth YMaxCut z_WallEnd) "HfO2"))

- (define BODY_RIGHT
-  (sdegeo:create-cuboid (position x_2ndWallStart 0 z_MiddleGoldEnd)
-   (position BottomDielectricWidth CellLength z_WallEnd) "HfO2"))
+ (define BODY_RIGHT
+  (sdegeo:create-cuboid (position x_2ndWallStart 0 z_MiddleGoldEnd)
+   (position BottomDielectricWidth YMaxCut z_WallEnd) "HfO2"))
```

Other differences are related to FILL_TOP parameter 3.14; only FILL_BOTTOM, FILL_LEFT, and FILL_RIGHT are retained. In the standard cell, by contrast, the upper SiO_2 cap (FILL_TOP) is present up to CellLength 3.5.

Listing 3.14: SDE: FILL_TOP removed [15]

```
- (define FILL_TOP
-   (sdegeo:create-cuboid (position 0 y_Rim_max z_WallEnd)
-     (position BottomDielectricWidth CellLength z_2ndGoldLayerEnd) "SiO2"))
-
- (sdegeo:bool-unite (list FILL_BOTTOM FILL_TOP FILL_LEFT FILL_RIGHT))
+ (sdegeo:bool-unite (list FILL_BOTTOM FILL_LEFT FILL_RIGHT))
```

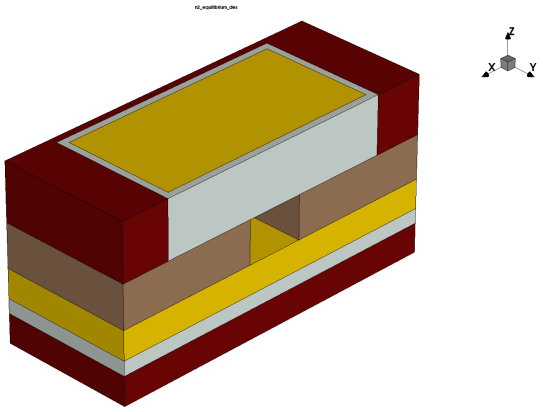
The refinement window (sdedr:define-refeval-window) in Cut-Y references YMaxCut. In the standard version, it references CellLength. The standard cell is symmetric in the Y direction; the Cut-Y configuration is asymmetric, featuring an "open" boundary on the +Y side. Cut-Y is designed to be abutted along Y without duplicating SiO_2 between adjacent cells; the standard cell is self-contained.The remaining differences are listed here below in 3.15.

Listing 3.15: SDE: refeval window [15]

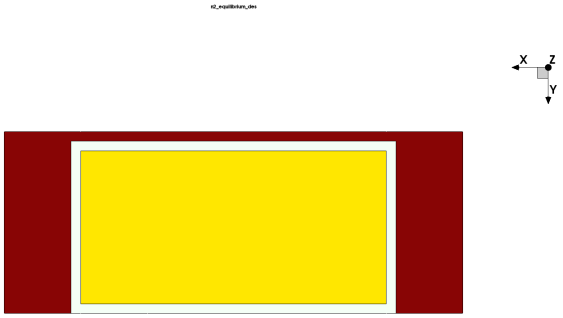
```
- (sdedr:define-refeval-window "InnerTrenchMesh" "Cuboid"
-  (position TrenchWallWidth 0 z_BottomAdhesionLayerEnd)
-  (position x_2ndWallStart CellLength z_2ndGoldLayerEnd))
+ (sdedr:define-refeval-window "InnerTrenchMesh" "Cuboid"
+  (position TrenchWallWidth 0 z_BottomAdhesionLayerEnd)
+  (position x_2ndWallStart YMaxCut z_2ndGoldLayerEnd))
```

The entire code lines are shown in the appendix section 2.

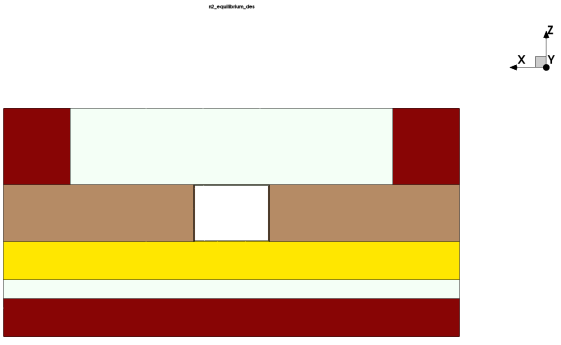
The resulting structure in fig.3.5 from Senturus Structure Editor SDE rendering.



(a) SDE: StdCell Cut-Y on XYZ axis.



(b) SDE: StdCell Cut-Y on XY plane.



(c) SDE: StdCell Cut-Y on XZ plane.

Figure 3.5: SDE: 3D project design overview of the Standard Cell

3.1.5 Standard Cell Cut-Y version: Physics implementation in Sentaurus SDevice and visualization in Svisual

The SDevice code, by contrast, requires no changes. The same physical parameters are applied to a structure that exhibits no substantive differences in contact definition or in its simulation behavior; in relation to this, it can be made a reference directly to the codes used for the standard cell. In particular from code lines 3.7 to 3.11. The entire code lines are shown in the appendix section 3

3.1.6 Standard Cell Cut-Y version: MATLAB GUI Design and Sentaurus Integration

This section documents a customizable tool in MATLAB designed to create tiny building blocks for nanoscale devices, as exactly made for the Standard cell in fig 3.3. It comes with simple controls for arranging these pieces into patterns either grids or plus shapes useful for decision-making circuits. This design focuses on creating reliable, adjustable shapes, able to modifying all parameters values, strictly made up in order to have always a cut plane aligned to titanium corner for respect the idea of the Cut-Y version of the Standard cell. Consequently, designs transfer smoothly to Sentaurus software for detailed electrical testing as done before for Standard Cell.Matlab Codes were built firstly from the file
stdcell_cutY_far_flush_with_Ti.m, and then implemented with a Graphical User Interface (GUI) in the relative Matlab file *stdcell_gui_cutY_far_flush_with_Ti.m*, shown in fig 3.6.

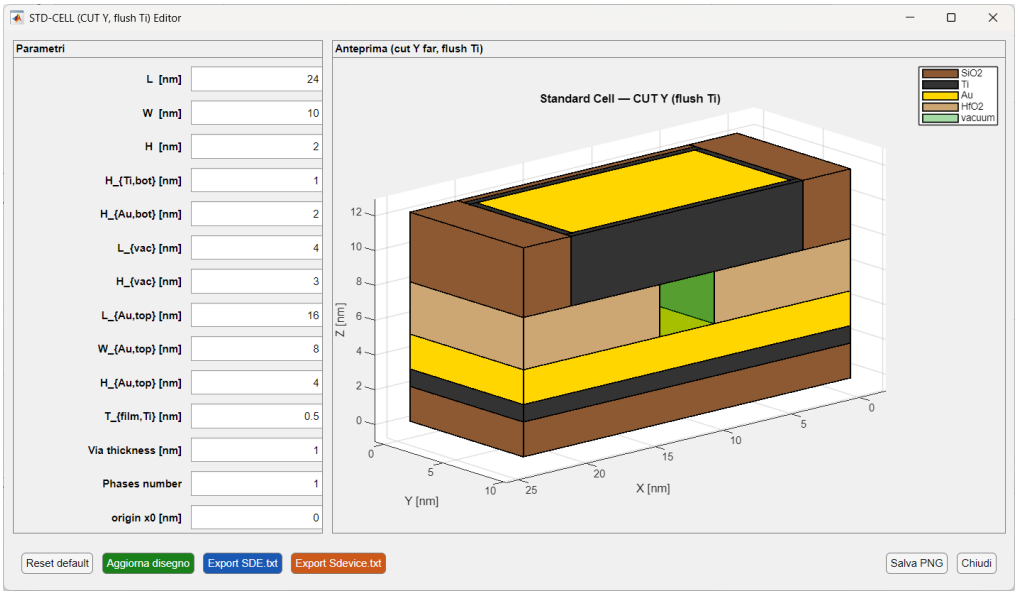
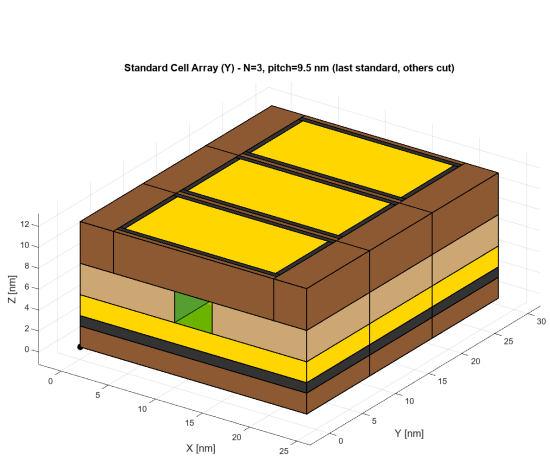


Figure 3.6: Matlab GUI for Standard Cell Cut-Y Version

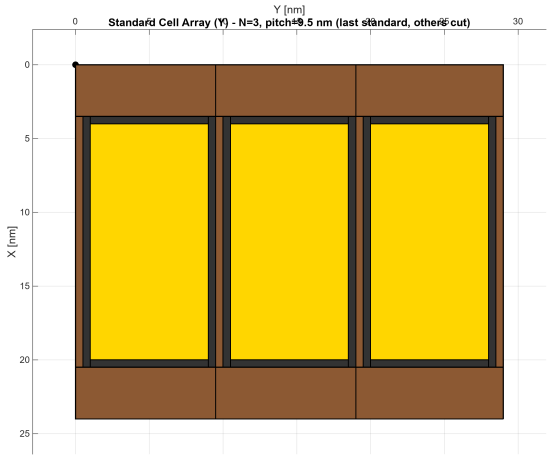
All the value are customizable by the user from the Matlab GUI according to relations explained in the table 3.2.

3.2 Array N-Phase

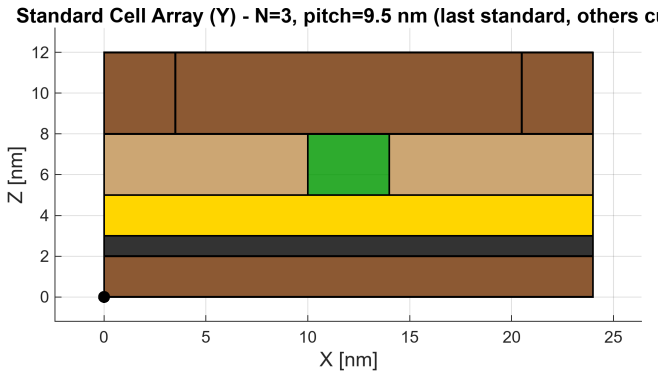
This section presents the first complex structure implementation of this project: the Array N-Phase. The first basic idea is the structure composed of two standard cells of the Cut-Y type and one standard cell aligned in sequence, adjacent along the long edge of the titanium film. These three cells were implemented to form a three-stage array, thereby operationalizing the bus concept. Geometric design of the Array 3-Phase standard cell is implemented, through code lines for Sentaurus Structure Editor, and, thereafter, how the physical properties were assigned to the created structure using code lines for Sentaurus SDevice.



(a) 3D Array 3-Phase on XYZ axes



(b) 3D Array 3-Phase on the XY plane



(c) 3D Array 3-Phase on the XZ plane

Figure 3.7: 3D project design overview of the Array 3-Phase.

3.2.1 Array Standard Cell 3-Phase: Geometrical Design in Sentaurus Structure Editor

Description of the line's code follows according to [15]. The block 3.1 and 3.2 locks down core size and stack settings for an SDE cell measurements in microns, so nanoscale values adjusted. This part of code lines remain the same of the Standard Cell.

The code lines 3.16 defines the Y-pitch between cells and the associated per-cell offsets y0_i and are defined also helper routine for converting local coordinates to cell coordinates.

Listing 3.16: SDE: Array variations structure [15]

```
    (define RimPackWidth (+ TopGoldHeightY (* 2.0 TitaniumRimThickness)))
(define Pitch (if (> CellLength RimPackWidth)
                  (min CellLength (* 0.5 (+ CellLength TopGoldHeightY (* 2.0 TitaniumRimThickness))
    ))
                  CellLength))

(define y0_1 0.0)
(define y0_2 (* 1.0 Pitch))
(define y0_3 (* 2.0 Pitch))
(define (y-shift local y0) (+ local y0))
```

In cells 1 and 2 of the array, YMaxCut_i = y_Rim_max + y0_i is used instead of y0_i + CellLength for all lower/central blocks in order to implement the Cut-Y cell as it can be seen in code lines below 3.17. Additionally, FILL_TOP_i is absent.

Listing 3.17: SDE: Array variations structure [15]

```
(define YMaxCut_1 (y-shift y_Rim_max y0_1))
...
(sdegeo:create-cuboid (position 0 y0_1 0)
                      (position BottomDielectricWidth YMaxCut_1 BottomDielectricThickness) "SiO2")
...
(sdegeo:create-cuboid (position TrenchWallWidth y0_1 z_MiddleGoldEnd)
                      (position x_2ndWallStart  YMaxCut_1 z_WallEnd) "Vacuum")
...
;; any FILL_TOP_1 (just bottom/left/right)
```

In the code section 3.18 it is implemented the Standard cell geometric parameter, in order to build the basic array/bus structure.

Listing 3.18: SDE: Array variations structure [15]

```
(define YMaxCut_3 (+ y0_3 CellLength))  ;; no cut
...
(define FILL_TOP_3
  (sdegeo:create-cuboid (position 0 y_Rim_max_3 z_WallEnd)
                        (position BottomDielectricWidth YMaxCut_3 z_2ndGoldLayerEnd) "SiO2"))
```

The top solids layout of Au contact, with Ti film around each sides of it, exhibit the identical geometry for each cell. This code lines explain how it is repeated the process for each cell according to all coordinates shifted by the respective y_*_i offsets.

Listing 3.19: SDE: Array TopContact for each cell [15]

```
;; ARRAY (for each i cells)
(define TOPCONTACT_i
  (sdegeo:create-cuboid (position x_Au_min y_Au_min_i z_WallEnd)
                        (position x_Au_max y_Au_max_i z_2ndGoldLayerEnd) "Gold"))
(sdegeo:bool-unite (list RIM_LEFT_i RIM_RIGHT_i RIM_BOTTOM_i RIM_TOP_i))

;; ARRAY (cell #1 & #2, CUT-Y): only BOTTOM/LEFT/RIGHT
(sdegeo:bool-unite (list FILL_BOTTOM_i FILL_LEFT_i FILL_RIGHT_i))

;; ARRAY (cell #3, STANDARD): TOP is present
(sdegeo:bool-unite (list FILL_BOTTOM_3 FILL_LEFT_3 FILL_RIGHT_3 FILL_TOP_3))
```

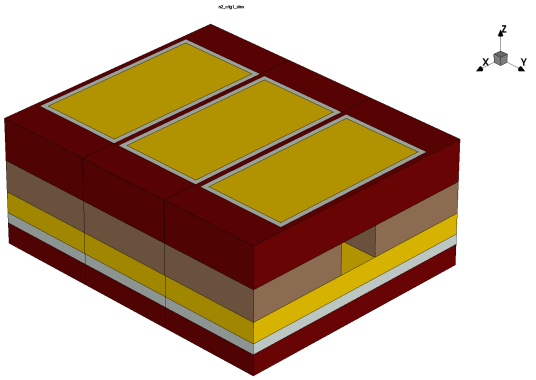
The most important aspect to underline are the next code lines, were the geometry contact definition are made. The BottomContact is the same for every cell, this means that the value for it is shared with all the cells, and remain unvariable. The concept is different for the TopContact value: every cells has is own value according to the relative phase in which the device is in a specific period of time. According to this each TopContact has is own variable name as shown below 3.20.

Listing 3.20: SDE: Array contact definition [15]

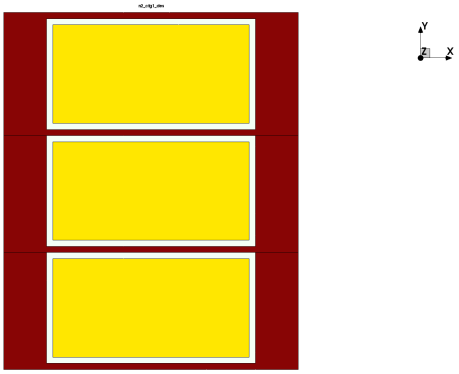
```
    ;; ARRAY
(sdegeo:define-contact-set "BottomContact" 4 (color:rgb 1 0 0) "##")
(sdegeo:set-contact BOTTOMCONTACT_1 "BottomContact")
(sdegeo:set-contact BOTTOMCONTACT_2 "BottomContact")
(sdegeo:set-contact BOTTOMCONTACT_3 "BottomContact")

(sdegeo:define-contact-set "TopContact_1" 4 (color:rgb 0 0 1) "##")
(sdegeo:set-contact TOPCONTACT_1 "TopContact_1")
(sdegeo:define-contact-set "TopContact_2" 4 (color:rgb 0 0 1) "##")
(sdegeo:set-contact TOPCONTACT_2 "TopContact_2")
(sdegeo:define-contact-set "TopContact_3" 4 (color:rgb 0 0 1) "##")
(sdegeo:set-contact TOPCONTACT_3 "TopContact_3")
```

The entire code lines are shown in the appendix section 4.

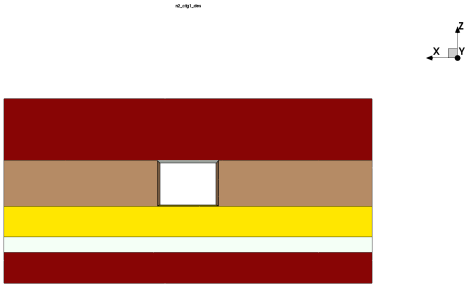
The resulting structure in fig.3.8 from Senturus Structure Editor SDE rendering.



(a) SDE: Array 3-Phase on XYZ axes



(b) SDE: Array 3-Phase on the XY plane

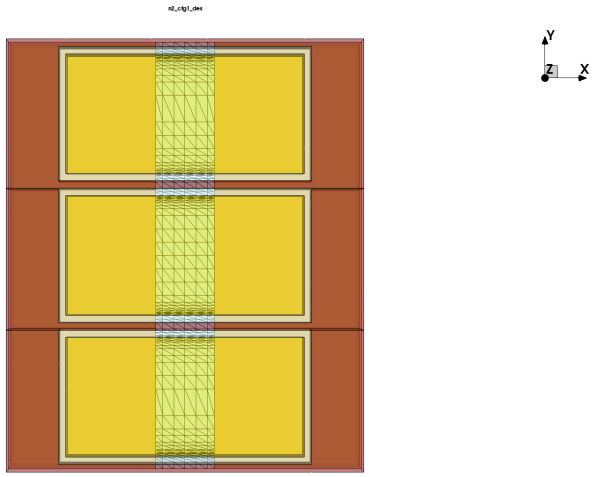


(c) SDE: Array 3-Phase on the XZ plane

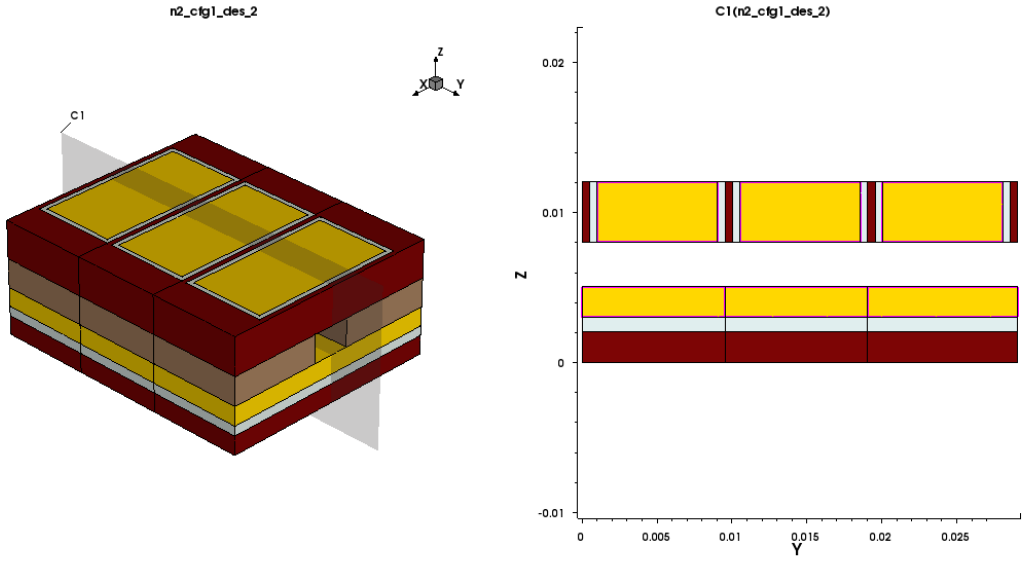
Figure 3.8: SDE project design overview of the Array 3-Phase structure from different perspectives.

In fig 3.9 we can see how the structure realized permit to have a linked channel all along the entire structure, were molecules can be placed (fig. 3.9a), and in particular in fig 3.9b all the

contacts that made up the structure.



(a) SDE: Vacuum Channel in the Array 3-Phase on XY plane



(b) SDE: Contacts in the section of Array 3-Phase on the YZ plane

Figure 3.9: SDE project design overview of the Array 3-Phase.

3.2.2 Array Standard Cell 3-Phase: Physics implementation in Sentaurus SDevice and visualization in Svisual

The relative code implementation of the Array 3-phase for SDevice, here appears interesting since the sequence of clock phases is shown . In particular here 3.21 are listed all the differences to the code lines of Sdevice for Standard cell and Cut-Y version as listed in the appendx 3.

Listing 3.21: SDevice: Differences in codes line of the Array 3 phase from Standard cell one [16]

```
=== Electrode ===

-Electrode {
-  { Name = "BottomContact"  Voltage = 0.0 }
+  { Name = "TopContact_1"   Voltage = 0.0 }
+  { Name = "TopContact_2"   Voltage = 0.0 }
+  { Name = "TopContact_3"   Voltage = 0.0 }}

=== Thermode ====

-Thermode {
-  { Name = "BottomContact" Temperature = 300 SurfaceResistance = 1e-5 }
+  { Name = "TopContact_1" Temperature = 300 SurfaceResistance = 1e-5 }
+  { Name = "TopContact_2" Temperature = 300 SurfaceResistance = 1e-5 }
+  { Name = "TopContact_3" Temperature = 300 SurfaceResistance = 1e-5 }}
```

To simulate the Clock cycle, 3 Phases of the cell were implemented: HIGH fase, related to a voltage value of $V = +3V$, an NULL fase, related to a voltage value of $V = 0V$ and a LOW fase, related to a voltage value of $V = -3V$, as we can see from the tables below 3.3 and 3.4.

Phase	Symbol	Value
High	H	+3V
Null	N	0V
Low	L	-3V

Table 3.3: Clock phase values.

CONFIG	Cell 1	Cell 2	Cell 3
1	H	N	L
2	L	H	N
3	N	L	H

Table 3.4: Clock Configurations of an array 3 phase.

To simplify the simulation, it was firstly formalize the clock-phase progression and the corresponding state transitions of each cell. Figure 3.10 illustrates the process from the inizialitazion of the device till the stable condition of clock phases state.
Starting from the initialization state (all cells rendered in brown), each cell's internal state is effectively unknown and may be considered "off." Once the device is powered on, an input (i.e., a High signal, H, represented by the red color) is assumed to arrive, while the other cells remain in their initialized state. This is followed by a LOW phase (L) (represented by blue color), which still belongs to the initialization procedure and serves to "clear" the cell of any residual data. A subsequent NULL phase (N) (represented by green color) provides a stabilization interval after

the reset. From this configuration onward, the system proceeds periodically through the sequence $H \rightarrow N \rightarrow L$. Accordingly, three steady configurations are considered—the last three shown in schematic 3.10.

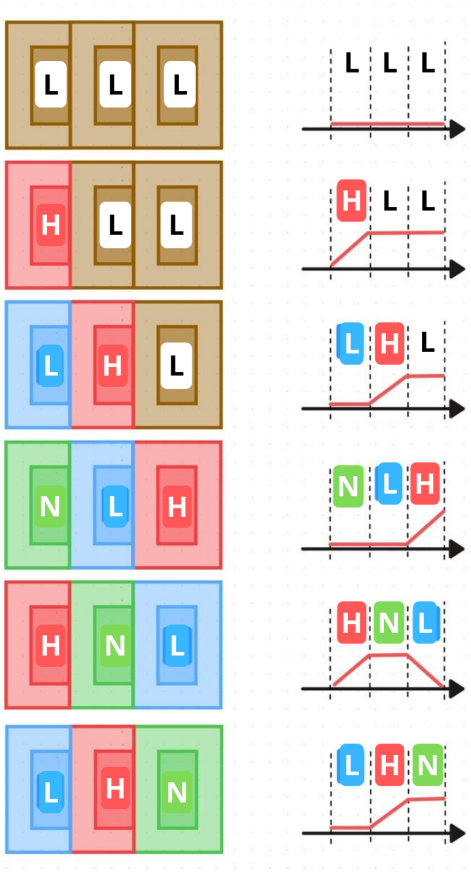


Figure 3.10: H/N/L CLK period assumption configuration

According to this, here are in 3.22 the code lines for the 3 configuration, in particular are underlined with different color in relation to the starting phase of the bus: 1) HNL, 2) NHL, 3)LHN. They are taken under analysis for Sdevice code lines.

Listing 3.22: SDevice: 3 Configuration of CLK phases of the Array 3-Phase [16]

```
Solve {
  Coupled (Iterations= 100 LineSearchDamping= 1e-8) {Poisson}
  Coupled{ Poisson Temperature Contact CondInsulator }
  Plot(FilePrefix="n@node@_equilibrium")

  # ====================================================================
  # ==== CONFIG 1 (phase rotation H/N/L) ====
  quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
    Goal { name= "TopContact_1" voltage = 3 }
    Goal { name= "TopContact_2" voltage = 0 }
    Goal { name= "TopContact_3" voltage = -3 }
    plot { range=(0, 1) intervals= 1}
  ){coupled { Poisson Temperature CondInsulator }}

  Plot(FilePrefix="n@node@_cfg1")

  # ---- return to 0 V on all top contacts
  quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
    Goal { name= "TopContact_1" voltage = 0.0 }
    Goal { name= "TopContact_2" voltage = 0.0 }
    Goal { name= "TopContact_3" voltage = 0.0 }
    plot { range=(0, 1) intervals= 1}
  ){coupled { Poisson Temperature CondInsulator }}

  # ====================================================================
  # ==== CONFIG 2 (phase rotation N/L/H) ====
  quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
    Goal { name= "TopContact_1" voltage = 0 }
    Goal { name= "TopContact_2" voltage = -3 }
    Goal { name= "TopContact_3" voltage = 3 }
    plot { range=(0, 1) intervals= 1}
  ){coupled { Poisson Temperature CondInsulator }}

  Plot(FilePrefix="n@node@_cfg2")

  # ---- return to 0 V on all top contacts
  quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
    Goal { name= "TopContact_1" voltage = 0.0 }
    Goal { name= "TopContact_2" voltage = 0.0 }
    Goal { name= "TopContact_3" voltage = 0.0 }
    plot { range=(0, 1) intervals= 1}
  ){coupled { Poisson Temperature CondInsulator }}

  # ====================================================================
  # ==== CONFIG 3 (phase rotation L/H/N) ====
  quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
    Goal { name= "TopContact_1" voltage = -3 }
    Goal { name= "TopContact_2" voltage = 3 }
    Goal { name= "TopContact_3" voltage = 0 }
    plot { range=(0, 1) intervals= 1}
  ){coupled { Poisson Temperature CondInsulator }}

  Plot(FilePrefix="n@node@_cfg3")
```

```
54
55  # ---- return to 0 V on all top contacts
56  quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
57    Goal { name= "TopContact_1" voltage = 0.0 }
58    Goal { name= "TopContact_2" voltage = 0.0 }
59    Goal { name= "TopContact_3" voltage = 0.0 }
60    plot { range=(0, 1) intervals= 1}
61  ){coupled { Poisson Temperature CondInsulator }}}
```

3.2.3 Array Standard Cell N-Phase: MATLAB GUI Design and Sentaurus Integration

This section documents a customizable tool in MATLAB designed to create a parametric Array N-Phase for nanoscale devices, as already shown in sections 3.1.3 and 3.1.6. It comes with simple controls for arranging these pieces into patterns either grids or plus shapes useful for decision-making circuits.
The principal difference from the previous codes is that, in addition to the already modifiable parameters, which otherwise remain unchanged, this version also exposes the choice of the number (N) of cells (with one Cut-Y cell automatically added), and the voltage value that can be customize by the user interface. The selected voltage is then propagated and adapted to the SDevice solve deck that is generated through the dedicated button. This creates a reliable process: choose settings, build shapes using MATLAB, then generate meshes and assign materials within the simulation software. Matlab Codes were implemented with a Graphical User Interface (GUI) in the relative Matlab file *stdcell_array_gui.m*, shown in fig 3.11

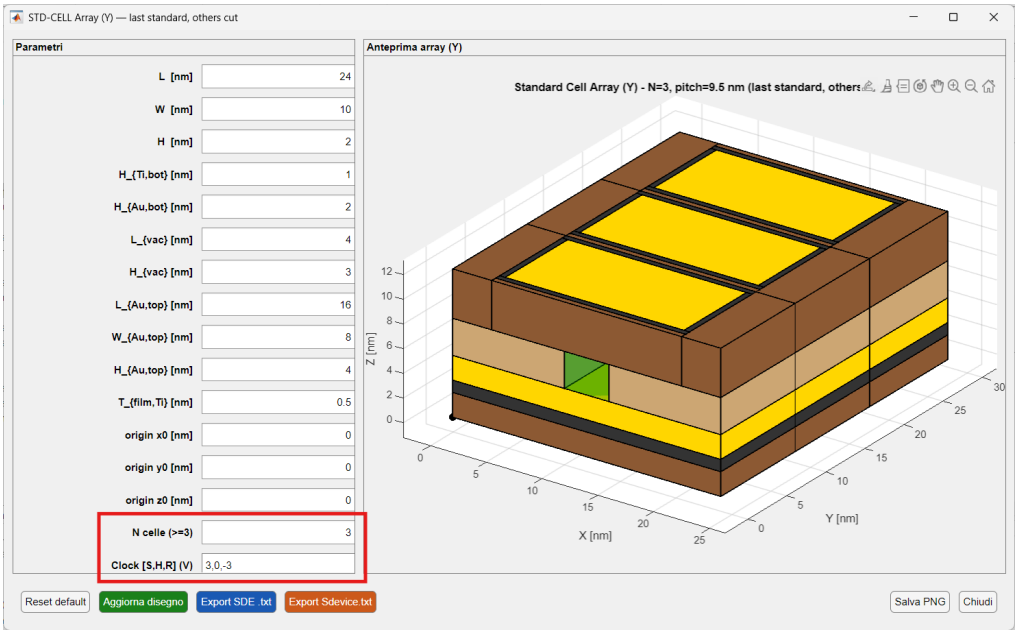


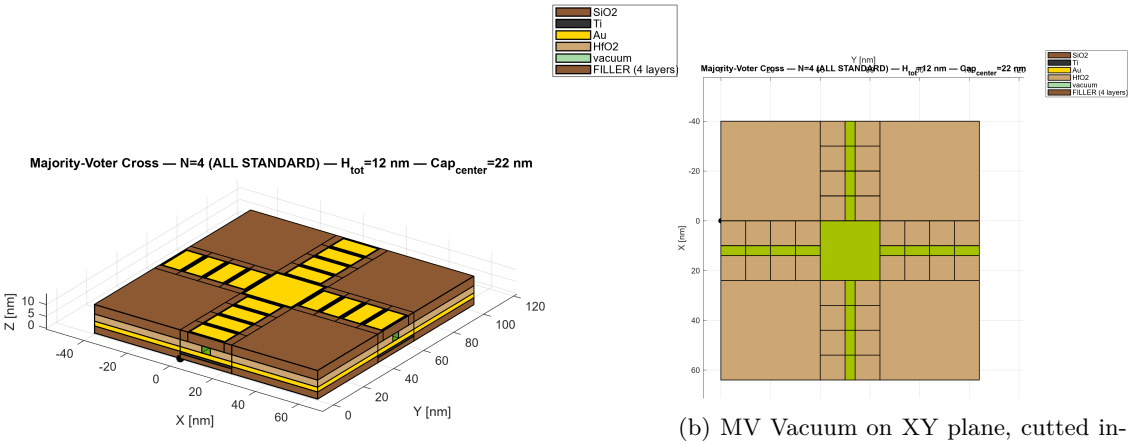
Figure 3.11: Matlab GUI for Array N-Phase, with N=3

All the value are customizable by the user from the Matlab GUI according to relations explained in the table 3.2.

3.3 Majority Voter

The most important implementation in this project is for sure the Majority Voter. Since its logic function, 2 type of MV were designed, according to the structure of the logic center cell, the one designed to make the logical operation between inputs:

- **MV Vacuum**: the center square cell has an entire layer of vacuum that replace the totality of HfO_2.The design is shown in fig. 3.12

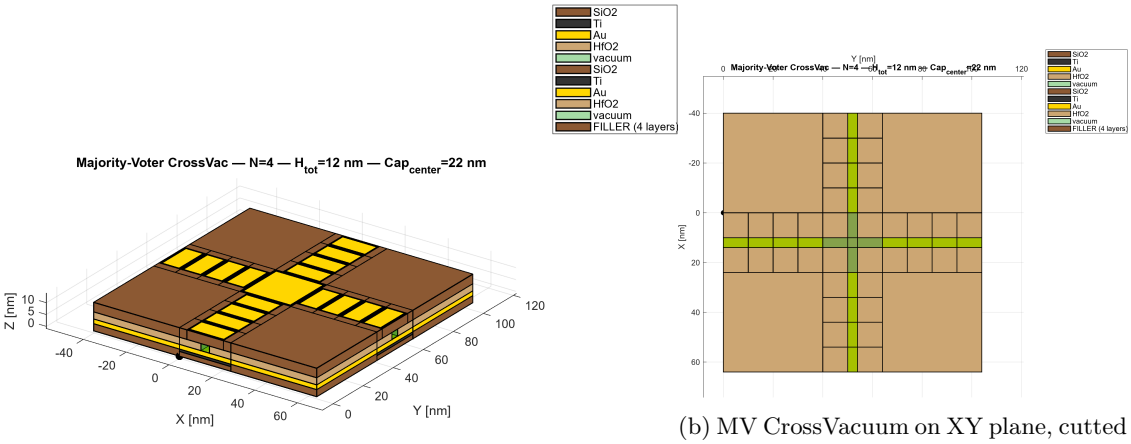


(a) MV Vacuum on XYZ axis

(b) MV Vacuum on XY plane, cutted inside vacuum trench

Figure 3.12: Majority Voter Vacuum Design

- **MV Cross Vacuum**: the center square cell has a vacum trench cross shaped, filled with HfO_2.The design is shown in fig. 3.13



(a) MV CrossVacuum on XYZ axis

(b) MV CrossVacuum on XY plane, cutted inside vacuum trench

Figure 3.13: Majority Voter Cross Vacuum Design

3.3.1 Majority Voter with vacuum center block: Geometrical Design in Sentaurus Structure Editor

This section explains how this version of the majority voter with center cell "Vacuum" was implemented in Sentaurus using SDE code, with commentary on the key portions of the script.

The entire code is reportend in the appendix section 5. From here on, longer code lines are reported partially in order to underline their meaning.

This first section 3.23 puts in place the shape controls along with some handy numbers that'll be used later on throughout the project. The main parameters declare all first-order dimensions (in µm) and materials for a single cell: in-plane sizes of the dielectric base and cell length, the vertical stack thicknesses, the trench aperture across the cell, the titanium rim thickness around the top pad, a maximum side for the central cap, and the global origin. It also fixes how many cells per arm are intended (N=4 cells for this case study) and names the bulk side material (BlockMaterial). The derived numbers give steady extra results based on these inputs. TrenchWallWidth means what's left of the insulating space beside each trench. H_tot adds up the full height from base to top metal layer. RimPackWidth roughly shows how much room the pad and rim take together along the arm. But Pitch sets a workable gap between nearby cells, keeping pads and rims from bumping into one another. The helper mk-cuboid's just a small tool, builds a box shape in 3D using a material, start point, or size. Instead of messy repeats, it makes later geometry neater; cuts down clutter from basic commands.

Listing 3.23: SDE: MV Vacuum initialization [15]

```
; --------------------------------
; MAIN PARAMETERS
; --------------------------------
(sde:define-parameter "BottomDielectricWidth"   0.024000)
...
(sde:define-parameter "BlockMaterial"           "HfO2")

; --------------------------------
; DERIVED & CONSTANTS
; --------------------------------
(sde:define-parameter "TrenchWallWidth"
...
               CellLength))

; --------------------------------
; HELPERS
; --------------------------------
(define (mk-cuboid name mat x y z dx dy dz)
...
                        mat))
```

This piece of code 3.24 sets up each groups of contacts for each breanch of the MV. It picks one lower contact at the base layer, BottomContact, while placing split upper ones, each tagged using cell number (from 1 to 4) combined with side position (North, South, East and West). Besides these, there's also an extra contact, TopContact_X, which is the contact for the central cell of the MV. Those name tags let apply voltage valuesto specific areas separately, so simulations can handle how cells interact without overlapping controls. This contacts variable separation is meaningful for the CLK phase value application.

Listing 3.24: SDE: MV Vacuum Contacts [15]

```
; --------------------------------
; CONTACT SETS
; --------------------------------
(sdegeo:define-contact-set "BottomContact" 4 (color:rgb 1 0 0) "##")

(sdegeo:define-contact-set "TopContact_south_1" 4 (color:rgb 0 0 1) "##")
...

(sdegeo:define-contact-set "TopContact_east_4" 4 (color:rgb 0 0 1) "##")

(sdegeo:define-contact-set "TopContact_X" 4 (color:rgb 0 0 1) "##")
```

The CellY part of the code 3.25 makes a single cell lined up along Y. First, it puts down the base layers (SiO_2, Ti, Au (BottomContact). On top of that, it carves out a gap filled with HfO_2+Vacuum+HfO_2 stretching across W_cell in the X direction. Next comes the upper Au piece centered and all along +Y direction. Afterward, a Ti film border wraps around the gold patch, while leftover spaces get packed with SiO_2 so the metal stays sealed and tidy.

Listing 3.25: SDE: MV Vacuum CellY [15]

```
; --------------------------------
; CELL +Y (with named top contact)
; --------------------------------
(define (CellY prefix x y z Wcell isCut cname)
  (mk-cuboid (string-append prefix "_SiO2") "SiO2" x y z
  ...
  zcap dxcore Tback TopGoldThicknessZ)))
```

The CellX in the next code lines 3.26 follow the same logic of 3.25.

Listing 3.26: SDE: MV Vacuum CellX [15]

```
; --------------------------------
; CELL +X (with named top contact)
; --------------------------------
(define (CellX prefix x y z WcellX isCut cname)
  (mk-cuboid (string-append prefix "_SiO2") "SiO2" x y z
  ...
  zcap dxcore Tback TopGoldThicknessZ)))
```

CornerLayers are the new introduction in geometrical design, since the MV is designed to be symmetric and squared, to facilitate the design process for building complexer structure. This part 3.27 builds a square-shaped, four-part fill near any edge using a set width. From level z upward, it adds: (L1) SiO_2 until reaching the Ti+Au base, (L2) Au linked to BottomContact, (L3) HfO_2 matching the trench height, then (L4) a final SiO_2 cover on top. It seals the outside border smoothly with full layers, fits neatly into grids while skipping narrow voids at each corner of the cross.

Listing 3.27: SDE: MV Vacuum Filler Cells [15]

```
; -------------------------------
; CORNER FILLERS (4 layers)
; -------------------------------
(define (CornerLayers prefix x y z span)
  (define zc z)
  (mk-cuboid (string-append prefix "_L1_SiO2") "SiO2" x y zc span span (+ BottomDielectricThickness
      AdhesionLayerThickness))
  (set! zc (+ zc (+ BottomDielectricThickness AdhesionLayerThickness)))
  (define GOLD_FILL (mk-cuboid (string-append prefix "_L2_Au") "Gold" x y zc span span
     GoldLayerThickness))
  (sdegeo:set-current-contact-set "BottomContact")
  (sdegeo:set-contact GOLD_FILL "BottomContact")
  (set! zc (+ zc GoldLayerThickness))
  (mk-cuboid (string-append prefix "_L3_HfO2") "HfO2" x y zc span span TrenchWallHeight)
  (set! zc (+ zc TrenchWallHeight))
  (mk-cuboid (string-append prefix "_L4_SiO2") "SiO2" x y zc span span TopGoldThicknessZ))
```

This block 3.28 puts together the entire structure. It positions three Cut cells alongside one regular cell on the southern (Y-) arm. Then it creates the center stack with the contact labeled TopContact_X. Next, it flips the setup toward the north (Y+), placing a normal cell close to the middle first, then adding cut ones, following a symmetrical to the center cell pattern. On the eastern (X+) and western (X-) sides, it repeats this pattern using CellX, giving each pad its own top contact name, so they can be powered separately. In the end, it calculates how far the arms stretch and adds four fill-in stacks at the corners (NE, NW, SW, SE) to seal off the outer edges for smoother mesh generation.

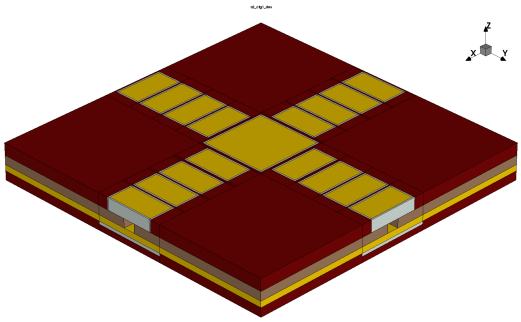
Listing 3.28: SDE: MV Vacuum overall structure building [15]

```

;; GEOMETRY BUILD (N=4)
;; ====================================================================

;; SOUTH branch (Y-): cut..cut..std near center
(define y_cursor y0)
(CellY "Ypre1" x0 y_cursor z0 Pitch #t "TopContact_south_1") (set! y_cursor (+ y_cursor Pitch))
(CellY "Ypre2" x0 y_cursor z0 Pitch #t "TopContact_south_2") (set! y_cursor (+ y_cursor Pitch))
(CellY "Ypre3" x0 y_cursor z0 Pitch #t "TopContact_south_3") (set! y_cursor (+ y_cursor Pitch))
(CellY "Ypre4" x0 y_cursor z0 CellLength #f "TopContact_south_4") (set! y_cursor (+ y_cursor
    CellLength))

; ================= CENTER BLOCK _ layered stack + TopContact_X
...
 TbackC TopGoldThicknessZ))

; NORTH branch (Y+): std near center, then cuts
(set! y_cursor (+ yB BottomDielectricWidth))
...
"TopContact_north_1") (set! y_cursor (+ y_cursor Pitch))

;; EAST branch (X+): std near center, then cuts
(define xB2 (+ xB BottomDielectricWidth))
...
"TopContact_east_4") (set! x_cursor (+ x_cursor Pitch))

;; WEST branch (X-): std near center, then cuts to the left
(set! x_cursor xB)
...
(CornerLayers "F_SE" xB2 (- yB arm_span) z0 arm_span)
```

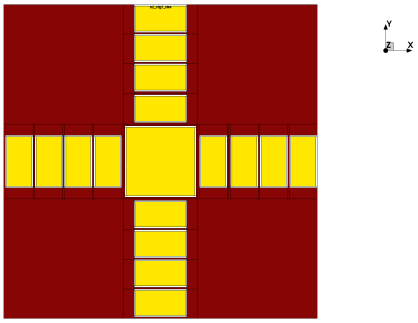
The SDE code ends with applying the mesh to the structure and build it to create the relative

file.

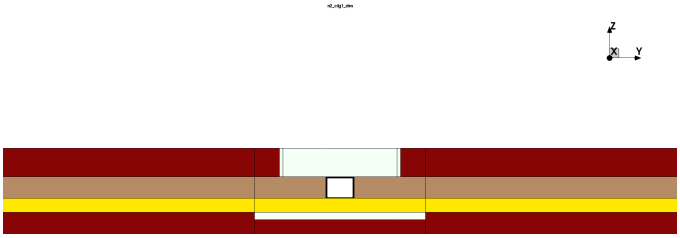
The entire code lines are shown in the appendix section 5. The resulting structure in fig.3.14 from Senturus Structure Editor SDE rendering.



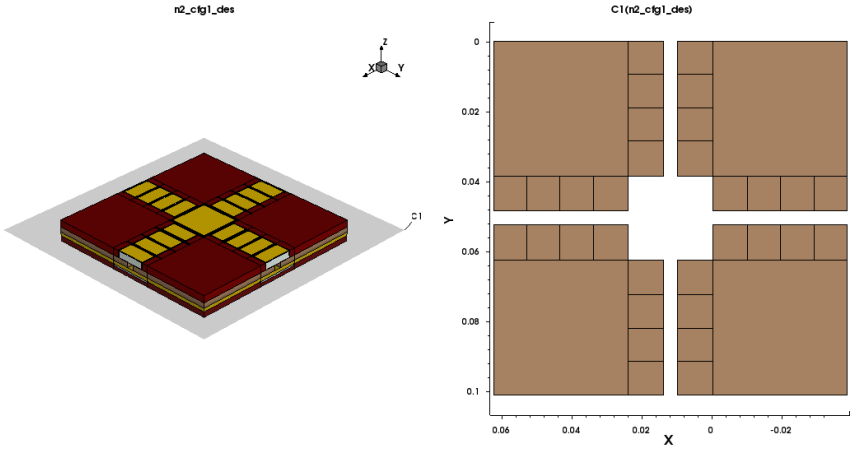
(a) SDE: MV Vacuum on XYZ axis.



(b) SDE: MV Vacuum on XY plane.



(c) SDE: MV Vacuum on XZ axis.



(d) SDE: Vacuum trench of the MV Vacuum

Figure 3.14: SDE project design overview of the Majority Voter Vacuum.

In fig 3.14d we can see how the structure of the center cell is characterize by the total empty layer of vacuum.

3.3.2 Majority Voter with vacuum center block: Physics implementation in Sentaurus SDevice and visualization in Svisual

As noted in 3.2.2, the device is driven by cyclic CLK phases that are applied consistently with the branch layout shown in Figs. 3.12 and 3.13. In the majority voter architecture, three branches serve as inputs and the remaining branch acts as the output, while the central block performs as logical operator of the inputs. Each top contact is assigned an independent control variable, as defined in the SDE code at lines 3.24, enabling the SDevice scripts to realize the three phase configurations specified in Fig. 3.10.

The initialization of the SDevice code is the same shown in code lines 3.21, with the analog inzialitation for all the TopContact declared in the SDE codes. The logic followed to assign to each contact a determinated value, is the same followed in code lines 3.17. All the 3 configuration were implemented:

- **HIGH → NULL → LOW**
  1° Configuration shown from lines [1-67] in 3.29 (red part)
- **LOW → HIGH → NULL**
  2° Configuration shown from lines [68-76] in 3.29 (blue part)
- **NULL → LOW → HIGH**
  3° Configuration shown from lines [76-83] in 3.29 (green part)

Listing 3.29: SDevice: 3 Configuration of CLK phases of the Array 3-Phase [16]

```
1  Solve {
2    Coupled (Iterations= 100 LineSearchDamping= 1e-8) {Poisson}
3    Coupled{ Poisson Temperature Contact CondInsulator }
4    Plot(FilePrefix="n@node@_equilibrium")
5
6    # H -> N -> L
7    # ======================================================================
8    # ==== CONFIG (cfg1) with startPhase=S; inputs 1..N, center N+1, east N+2..N+1+N ====
9    quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
10
11   Goal{ name = "BottomContact"                  voltage = 0.0 }
12
13   Goal{ name = "TopContact_south_1"     voltage = 3.0 }
14   Goal{ name = "TopContact_south_2"     voltage = 0.0 }
15   Goal{ name = "TopContact_south_3"     voltage = -3.0 }
16   Goal{ name = "TopContact_south_4"     voltage = 3.0 }
17
18   Goal{ name = "TopContact_north_1"     voltage = 3.0 }
19   Goal{ name = "TopContact_north_2"     voltage = 0.0 }
20   Goal{ name = "TopContact_north_3"     voltage = -3.0 }
21   Goal{ name = "TopContact_north_4"     voltage = 3.0 }
22
23   Goal{ name = "TopContact_west_1"      voltage = 3.0 }
24   Goal{ name = "TopContact_west_2"      voltage = 0.0 }
25   Goal{ name = "TopContact_west_3"      voltage = -3.0 }
26   Goal{ name = "TopContact_west_4"      voltage = 3.0 }
27
28   Goal{ name = "TopContact_X"                   voltage = 0.0 }
29
30   Goal{ name = "TopContact_east_1"      voltage = -3.0 }
```

```
  Goal{ name = "TopContact_east_2"      voltage = 3.0 }
  Goal{ name = "TopContact_east_3"      voltage = 0.0 }
  Goal{ name = "TopContact_east_4"      voltage = -3.0 }

        plot { range=(0, 1) intervals= 1}
  ){coupled { Poisson Temperature CondInsulator }}

  Plot(FilePrefix="n@node@_cfg1")

  # ---- return to 0 V on all top contacts
  quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
  Goal{ name = "BottomContact"                  voltage = 0.0 }

  Goal{ name = "TopContact_south_1"     voltage = 0.0 }
  Goal{ name = "TopContact_south_2"     voltage = 0.0 }
  Goal{ name = "TopContact_south_3"     voltage = 0.0 }
  Goal{ name = "TopContact_south_4"     voltage = 0.0 }

  Goal{ name = "TopContact_north_1"     voltage = 0.0 }
  Goal{ name = "TopContact_north_2"     voltage = 0.0 }
  Goal{ name = "TopContact_north_3"     voltage = 0.0 }
  Goal{ name = "TopContact_north_4"     voltage = 0.0 }

  Goal{ name = "TopContact_west_1"      voltage = 0.0 }
  Goal{ name = "TopContact_west_2"      voltage = 0.0 }
  Goal{ name = "TopContact_west_3"      voltage = 0.0 }
  Goal{ name = "TopContact_west_4"      voltage = 0.0 }

  Goal{ name = "TopContact_X"                   voltage = 0.0 }

  Goal{ name = "TopContact_east_1"      voltage = 0.0 }
  Goal{ name = "TopContact_east_2"      voltage = 0.0 }
  Goal{ name = "TopContact_east_3"      voltage = 0.0 }
  Goal{ name = "TopContact_east_4"      voltage = 0.0 }

        plot { range=(0, 1) intervals= 1}
  ){coupled { Poisson Temperature CondInsulator }}


  # L -> H -> N
  # ======================================================================
  # ==== CONFIG (cfg2) with startPhase=R; inputs 1..N, center N+1, east N+2..N+1+N ====
  quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
...
  ){coupled { Poisson Temperature CondInsulator }}


  # N -> L -> H
  # ======================================================================
  # ==== CONFIG (cfg3) with startPhase=H; inputs 1..N, center N+1, east N+2..N+1+N ====
  quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
...
  Plot(FilePrefix="n@node@_cfg3")}
```

The entire code is shown in the appendix 6.

3.3.3 Majority Voter with vacuum center block Standard Cell N-Phase: MATLAB GUI Design and Sentaurus Integration

This section documents a customizable tool in MATLAB designed to create a parametric Majority Voter Vacuum type for nanoscale devices, as already shown in sections 3.1.3, 3.1.6 and 3.2.3. Matlab Codes were built firstly from the file *stdcell_mv.m*, and then implemented with a Graphical User Interface (GUI) in the relative Matlab file *stdcell_mv_gui.m*, shown in fig. 3.15.

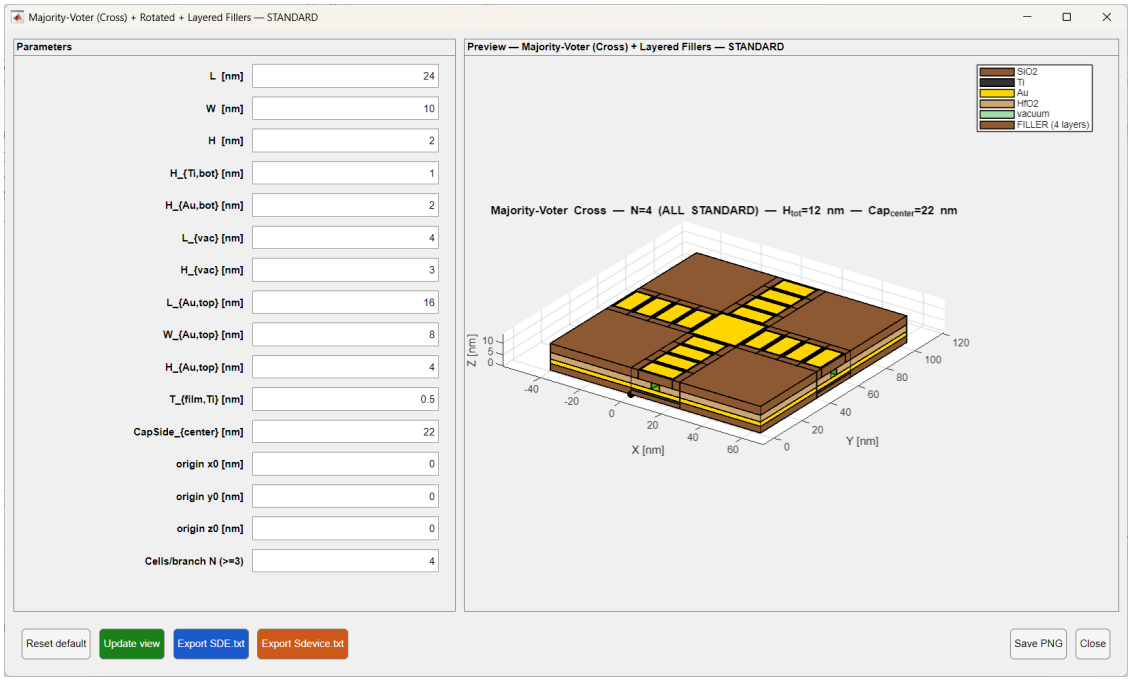


Figure 3.15: Majority Voter Vacuum Matlab GUI

3.3.4 Majority Voter with cross vacuum center block: Geometrical Design in Sentaurus Structure Editor

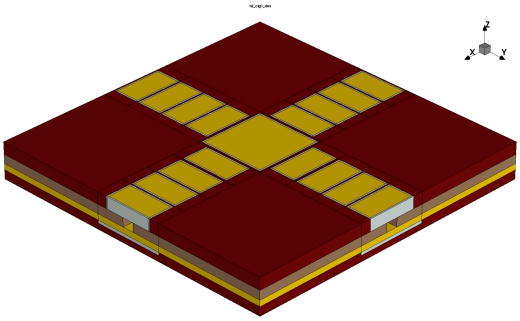
In the second version of the Majority Voter named "*CrossVacuum*", the big change, already mentioned (in section 3.3) and showed in fig. 3.13), is just around the central block. There, instead of leaving empty space over the gold piece as showed in fig. 3.12, we fit a cross-shaped gap inside an HfO_2 layer. Looking at this twist matters since it shows whether reshaping that core helps hold steady when processing signals across timing steps. So, only the updated chunk of script for this center zone appears here in code lines 3.30 in green color, the rest stays the same as the earlier air-gap design (refered to red lines).

Listing 3.30: SDE: MV CrossVacuum Extract Center Block variation from MV Vacuum type [15]

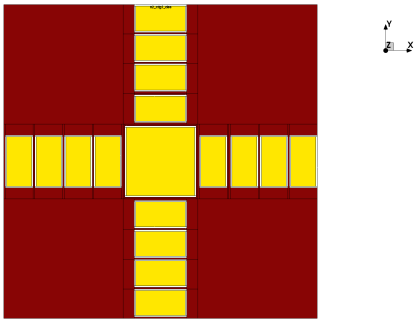
```
1
2  (sde:clear)
3  ...
4  CellLength))
5
6
7  ; ================= CENTER BLOCK _ HfO2 mid + Cross vacuum  TopContact_X
8  (define xB x0)
9  (define yB y_cursor)
10
11 (mk-cuboid "BLOCK_SiO2" "SiO2" xB yB z0 BottomDielectricWidth BottomDielectricWidth
       BottomDielectricThickness)
12 (mk-cuboid "BLOCK_TiB" "Titanium" xB yB (+ z0 BottomDielectricThickness) BottomDielectricWidth
       BottomDielectricWidth AdhesionLayerThickness)
13 (define AUB_CENTER (mk-cuboid "BLOCK_AuB" "Gold" xB yB (+ z0 BottomDielectricThickness
       AdhesionLayerThickness) BottomDielectricWidth BottomDielectricWidth GoldLayerThickness))
14 ...
15 (if (and (> TbackC 0.0)  (> dxcoreC 0.0)) (mk-cuboid "BLOCK_SiO2FrB" "SiO2" xin_min yin_max zcapC
       dxcoreC TbackC TopGoldThicknessZ))
16
17
18 ================================================
19 ; NORTH branch (Y+): std near center, then cuts
20 (set! y_cursor (+ yB BottomDielectricWidth))
21 ...
22 (sde:build-mesh "n@node@")
23 ================================================
24
```

The entire Code lines variation are shown in the appendix in code 7.

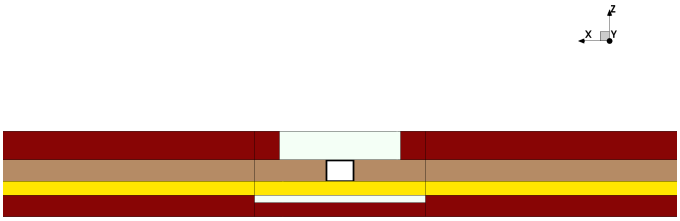
The resulting structure in fig.3.16 from Senturus Structure Editor SDE rendering.



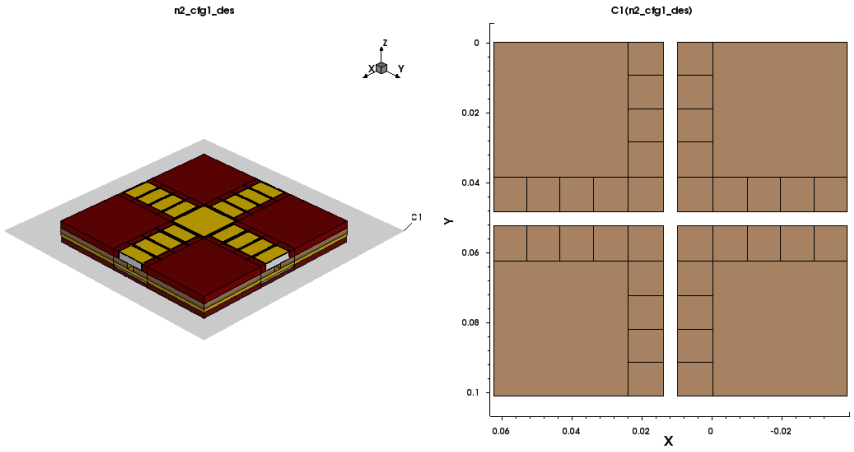
(a) SDE: MV CrossVacuum on XYZ axis.



(b) SDE: MV CrossVacuum on XY plane.



(c) SDE: MV CrossVacuum on XZ axis.



(d) SDE: CrossVacuum trench of the MV Vacuum

Figure 3.16: SDE project design overview of the Majority Voter CrossVacuum.

3.3.5 Majority Voter with cross vacuum center block: Physics implementation in Sentaurus SDevice and visualization in Svisual

The SDevice code lines are exactly the one explained in section 3.3.2.

3.3.6 Majority Voter with cross vacuum center block Standard Cell N-Phase: MATLAB GUI Design and Sentaurus Integration

This section documents a customizable tool in MATLAB designed to create a parametric Majority Voter CroosVacuum type for nanoscale devices, as already shown in previous sections as 3.1.6 and 3.2.3. Matlab Codes were built firstly from the file *stdcell_mv_crossvac.m*, and then implemented with a Graphical User Interface (GUI) in the relative Matlab file *stdcell_mv_crossvac_gui.m*, shown in fig. 3.17, particularly slide along Z axis to show the cross vacuum pattern of the center block.

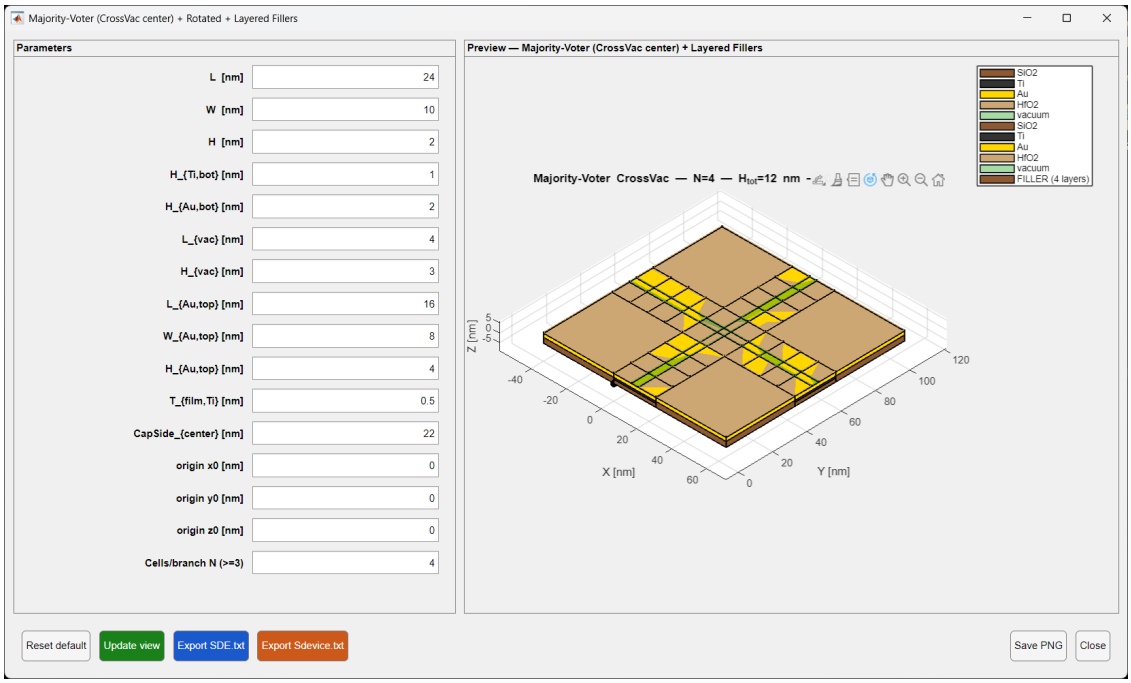
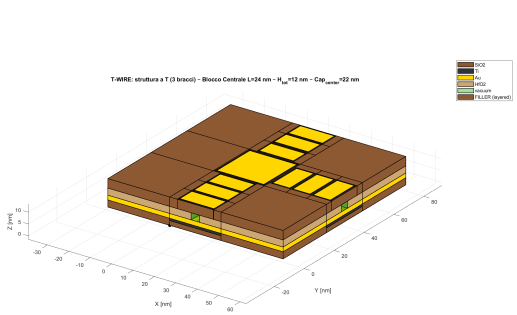


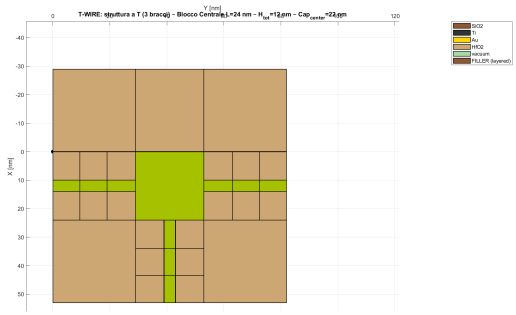
Figure 3.17: Majority Voter CrossVacuum Matlab GUI: 3D rendering cutted with XY plane in the middle of HfO_2 layer

3.3.7 T Wire and L Wire

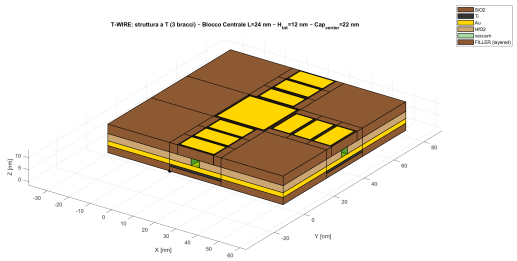
Two more basic circuit elements are derived in the following subsections of this project by applying geometric design modifications to the Majority Voter structure in both its Vacuum and CrossVacuum versions. The first is the T-wire, which is a component with two inputs and one output that is created by cutting off one branch from the majority voter. The second is the L-wire, which is a straight-through element with a single input and output that is produced by keeping only two of the Majority Voter's neighboring branches. In figures 3.18 3.19 are shown respectivetly the two idea designs.



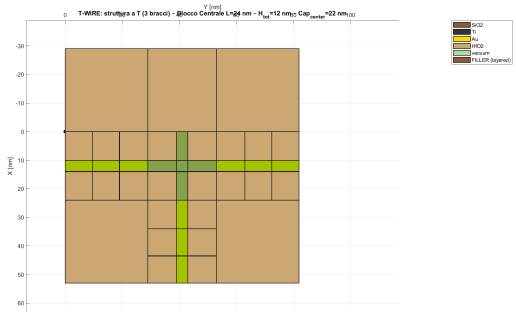
(a) T Wire Vacuum on XYZ axis



(b) T Wire Vacuum on XY plane.

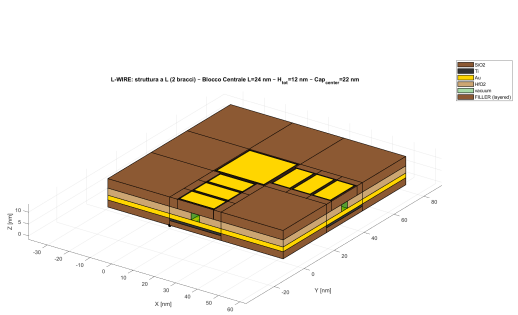


(c) T Wire CrossVacuum on XYZ axis.

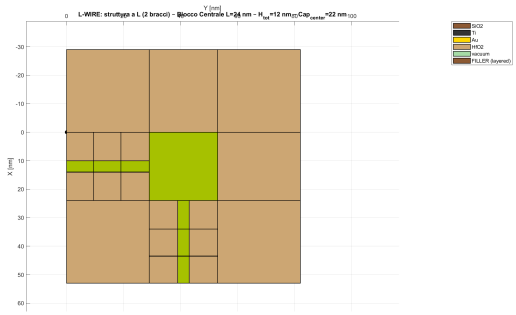


(d) T Wire CrossVacuum on XY plane.

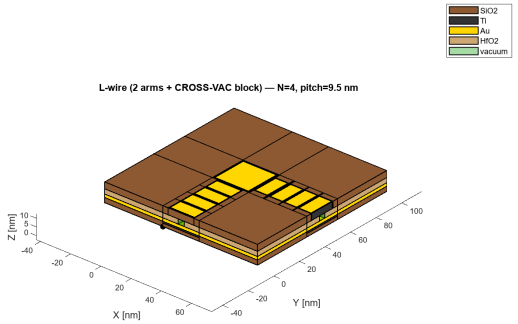
Figure 3.18: T wire in both variation: Vacuum and CrossVacuum.



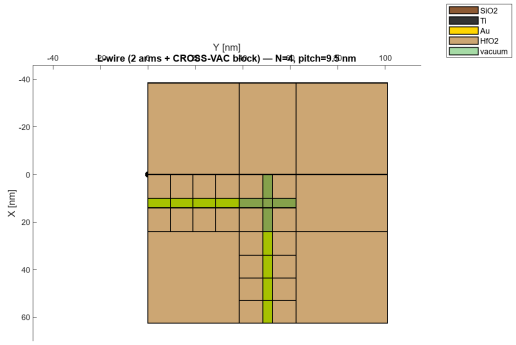
(a) L Wire Vacuum on XYZ axis



(b) L Wire Vacuum on XY plane.



(c) L Wire CrossVacuum on XYZ axis.

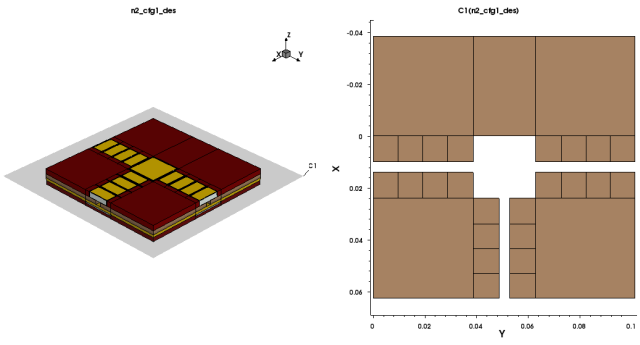


(d) L Wire CrossVacuum on XY plane.

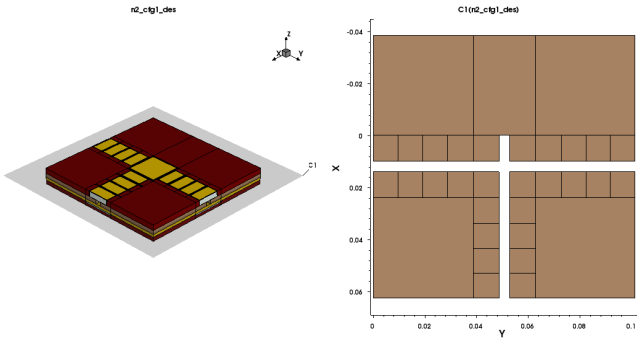
Figure 3.19: L wire in both variation: Vacuum and CrossVacuum.

T & L wire: Geometrical Design in Sentaurus Structure Editor

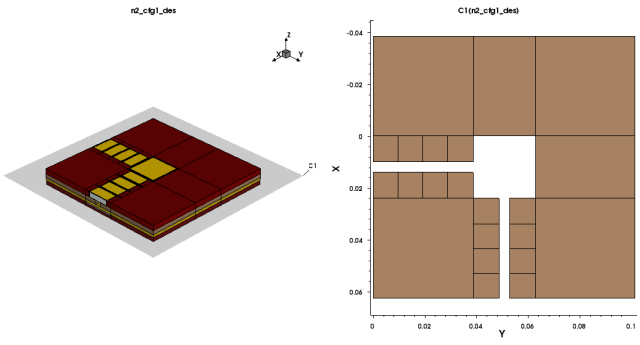
As observed in the preceding sections, the geometric modifications introduced for the simpler logic elements (T-wire and L-wire) are deliberately designed to streamline and modularize the overall Sentaurus Device Editor (SDE) scripting process, directly extending the methodology already established and thoroughly exemplified in the majority voter implementation. In the interest of avoiding unnecessary repetition and preserving the conciseness of the present discussion, it suffices to emphasize that, for both the T-wire vacuum and T-wire cross-vacuum configurations, the only substantive difference lies in the definition of the central active block, which is adapted according to the specific variant under consideration—precisely mirroring the modular strategy previously adopted for the majority voter, see the differences code lines in 5 and 7. The remainder of the script, which describes the peripheral structural components surrounding this central block, adheres to an identical topological layout in both T-wire variants.
An entirely analogous design philosophy and code-reuse strategy is applied to the L-wire family, encompassing both its vacuum and cross-vacuum realizations, thereby ensuring maximal consistency across the entire library of QCA-inspired devices examined in this work. The complete geometries generated with the Sentaurus Device Editor (SDE) for these structures are reported in the listings that follow.



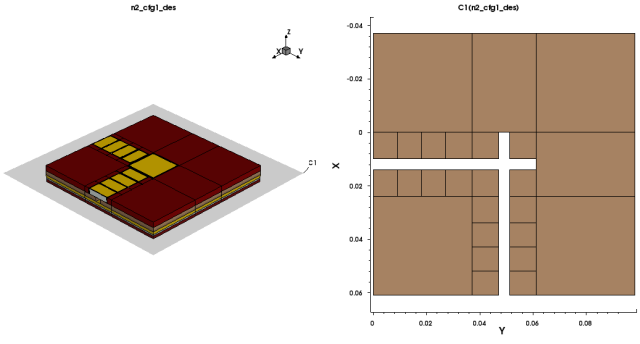
(a) SDE: T-wire (vacuum configuration)



(b) SDE: T-wire (cross-vacuum configuration)



(c) SDE: L-wire (vacuum configuration)



(d) SDE: L-wire (cross-vacuum configuration)

Figure 3.20: Geometries of T-wire and L-wire structures in vacuum and cross-vacuum configurations generated with Sentaurus Device Editor (SDE). Top two panels: T-wire variants; bottom two panels: L-wire variants.

T & L wire: Physics implementation in Sentaurus SDevice , visualization in Svisual and Matlab GUI

Similarly, the Sentaurus Device Editor (SDE) code for the T-wire and L-wire structures directly reuses the modular framework established for the majority voter as shown in appendix 6, requiring only minimal modifications to account for the reduced number of arms. Specifically, in both the T-wire vacuum and T-wire cross-vacuum configurations, the west arm present in the majority voter is entirely omitted, whereas all remaining geometric components are retained without alteration. Variation follow in 3.31.

Listing 3.31: SDevice: Differences in codes line of T wire N=4 cells from MV [16]

```
...
Electrode {
*BOTTOMCONTACT
  { Name = "BottomContact"                      Voltage = 0.0 }
*INPUTS SEQUENCES TOPCONTACT
  { Name = "TopContact_south_1"   Voltage = 0.0 }
  { Name = "TopContact_south_2"   Voltage = 0.0 }
  { Name = "TopContact_south_3"   Voltage = 0.0 }
  { Name = "TopContact_south_4"   Voltage = 0.0 }

  { Name = "TopContact_north_1"   Voltage = 0.0 }
  { Name = "TopContact_north_2"   Voltage = 0.0 }
  { Name = "TopContact_north_3"   Voltage = 0.0 }
  { Name = "TopContact_north_4"   Voltage = 0.0 }

-  { Name = "TopContact_west_1"   Voltage = 0.0 }
-  { Name = "TopContact_west_2"   Voltage = 0.0 }
-  { Name = "TopContact_west_3"   Voltage = 0.0 }
-  { Name = "TopContact_west_4"   Voltage = 0.0 }

*CENTER CELL TOPCONTACT
  { Name = "TopContact_X"       Voltage = 0.0 }
*OUTPUT TOPCONTACT
  { Name = "TopContact_east_1"   Voltage = 0.0 }
  { Name = "TopContact_east_2"   Voltage = 0.0 }
  { Name = "TopContact_east_3"   Voltage = 0.0 }
  { Name = "TopContact_east_4"   Voltage = 0.0 }}
```

An analogous simplification is applied to the L-wire family (vacuum and cross-vacuum variants), in which both the west and north arms of the original majority-voter layout are removed, leaving only the east and south arms connected to the central active region.Variation follow in 3.32.

Listing 3.32: SDevice: Differences in codes line of T wire N=4 cells from MV [16]

```
...
Electrode {
*BOTTOMCONTACT
  { Name = "BottomContact"                      Voltage = 0.0 }
*INPUTS SEQUENCES TOPCONTACT
-  { Name = "TopContact_south_1"   Voltage = 0.0 }
-  { Name = "TopContact_south_2"   Voltage = 0.0 }
-  { Name = "TopContact_south_3"   Voltage = 0.0 }
-  { Name = "TopContact_south_4"   Voltage = 0.0 }

  { Name = "TopContact_north_1"   Voltage = 0.0 }
  { Name = "TopContact_north_2"   Voltage = 0.0 }
  { Name = "TopContact_north_3"   Voltage = 0.0 }
  { Name = "TopContact_north_4"   Voltage = 0.0 }

-  { Name = "TopContact_west_1"   Voltage = 0.0 }
-  { Name = "TopContact_west_2"   Voltage = 0.0 }
-  { Name = "TopContact_west_3"   Voltage = 0.0 }
-  { Name = "TopContact_west_4"   Voltage = 0.0 }

*CENTER CELL TOPCONTACT
  { Name = "TopContact_X"       Voltage = 0.0 }
*OUTPUT TOPCONTACT
  { Name = "TopContact_east_1"   Voltage = 0.0 }
  { Name = "TopContact_east_2"   Voltage = 0.0 }
  { Name = "TopContact_east_3"   Voltage = 0.0 }
  { Name = "TopContact_east_4"   Voltage = 0.0 }}
```

Owing to this high degree of structural inheritance, the complete SDE scripts are not reproduced here; the reader is referred to the majority-voter implementation (Listings X–Y) and to the parametric differences highlighted above.
This section documents a customizable tool in MATLAB designed to create a parametric Twire and Lwire in both Vacuum and Crossvacuum for nanoscale devices, as already shown in sections 3.2.3 and 3.3.3/3.3.6. Matlab Codes were implemented with a Graphical User Interface (GUI) in the relative Matlab files:

- "*stdcell_twire_gui.m*"(fig.3.21).
- "*stdcell_twire_crossvac_gui.m*"(fig.3.21).
- "*stdcell_lwire_gui.m*" (fig.3.22).
- "*stdcell_lwire_crossvac_gui.m*" (fig.3.22).

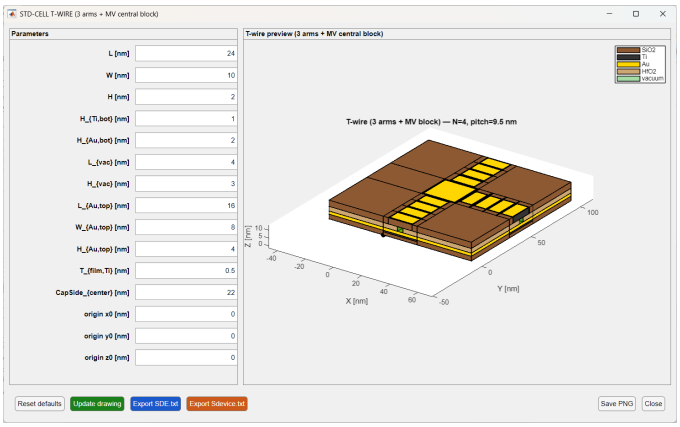


Figure 3.21: Matlab T wire Vacuum GUI (analog interface for CrossVacuum type)

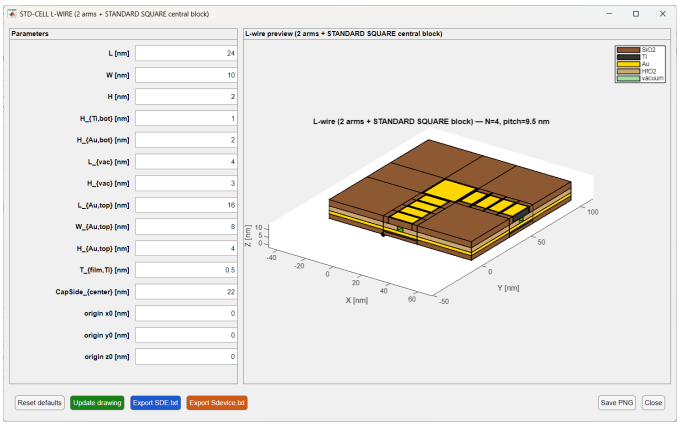


Figure 3.22: Matlab L wire Vacuum GUI (analog interface for CrossVacuum type)

Chapter 4

Costum Circuit Layout TOOL

In this part, it is shown what was done using MATLAB to build a visual tool letting users create adjustable circuits. This app gives designers the chance to pick basic building blocks from earlier sections, set how they're positioned on the grid, also specify shape, values for parameters and details for every single component. The interface works like a handy tool, giving to the user a quick, adaptable method to build the layout while at the same time creating matching input files for Sentaurus SDE and SDevice, very close to each GUI implementation done in the previous chapter 3. This means users shift smoothly from drawing ideas to running device simulations, cutting down on hand-written scripts and mistakes that might pop up. Also, since the code is made automatically, it's easier to test various designs quickly, so it is possible to check how each version performs electrically or structurally without extra hassle. In the following section it is shown how the tool was implemented.

4.1 Matlab GUI for the costum circuit layout

The automated design tool got built in MATLAB using a graphic interface called "launcher", seen in Figure 4.1. Through the main screen (file:"*launcher.m*"), users pick any device from earlier thesis parts to tweak settings or view right away, otherwise they open another deeper interface known as "*custom_layout_gui.m*".

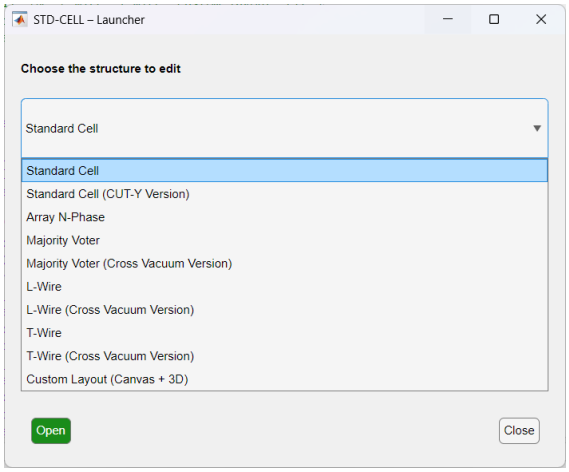
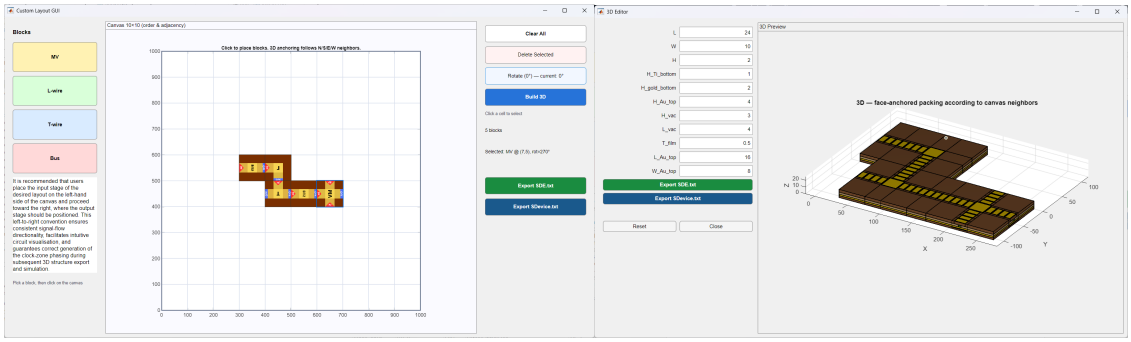


Figure 4.1: Launcher GUI Matlab.

As mentioned before, the *custom_layout_gui.m*" gives a flat workspace where you can build any circuit by dragging and linking basic parts, like the Majority Voter, T-wire, L-wire, or the Bus, which is just a straight three-part line. Instead of drawing from scratch, users put these pieces together freely; this speeds up creating new MolFCN designs without messing up timing zones or shapes. An example is shown in fig 4.2.



(a) Costum Layput Circuit GUI. (b) Relative 3D structure of the device

Figure 4.2: User Matlab interface of Custom Layout Circuit

In the screen from Figure 4.2a, users pick one of four core pieces. Go by the design tip shown here while arranging things:

> *"It is recommended that users place the input stage of the desired layout on the left-hand side of the canvas and proceed toward the right, where the output stage should be positioned. This left-to-right convention ensures consistent signal-flow directionality, facilitates intuitive circuit visualisation, and guarantees correct generation of the clock-zone phasing during subsequent 3D structure export and simulation."*

After picking the needed blocks, you put them together on the middle XY grid a flat space seen from above. Every basic block gets placed with the input side on the left, output on the right, so signals flow smoothly from left to right. While arranging this way isn't forced, it just makes sense and works well. Over on the right side of the screen, you'll find several buttons for working with your current layout. The interface comes with handy tools for tweaking the 2D canvas. Hit "Clear All" to wipe out all pasted blocks, this resets everything at once; on the flip side, you can yank single pieces if only some need removing. Once picked, items can turn in quarter turns whenever needed, so inputs and outputs line up right next to each other. A few extra windows show specifics on the chosen section, like shape details or how many components there are and size estimate. After finishing the 2D setup, hitting "Build 3D" brings up a 3D editing screen, seen in Figure 4.2b. This extra window, built with the same MATLAB logic as the rest (i.e. 3.3.7), lets you adjust every upright measurement (like layer depths, groove cut, sensor elevations) that stayed unchanged earlier during the flat layout step. From the 2D editor or the 3D setup screen, you hit a specific button, "Export SDE" or "Export SDevice", to save your finished design. Each one spits out full input files ready for Synopsys tools: SDE uses the first, SDevice takes the second. That way, you start voltage checks or test runs right away on your MolFCN chip. Keep in mind this version works only with vacuum setup, so every available block includes a hollow center space, no exceptions.

4.1.1 Examples of circuits made with the custom circuit layout TOOL

In this section, several example circuit structures created using the Custom Layout Circuit tool will be presented.

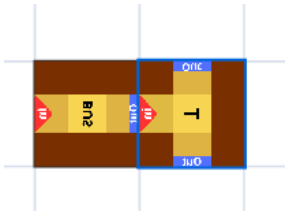
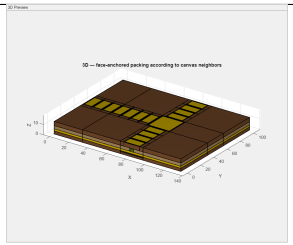
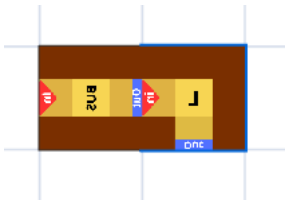
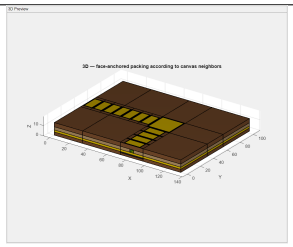
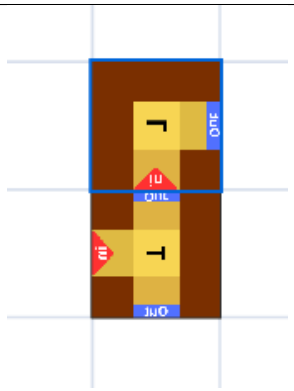
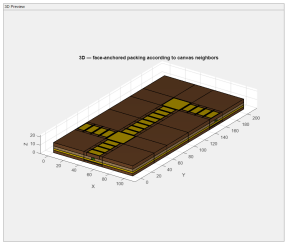
2D Canvas Layout	3D Rendering Structure	Description
		Bus + T Wire structure
		Bus + L Wire structure
		T Wire + L Wire structure

Table 4.1: Examples of complex structures (I).

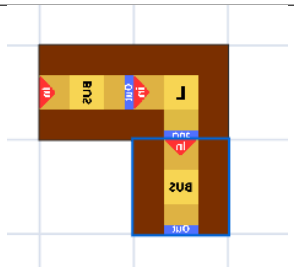
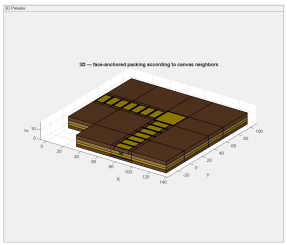
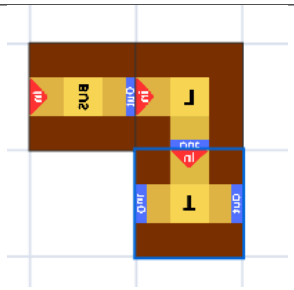
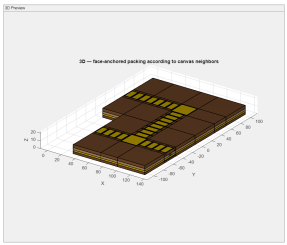
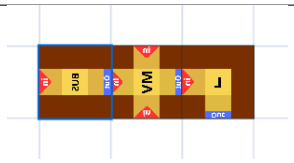
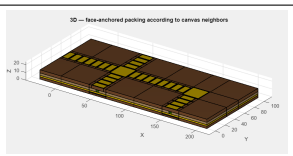
2D Canvas Layout	3D Rendering Structure	Description
		Bus + L Wire + Bus structure
		Bus + L Wire + T Wire structure
		L + Majority Voter + Bus

Table 4.2: Examples of complex structures (II).

As pointed out, this tool lets you export working input decks straight into Sentaurus Structure Editor or SDevice. Instead of tweaking things by hand, the output adjusts itself based on the layout and shape details drawn on screen. Files come out ready for sim runs thanks to auto-generated scripts shaped by your design setup. The method used till this section underlined a full scripted setup for the SDE input, that's no longer in use for this GUI. Now, thanks to MATLAB's extra freedom, the SDE code gets built on the fly. It lays out every needed cuboid step by step, tagging each with its own contacts and mesh settings. The output is a precise 3D model, fresh from the GUI, good to go. What you see matches exactly what Sentaurus runs. No more mismatches from split design methods.

Here follows examples of SDE structure codes 4.1

Listing 4.1: SDE:Example of code structure for a complex device

```
;; ======================================================
;; SENTARUS SDE _ Exported from MATLAB GUI (MV/L/T/Bus)
;; Geometry emitted as explicit cuboids already rotated & placed.
;; Any box with non-empty name becomes a contact region.
;; Date: 2025-11-22 18:58:51
;; ======================================================
(sde:clear)
(define (mk-cuboid x1 y1 z1 x2 y2 z2 mat)
  (sdegeo:create-cuboid (position x1 y1 z1) (position x2 y2 z2) mat))
;; ---- Global parameters (nm) ----
(sde:define-parameter "L" 0.024)
(...)
(sde:define-parameter "NcellsArm" 4)
```

```
;; ================== GEOMETRY (from canvas, rotated & placed) ==================
;; ---- BLOCK 1 _BUS  rot=270deg  LL=(0, 0)
(define REG_001 (mkcuboid 7.10542736e_15 40 0 10 64 2 "SiO2"))
...
(define CNT_001 (mk-cuboid 1 44 8 9 60 12 "Gold"))
...
(define REG_050 (mk-cuboid 0 0 8 30 40 12 "SiO2"))
;; ---- BLOCK 2  _T-WIRE  rot=180deg  LL=(30, 0)
(define REG_051 (mk-cuboid 70 94 0 94 104 2 "SiO2"))
...
(define REG_250 (mk-cuboid 94 40 8 134 64 12 "SiO2"))
;; ================== CONTACTS (auto-detected) ==================
(sdegeo:define-contact-set "BottomContact" 4 (color:rgb 0.1 0.1 0.1) "##")
(sdegeo:set-current-contact-set "BottomContact")
(sdegeo:set-contact REG_003 "BottomContact")
...
(sdegeo:set-contact CNT_003 "TopContact_b_3")
;; ================== MESH & BUILD ==================
(sdedr:define-refinement-function "RFn" "MaxLenInt" "Gold" "Vacuum"   0.0001 "DoubleSide")
(sdedr:define-refinement-function "RFn" "MaxLenInt" "Gold" "Titanium" 0.0001)
(sde:build-mesh "n@node@")
```

Clock Phases implementation

Just like with the linear setup in 3.2.2, the layout tool now auto-sets clock phases too. It starts at the far-left block, the circuit's entry point, then steps through each next block from left to right. Each one gets its matching phase voltage along the way. When it hits the output block, or multiple outputs, it stops there. This keeps clock zones lined up exactly how the user set them, making sure signals move smoothly, correctly, and without energy waste across the whole design. The setup on the canvas follows clear guidelines picked carefully to work smoothly with a regular three-step timing method. In particular:

- The Bus block's made up of three molecular cells lined up straight.
- All other main blocks, like Majority Voter or T-wire along with L-wire, are built using arms that are precisely four cells long while also featuring a center cell.

This setup makes sure the cell count in every working circuit built on the canvas divides evenly by three. That repeating pattern fits exactly what's needed for a three-phase adiabatic clock method, so phases get set automatically without mistakes - also boosting how well and how steadily later Sentaurus runs perform. Because of this, the canvas layout was planned around that rule right away, keeping designs aligned with clock needs while not making things harder for users.

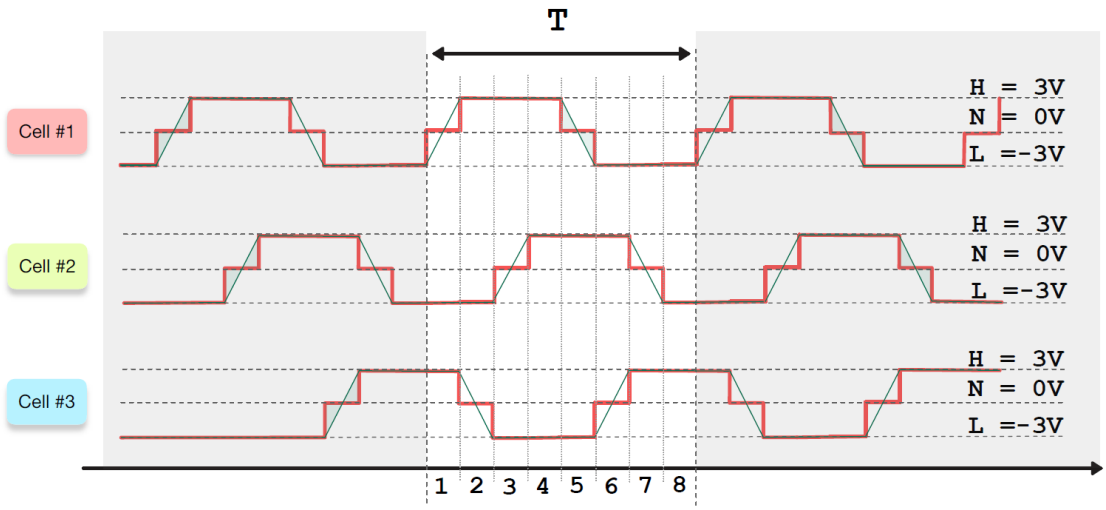


Figure 4.3: Clock sequence cycle of each cell

The chart in fig 4.3 shows how the timing signal moves through every part of the circuit. From the first cell on the left, each next one gets shifted by three ticks. Instead of lining up, they're spaced out step by step. This repeating sequence runs the same way across all cells. One complete loop holds eight unique setups. You'll find those listed in the table just beneath this 4.3. While some signals overlap, others don't match at any point. It has to be noticed that the graph with three cells plus their clock patterns uses a green line for the real wave, whereas the red one shows a rough, stair-like version to help mark clear voltage steps. You'll notice it switches back and forth from –3 volts to +3 volts, sitting briefly at 0 volts between changes.

CONFIG	Cell 1	Cell 2	Cell 3
1	N	L	H
2	H	L	N
3	H	N	L
4	H	H	L
5	N	H	L
6	L	H	N
7	L	N	H
8	L	L	H

Table 4.3: Clock Configurations of an array 3 cell, with 8 configuration.

In real use, every cell's clock signal graph displays how voltage changes over time on its hidden electrode, so it is shown how data move smoothly through the path implemented in the Matlab TOOL. The table 4.3 shows eight repeating patterns every three cells throughout the layout. These setups make up a full clock cycle, also ensuring the molecular QCA line works right. Here, code lines 4.2 shows a complete SDevice setup example, made up by Custum Layout Circuit TOOL; it is particularly refered to the structure composed by Bus, and L wire and another Bus, as shown in the fourth row of the table **??**. In that table, every strucuture has it's own SDE code and Sdevice code generated by the tool.

Listing 4.2: SDE:Example of code structure for a complex device

```
File {
  Grid = "n1_msh.tdr"
  Plot = "n@node@_clock_des.tdr"
  Current = "n@node@_clock_des.plt"
  Parameter = "sdevice.par"}
Electrode {
  { Name = "BottomContact"       Voltage = 0.0 }
...
  { Name = "TopContact_L_OUT_4"  Voltage = 0.0 }}
...
Solve {
  Coupled (Iterations= 100 LineSearchDamping= 1e-8) {Poisson}
  Coupled{ Poisson Temperature Contact CondInsulator }
  Plot(FilePrefix="n@node@_equilibrium")
  # ==== CONFIG (cfg1) NLH ====
  quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
  Goal{ name = "BottomContact"         voltage = 0.0  }
  Goal{ name = "TopContact_b_3"         voltage = 0.0  }
  Goal{ name = "TopContact_b_2"         voltage = -3.0 }
  Goal{ name = "TopContact_b_1"         voltage = 3.0  }
  Goal{ name = "TopContact_L_OUT_4"     voltage = 0.0  }
  Goal{ name = "TopContact_L_OUT_3"     voltage = -3.0 }
  Goal{ name = "TopContact_L_OUT_2"     voltage = 3.0  }
  Goal{ name = "TopContact_L_OUT_1"     voltage = 0.0  }
  Goal{ name = "TopContact_L_X"         voltage = -3.0 }
  Goal{ name = "TopContact_L_IN_4"      voltage = 3.0  }
  Goal{ name = "TopContact_L_IN_3"      voltage = 0.0  }
  Goal{ name = "TopContact_L_IN_2"      voltage = -3.0 }
  Goal{ name = "TopContact_L_IN_1"      voltage = 3.0  }
        plot { range=(0, 1) intervals= 1}
  ){coupled { Poisson Temperature CondInsulator }}
  Plot(FilePrefix="n@node@_cfg1")
  # ---- return to 0 V on all top contacts  [...]
  # ==== CONFIG (cfg2) HLN ====             [...]
  # ==== CONFIG (cfg3) HNL ====             [...]
  # ==== CONFIG (cfg4) HHL ====             [...]
  # ==== CONFIG (cfg5) NHL ====             [...]
  # ==== CONFIG (cfg6) LHN ====             [...]
  # ==== CONFIG (cfg7) LNH ====             [...]
  # ==== CONFIG (cfg8) LLH ====             [...]
...
        plot { range=(0, 1) intervals= 1}
  ){coupled { Poisson Temperature CondInsulator }}}
```

Since these are merely illustrative examples, the appendix does not include the full code for all simulated structures. Their implementation directly reuses and references the routines already developed and explicitly documented for the main structures analyzed in the thesis.

Chapter 5

Simulation Results

In this section, It has been shares every simulation run on Synopsys Sentaurus SDevice for the tiny structures from earlier parts, so it can be clear connect their shape with how they handle electric fields and electron flow.

First off, Section 1 shows what happened when we ran sims on the basic cell 3.1 the core piece used later in bigger designs. Instead of just listing outcomes, it walks through performance under normal conditions; also the Cut-Y version. Moving on, Section 2 dives into setups with multiple cells lined up 3.2, using repeating patterns to check how they interact when linked together. Rather than focusing only on design, it looks at shifts in behavior across the group. Then comes Section 3 , where we explore the majority voter setup3.3, basically a decision-making unit made with the exact same tech layers. Unlike simpler blocks, this one combines inputs to deliver logical outputs based on weighted responses. In this section also T and L wire are simulated. Lastly, Section 4 pulls in real circuit cases pulled straight from an automated layout tool built earlier in chapter 4; instead of theory, it demonstrates practical scaling, from individual parts right up to networks involving connections and logic chains.

All tests run with identical tech settings and setup tweaks from the early design steps, this way every layout shows how real hardware would behave using standard blocks. Focus stays on steady voltage patterns, where charges sit on molecules, yet also checks current flow at set connection points since those reveal whether logic outputs match predictions plus show how clocks influence behavior. When needed, it has been tweaked inputs step by step to see how shifts in size matter, while pointing out tensions among small footprint, clean signals, or ease of making them.

5.1 Standard Cell

5.1.1 Electric Field

The graphs in Fig. 5.1 display how strong the electric field is inside the regular cell when running on DC voltage, side by side, one with voltage applied (on the left), another without any voltage, just balanced (on the right). When powered, there's a 10-volt gap between the lower connector set at zero volts and the upper one set at ten volts. Most of the action happens in the layered core tucked between both conductors: high intensity shows up near the HfO_2 or empty space zone, also around the outer tips of the top gold patch, simply because the field curves there and spreads out slightly, boosting strength locally. The SiO_2 layers stay under weak field, showing most voltage falls where it should, across the active part. When no voltage is on, the setup sits still; electric effects fade out nearly everywhere except near borders between materials. Tiny leftovers show up there but nothing more. That means any real push comes only when you apply

5–8 V, not before. The structure acts like a clean stratifyed layout: metal, insulator, metal, all lined up right without leaks or surprises.

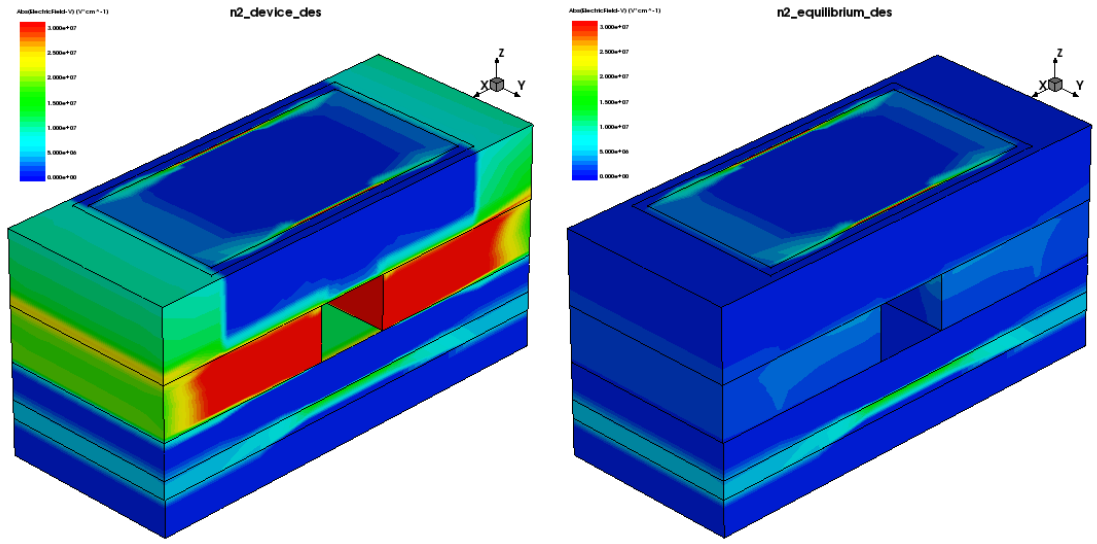


Figure 5.1: Electric Field applied on the Standard Cell; on the right in the equilibrium state, on the left BottomoContact = 0V, TopContact = 10V

In this next outcome, the EF is simulated across a flat slice (see fig. 5.2, also through a straight line:

- Left panel ($C2(n2\_device\_des)$): The layout reveals a slice from (x) to (z) across the regular cell. Colors show how strong the electric field is, whereas dark arrows point which way it flows. Peak levels pop up once more near the $vacuum/HfO_2$ area stacked by upper and lower contacts, showing visible spread at contact borders plus sharp spots on the top cover's edges. Field strength drops in the SiO_2 foundation along with metal parts.
- Right panel ($Abs(ElectricFiel - V)(C1C2)$): The red line shows how strong the electric field is along the horizontal path $C1 - C2$, right at the level where molecules sit. Near the two ends, where fields spike because of edge spreading, the shape stays mostly steady, hovering near $3.3 \cdot 10^7 V/m$ over much of the working area. That means when we apply 0 V on the bottom and 10 V up top, the middle section gets a pretty even push, but uneven spots only stick close to the electrode borders.

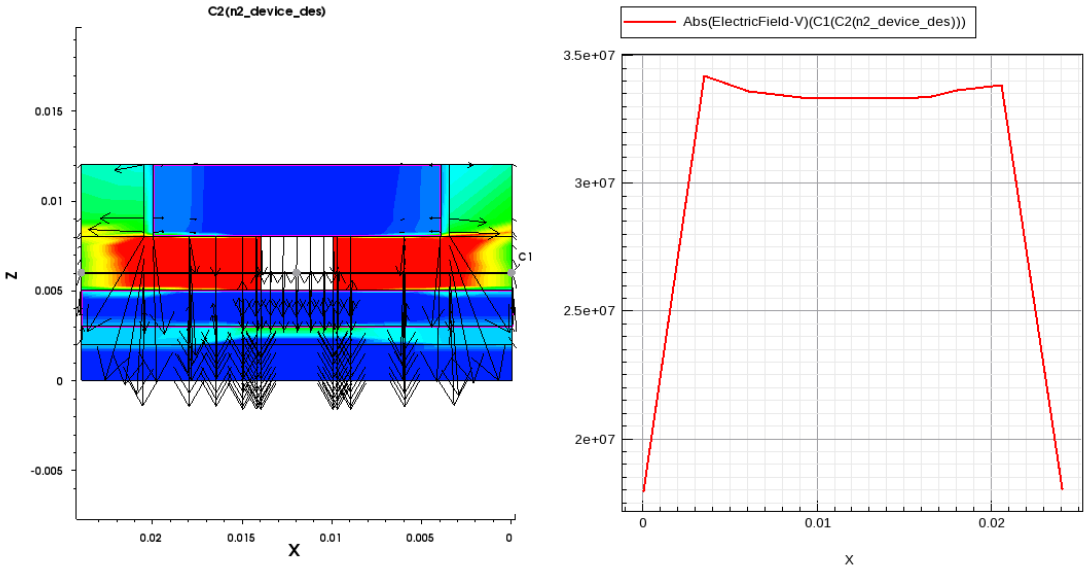


Figure 5.2: Cut section XZ at Y= 5nm (exactly in the middle of the structure) of the Standard Cell to underline EF Vectors

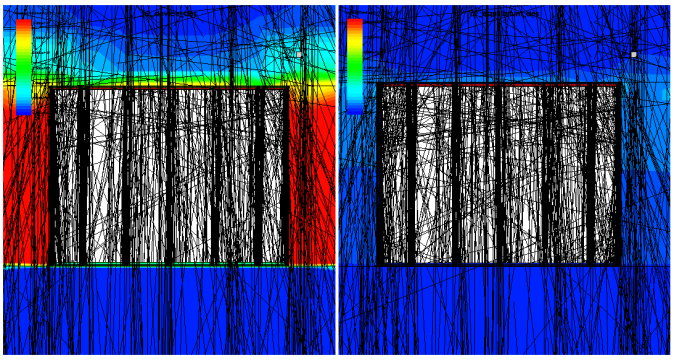


Figure 5.3: Closer visualization of the EF vectors inside the vacuum trench (Cut Section Y = 5nm)

Here's a graph 5.4that shows how strong the electric field is across the Y direction within the empty space, along line C1, at one steady level between the upper and lower plates. In the left pane, colours plus arrows show the field's mostly up-down in the vacuum trench, stronger near electrode edges. The right side shows it clearly, notice how the line plot forms a U-shape, tracing EF against Y, dipping close to the middle of the setup while rising toward the edges on either side. This behavior makes sense if you think of it like two flat plates with limited width: in the middle, the field stays mostly even, but near the sides, the lines bend and pile up, making the field stronger there. So for molecules sitting in the center area, treating the field as nearly constant works fine, whereas those closer to the edges feel a bit more push because of how the field spreads out at boundaries.

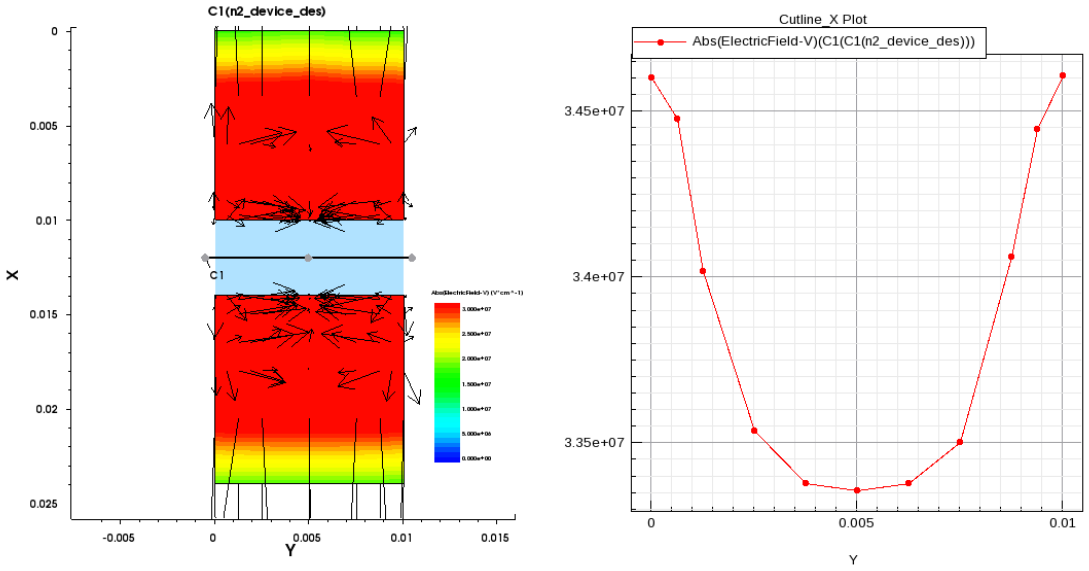


Figure 5.4: Cut section XY at Z= 6nm (exactly $1nm$ upper the Bottom Contact) of the Standard Cell to underline EF Vectors and values

5.1.2 Total Current Density

Figure 5.5 shows how strong the current is in a powered device compared to one at rest. When voltage goes from 0 to 10 V, the flow stays nearly negligible, hitting highs around $10^{-5} A/cm^2$ mostly piling up close to electrode borders or where the electric push gets intense across the insulating layers. Outside those tiny zones, the signal plummets fast, losing tons of strength, and nears the bottom edge of what the image can show. That pattern suggests the voltage just shifts internal pressure around inside the layered setup, but doesn't punch a real conductive trail through it. Specifically, gaps and thick HfO_2 films block most movement, letting only faint seepage sneak through, even under forces as big as $10^7 V/cm$. In the balanced setup (right side), the whole thing looks like there's almost no current flowing, current levels are nearly zero everywhere, plus tiny leftover bits seem just like computer glitches from the math tool. Comparing both voltage setups shows the layered design blocks current really well, keeping the device stable even when stray DC tries to sneak through. The cutline in fig. 5.6 shows that, when a 0–10 V voltage is used, the overall current flow across the active layer stays nearly steady, around $10^{-5} A/cm^2$ in the sideways direction. Near the electrode tips, though, it plunges sharply down to near-zero levels, signaling how conduction shifts from slight leaks through insulation to zones with basically no current. Along this line, there's no sign of strong localized surges or concentrated high-current areas.

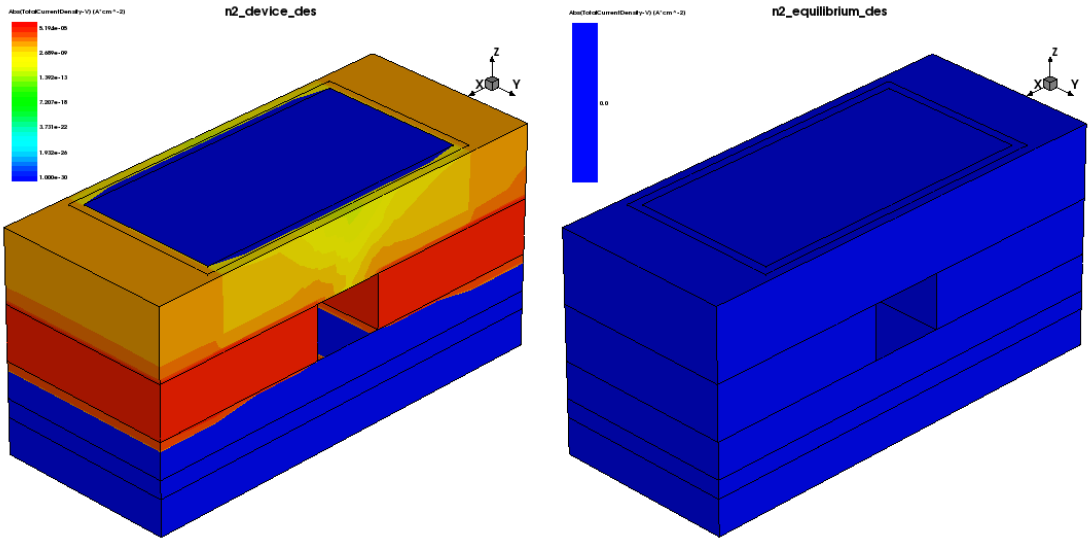


Figure 5.5: Total Current Density for the Standard Cell

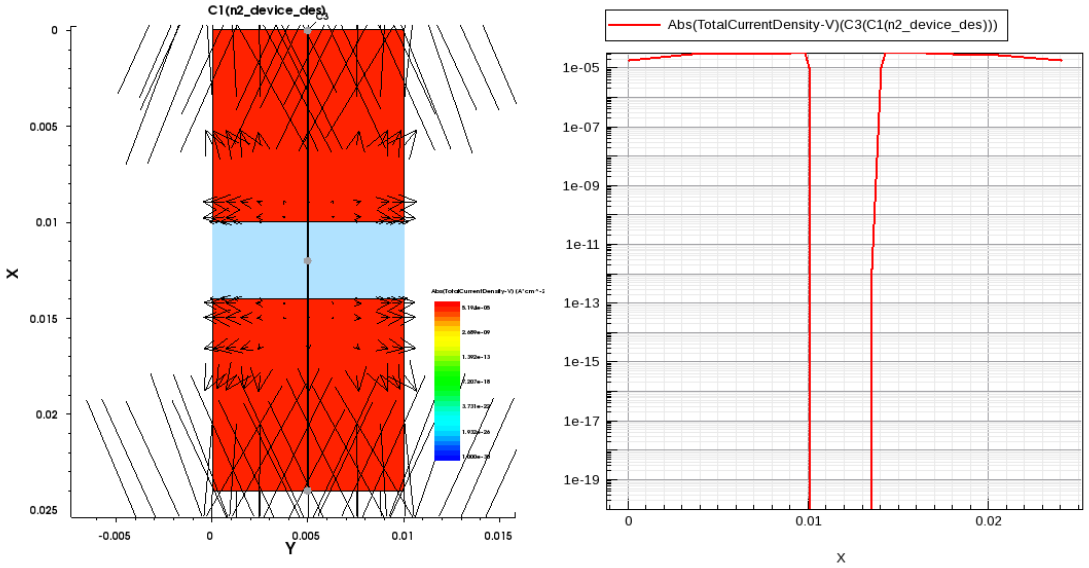


Figure 5.6: Cut section XY at Z= 6nm (exactly $1nm$ upper the Bottom Contact) of the Standard Cell to underline Total Current Density Vectors and values

5.1.3 Electrostic Potential

Figure 5.7 shows how electrostatic potential (EP) spreads inside the regular cell when powered (on the left), while also revealing its natural state when no voltage is applied (on the right). In the tilted setup, the upper electrode gets pushed to about 10 volts while the lower connection stays near zero. The color pattern reveals most voltage shift happens through the insulating layers separating the contacts: a steady high level forms above the top gold pad (shown in red), but the bottom $SiO_2/Ti/Au$ parts stay low (in blue). A clear jump links these levels, tracing the material borders, though slight spreading and curving of equal-voltage paths show up at contact edges and beside the hole walls, where fields pile up. In balance, with no outside push, the voltage shift across the gadget stays under a few hundred millivolts. Metals act like flat zones where voltage hardly changes, while insulators show slight natural shifts, tiny slopes just near surface edges. That means the layered structure works nearly as a perfect blocker when idle; during active use, big voltage drops come straight from outer electrode pushes, not inner charge buildup.

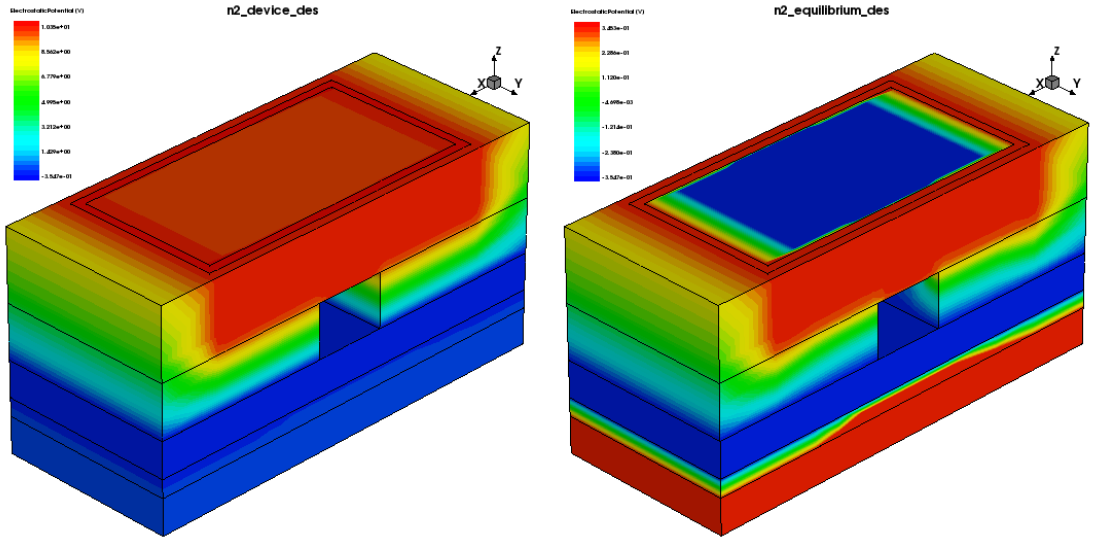


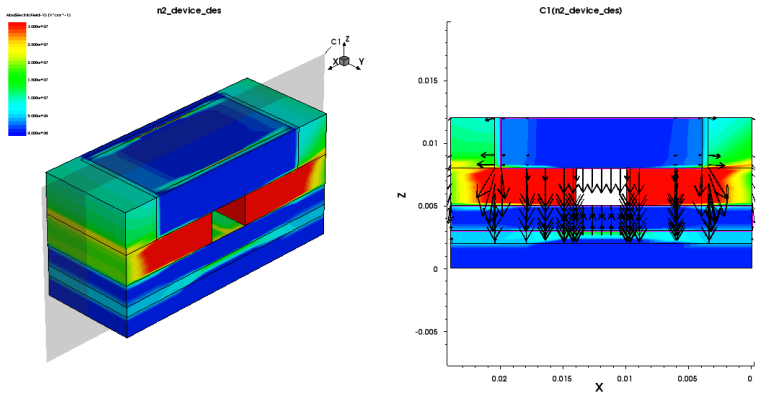
Figure 5.7: Electrostatic Potential on the Standard Cell

5.1.4 Cut-Y Version

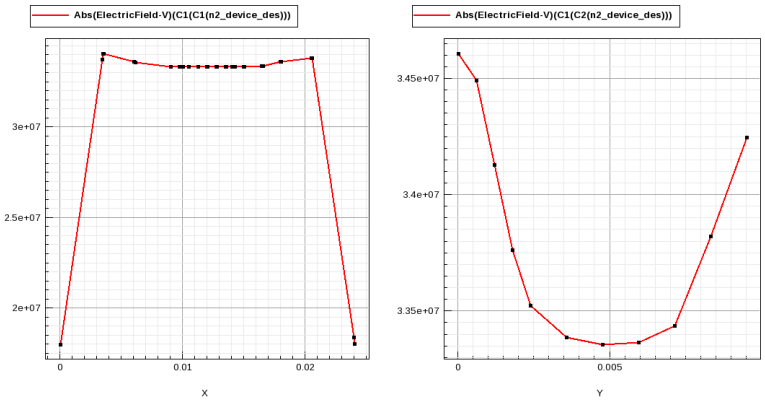
In the cut-Y setup, nothing new shows up - no sharp changes or odd shapes appear: everything stays even and balanced, while unusual spikes don't pop up. That suggests the results fit well with how the device is shaped and the set limits at edges, which also backs confidence in the grid detail and layer materials used during testing. For full coverage, the cut-Y outcomes show up not just in overall current density, but through electric field strength or electrostatic voltage too. Each of these shifts follows what we'd expect from a layered capacitor setup: fields stay almost steady in the middle area while steeper changes stick near electrode–dielectric borders, meanwhile currents hover around baseline levels once you're off the edge zones.

Electric Field

Here in fig 5.8 are shown the results for the Electric Field as investigated in 5.1.1.



(a) EF on CutY version with cut section XZ with y= 4.8nm to underline EF vector lines inside the trench



(b) Plot of EF for cut section XZ at Y=4.8nm and for cut section XY for Z=6nm

Figure 5.8: Overview of the EF on the Standard Cell CutY version

Total Current Density and Electrostatic Potential

Here in fig 5.9 are shown the results for the Total Current Density and Electrostatic Potential as investigated in 5.1.2 and 5.1.3.

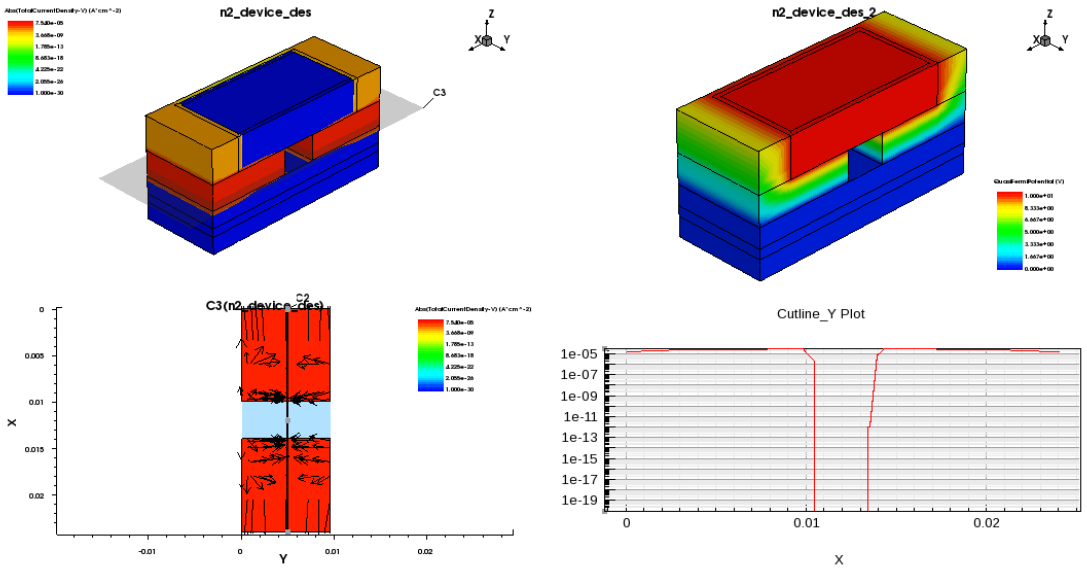


Figure 5.9: In the first plot it is shown the TCD, then the EP and the relative plot of the TCD to underline the behaviour in the trench

5.2 Array 3 Phase

The simulations carried out for the array structure are particularly interesting, as they represent the first attempt to investigate the device behaviour while introducing the clock concept, by assigning different voltage levels to the individual cells that compose the structure.

5.2.1 Electric Field

The electric field got modeled using three setups, matching the SDevice voltage approach along with the simplified timing steps from Chapter 3. Here are shown the resulting figures.

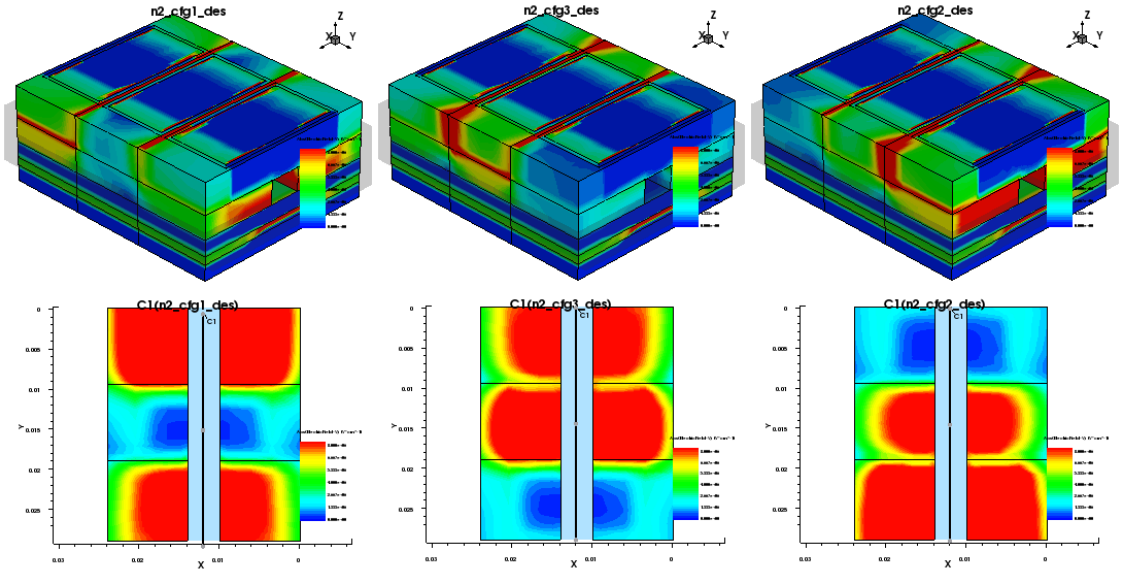


Figure 5.10: EF for each 3 configuration in Array 3 Phase from left to right: $HNL \rightarrow LHN \rightarrow NLH$

The graph in Fig.5.11 shows changes in the electric field inside the vacuum trench along the entire lenght of the trench. Every line displays the field behavior when the signal moves through one clock stage: red stands for HNL, while blue represents LHN, whereas green matches NLH.

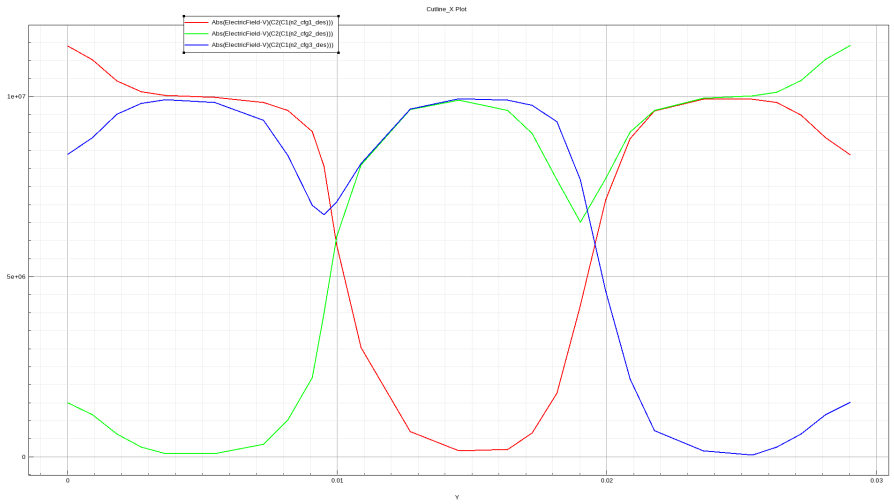
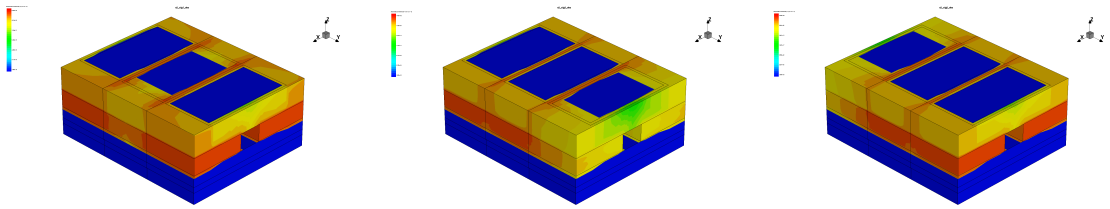


Figure 5.11: EF values across the vacuum trench for each of the 3 CLK configurations

5.2.2 Total Current Density

In the three-part bus shown in Fig.5.12, set up from left to right rispectivitly as H-N-L (fig.5.12a), L-H-N (fig.5.12b), N-L-H (fig.5.12c), the overall current density stays nearly negligible across the whole setup. While much of the layered material shows blue or green on the map, meaning levels

near the lowest detectable, the stronger values (in yellow or red) appear just near electrode borders or where insulation layers meet. in particular for fig.5.12a, the left "H" cell has a bit more current near the top edge, since voltage shifts there to start switching, though it quickly fades into the insulator. Meanwhile, the middle "N" cell stays pretty flat and barely conducts, which fits with holding data using steady, low power. On the right, the "L" cell shows mild concentration of current around the output groove and nearby sides, thanks to higher stress during release, but flow still sticks to tiny zones.

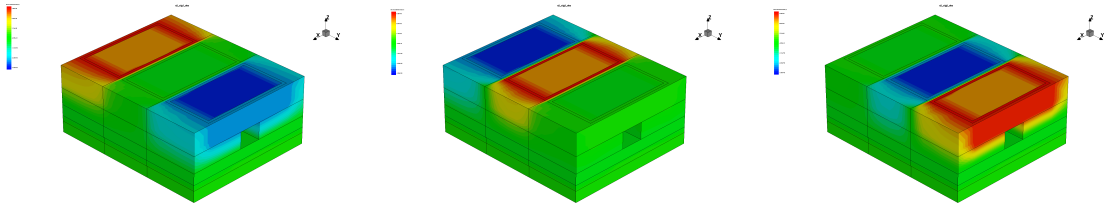


(a) TCD for HNL (b) TCD for LHN (c) TCD for NLH

Figure 5.12: Caption generale delle tre immagini affiancate.

5.2.3 EP

This graph 5.13 shows the EP voltage along a flat line through the three-cell bus setup, using clock settings cfg1 to cfg3 (see fig.5.13a, 5.13b, 5.13c). The layout runs side by side across the cells. Each configuration changes how the signal shifts. Voltage levels shift depending on timing inputs. Data here reflects steady-state behavior under different switching patterns.



(a) EP for HNL (b) EP for LHN (c) EP for NLH

Figure 5.13: EP for each 3 config for Array 3 cells

Every colored line, red, green, or blue, shows flat steps near $\mathbf{3}V$, sitting alongside a middle part close to 0 V. Sudden jumps happen right at vacuum trench spots and electrode borders, where voltage changes nearly straight up or down. The three curves sit slightly apart from each other, showing how each cell gets a unique H-N-L setup within every layout. In real measurements, the line cut shows the clock timing works right, every cell area hits the target voltage level, whereas sharp changes stay only in the spaces between cells, matching how QCA clocks should behave when using separate but steady electrode voltages.

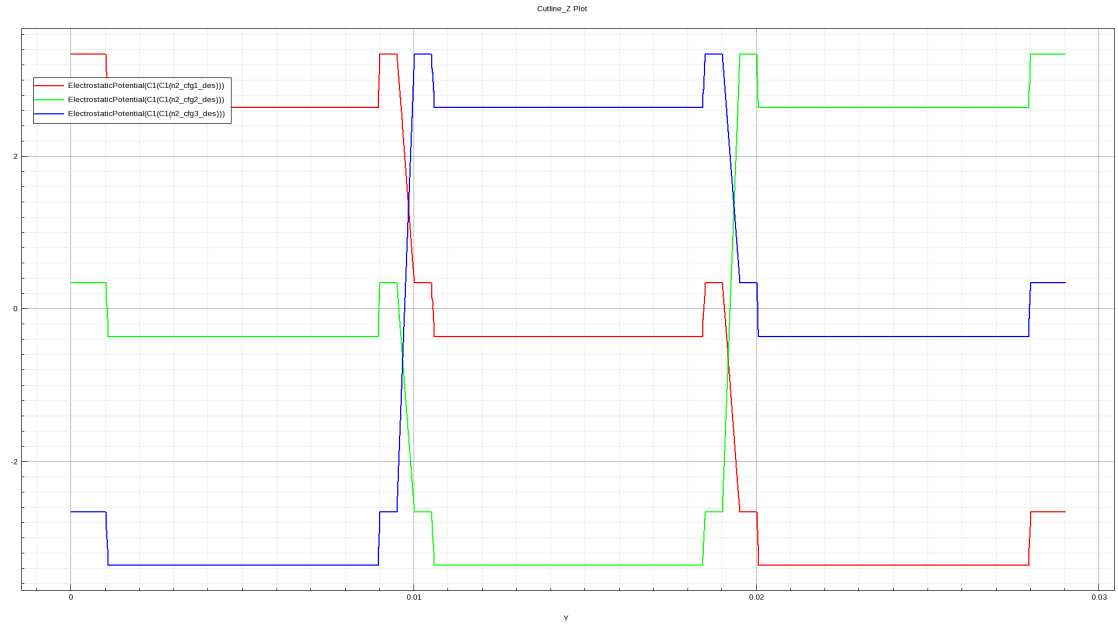


Figure 5.14: Plot of the EP for each config shows in 5.13.

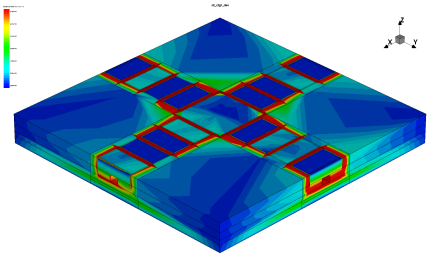
In particular here the red function is the config shows in fig.5.13a,the blue function is the config shows in fig.5.13b and green function is the config shows in fig.5.13c.

5.3 Majority Voter

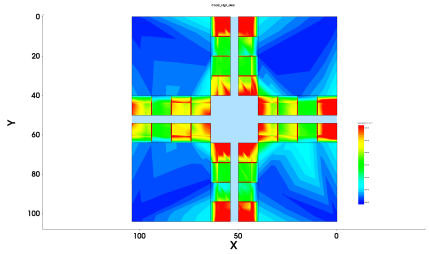
In this section MV is simulated. The simulations check how timing signals move across the four arms while electric interaction in the middle part performs a logic operation at the hardware stage. Specifically, attention goes to where voltage sits, which way fields point, and how current spreads through the layered structure, also whether polarity shifts properly from incoming paths to the outgoing one depending on varied signal setups and timing steps. Both setups investigated in previous chapter 3 were tested, one with a complete vacuum core, the other with a cross-style vacuum layout, using simulations. Differences that actually matter will be pointed out.

5.3.1 EF for both version

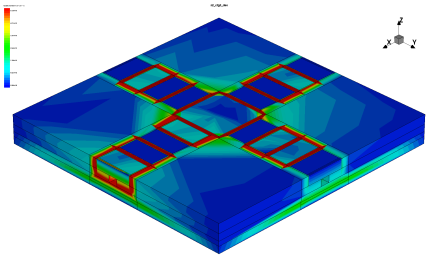
The next part shows how the electric field spreads across the main voter setup, using two shapes, one with a solid empty gap, another with a cross-like hole in the middle block. In every case, we look at the field patterns along the four branches and inside the core unit, checking how clock signals move through and how layout changes influence field focus and evenness. When needed, we side-by-side compare both versions to point out how the air cutout alters real-world clock performance.



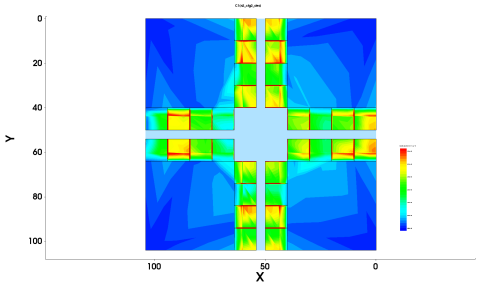
(a) EF in HNL config.



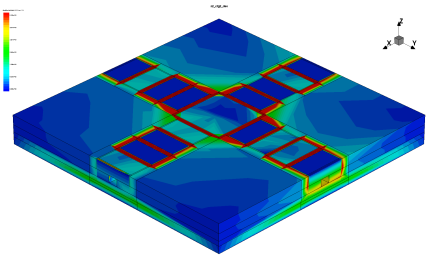
(b) EF in HNL config: cut plane z=6nm



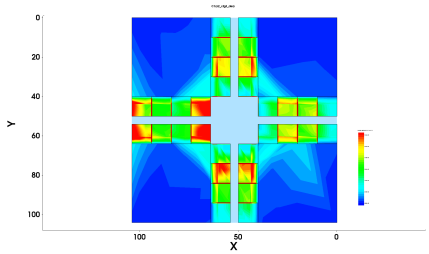
(c) EF in LHN config.



(d) EF in LHN config: cut plane z=6nm



(e) EF in NLH config.



(f) EF in NLH config: cut plane z=6nm

Figure 5.15: MV Vacuum EF.

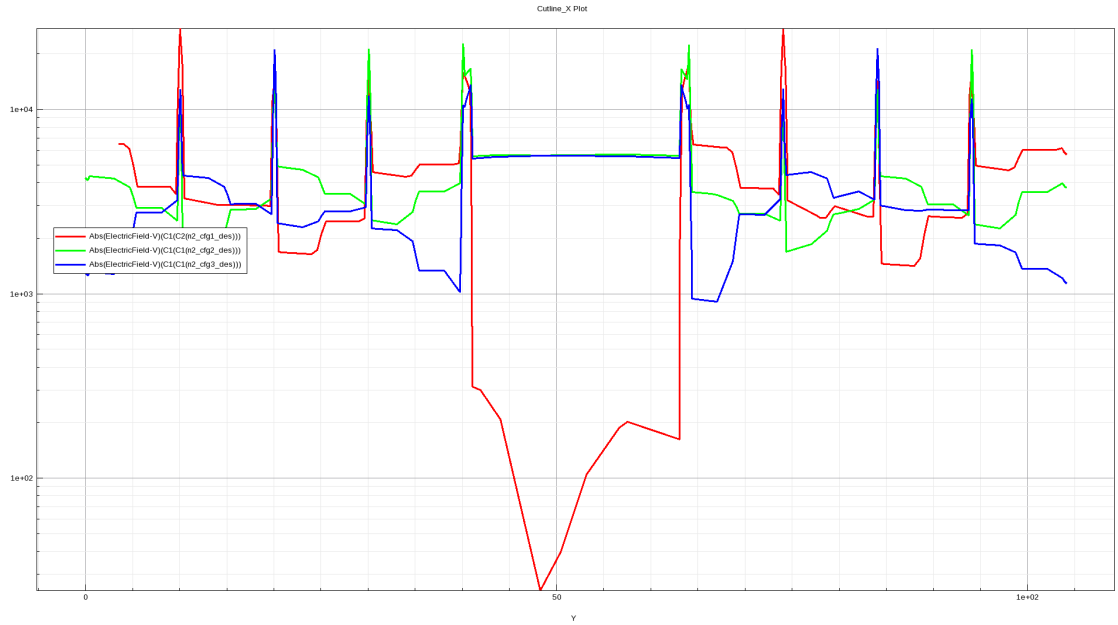
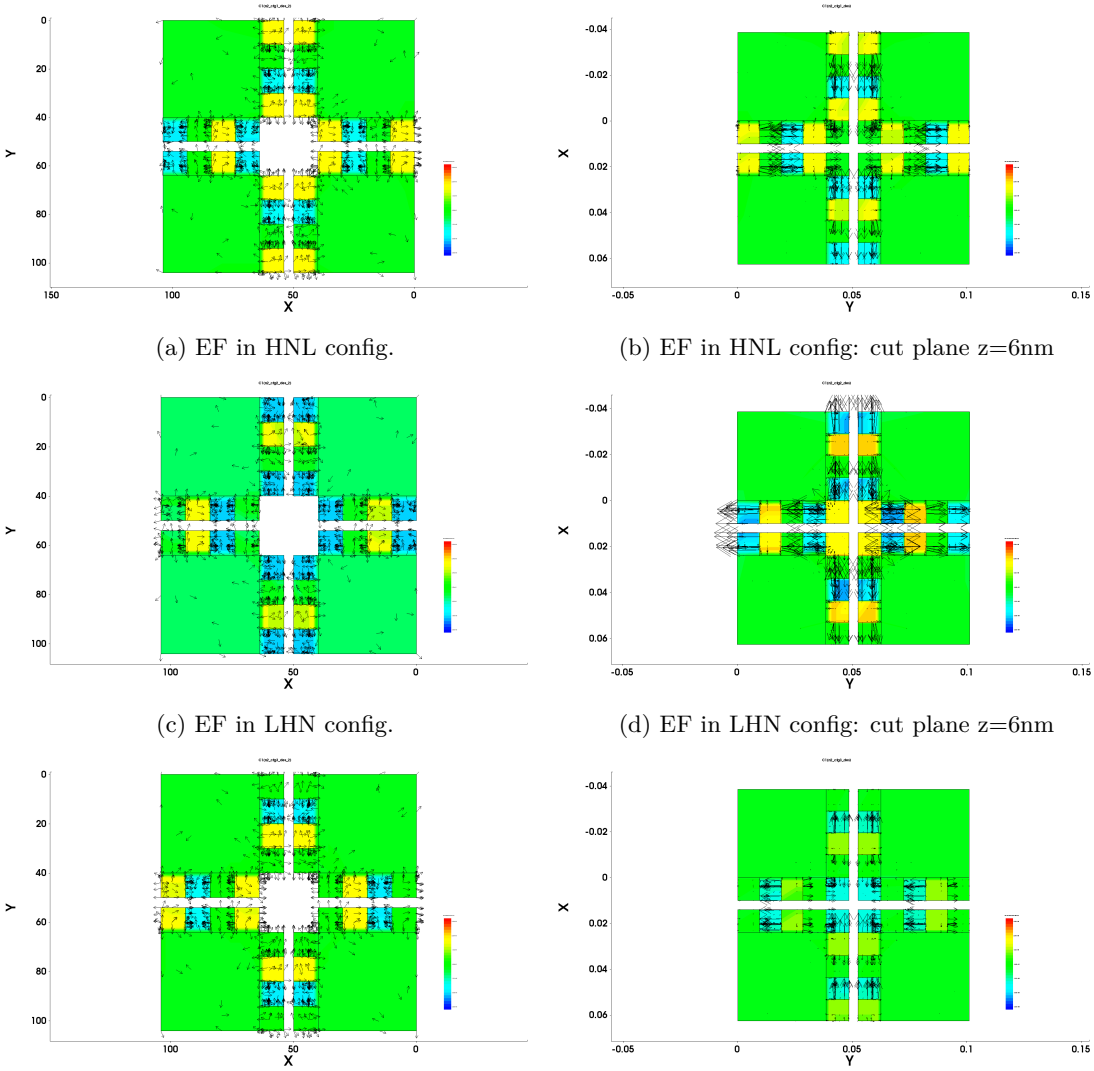


Figure 5.16: EF for all 3 config.

EF vector lines



(a) EF in HNL config.

(b) EF in HNL config: cut plane z=6nm

(c) EF in LHN config.

(d) EF in LHN config: cut plane z=6nm

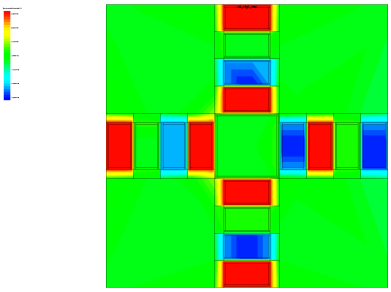
(e) EF in NLH config.

(f) EF in NLH config: cut plane z=6nm

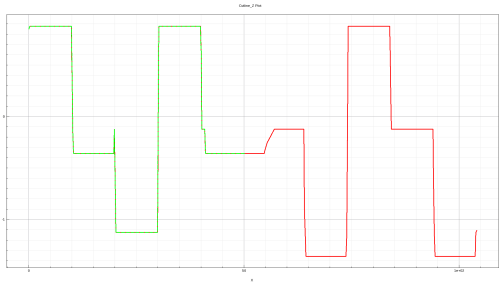
Figure 5.17: MV Vacuum EF.

in figure 5.17, the main change comes down to how the shape of the channel shifts the electric field, especially inside the middle cell. Actually, with cross-shaped vacuum versions, that shift happens more smoothly - easy to spot in the modeled results.

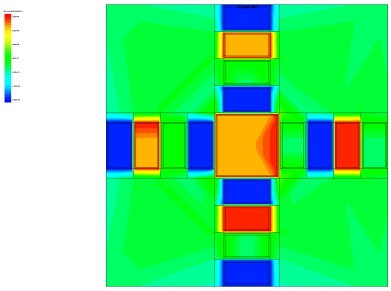
5.3.2 EP for both version



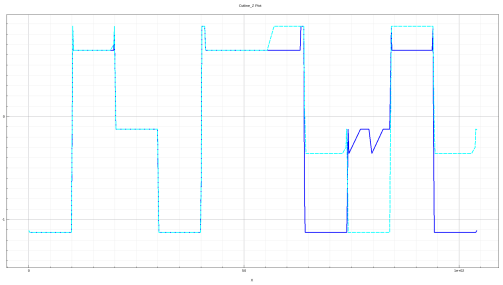
(a) EP in HNL config.



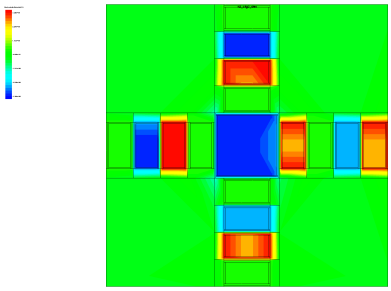
(b) EP in HNL config: green line is IN red line and OUT.



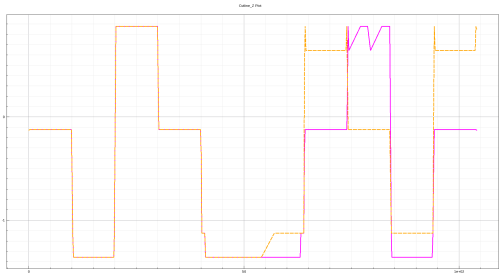
(c) EP in LHN config.



(d) EP in LHN config: light blue line is from IN to OUT



(e) EF in NLH config.

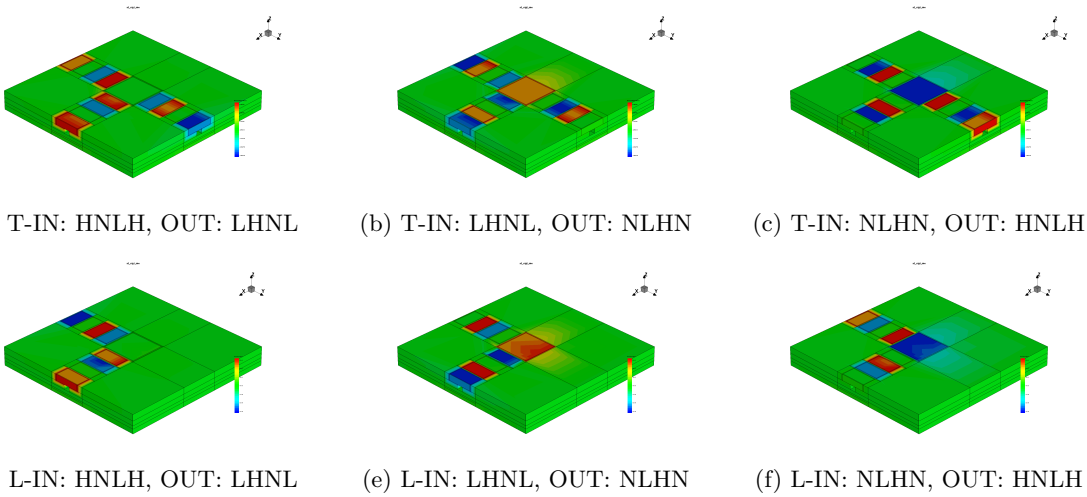


(f) EP in NLH config: light blue line is from IN to OUT

Figure 5.18: EP functions for each configuration: from input (North/West/South) to output (East).

5.3.3 L e T of both version

With T- and L-wire setups, things look pretty much alike: graphs for the majority voter match up closely to what we see in those shapes, showing that field behavior fits the design layout. Still, one key thing sets them apart is how voltage gets delivered across the arms of the T and L forms so signals can move right along, timed with the clock setup and matching each unit's role. Instead of uniform input, voltages must be set just right per section to control where data enters, exits, and flows properly. Check the next image - it shows exactly which voltage goes on every block in both T and L versions, laying out how power should tilt to keep everything running smooth.



(a) T-IN: HNLH, OUT: LHNL (b) T-IN: LHNL, OUT: NLHN (c) T-IN: NLHN, OUT: HNLH

(d) L-IN: HNLH, OUT: LHNL (e) L-IN: LHNL, OUT: NLHN (f) L-IN: NLHN, OUT: HNLH

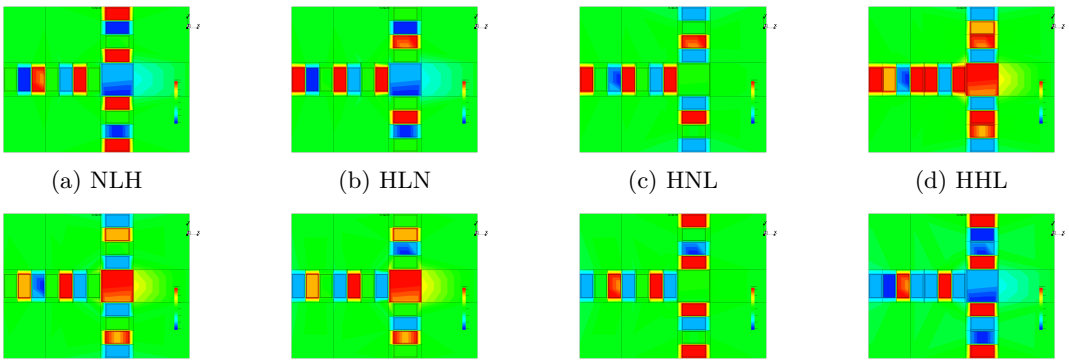
Figure 5.19: L e T wire EP values for each configuration

5.4 Circuit examples

The example circuits from Chapter 4 in tab. **??** got tested through simulations using SDevice setups made automatically by the tool explained there. Here, the voltage patterns were picked so they'd match nearly exactly how clocks actually work in real use, the kind listed in Table 4.3. Because of that, we looked at eight different clock setups, unlike before when simpler models ran on fewer variations. As seen in plots 5.18b, 5.18d and 5.18f, jumping fast between $+3V$ and $-3V$ under a basic 3-level method can create fake glitches and things that wouldn't happen in reality. On the flip side, breaking it into eight steps follows much closer to the gradual pulse shape in Figure 4.3, giving a truer picture of actual circuit performance. In the next part, just the EP findings are shared, because the electric field plus current patterns match what we've seen before in earlier setups.

5.4.1 Bus & T wire

In fig. 5.20 it is shown the simulation for each of 8 CLK phase configuration.



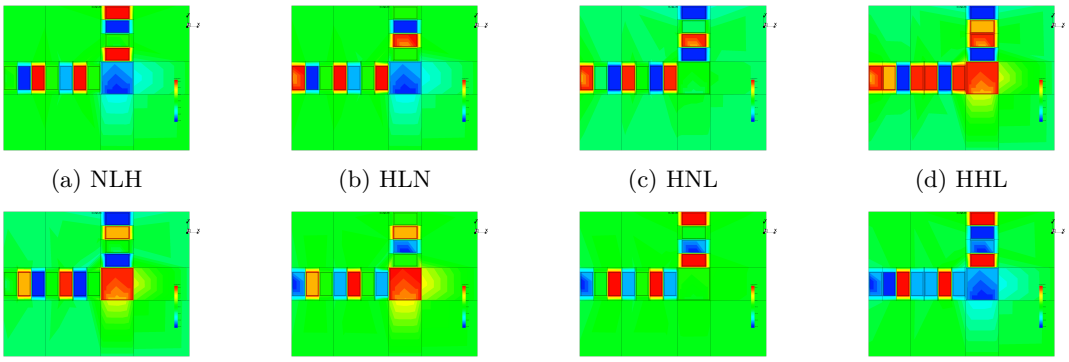
(a) NLH (b) HLN (c) HNL (d) HHL

(e) NHL (f) LHN (g) LNH (h) LLH

Figure 5.20: EP for a Bus + T wire

5.4.2 Bus & L wire

In Fig. 5.21 the electrostatic potential is shown for each of the 8 CLK–phase configurations.



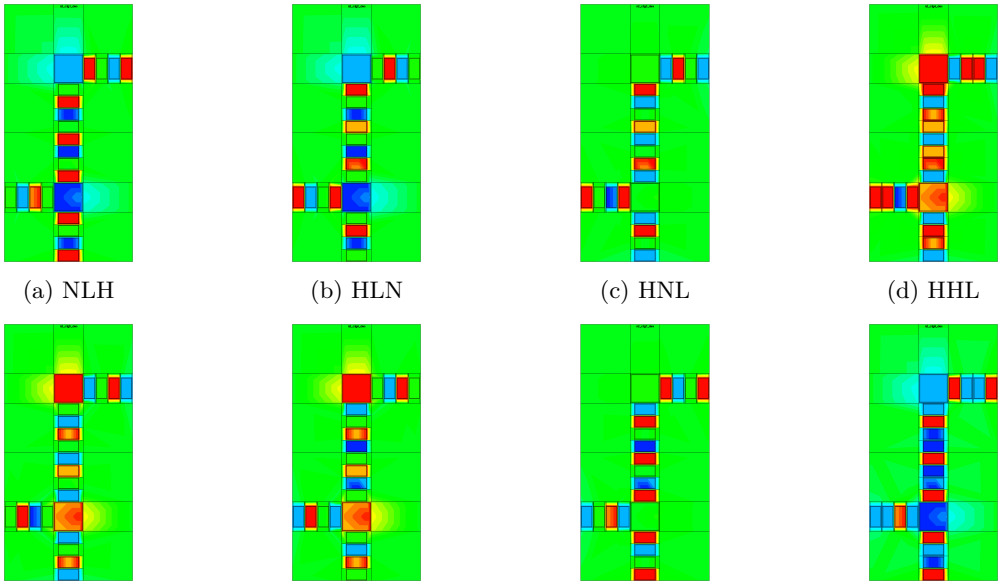
(a) NLH (b) HLN (c) HNL (d) HHL

(e) NHL (f) LHN (g) LNH (h) LLH

Figure 5.21: EP for a Bus + L wire.

5.4.3 L & T wire

In Fig. 5.22 the electrostatic potential is shown for each of the 8 CLK–phase configurations.



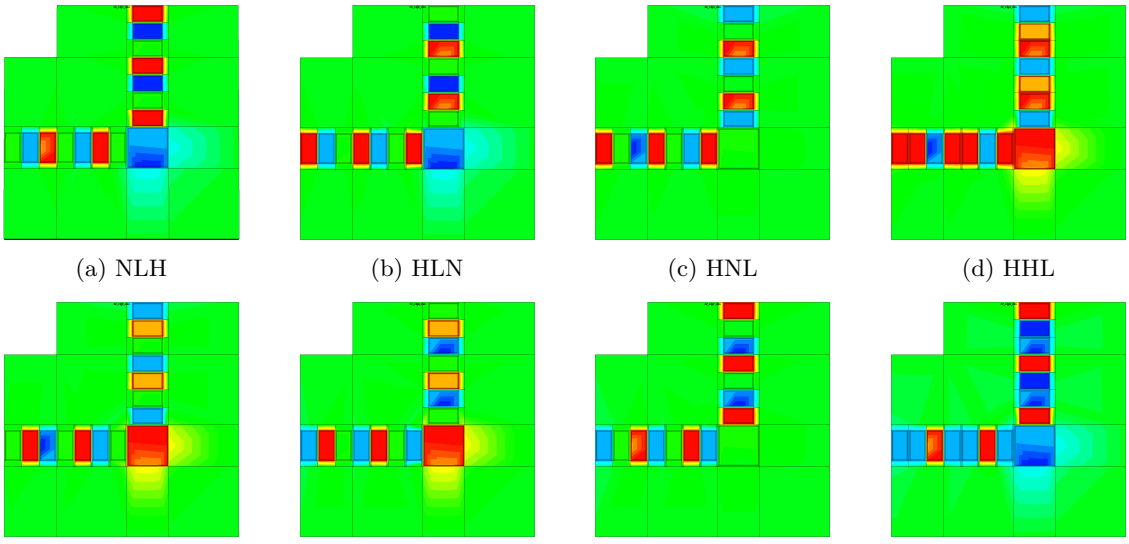
(a) NLH (b) HLN (c) HNL (d) HHL

(e) NHL (f) LHN (g) LNH (h) LLH

Figure 5.22: EP for an L + T wire.

5.4.4 Bus–L–Bus

In Fig. 5.23 the electrostatic potential is shown for each of the eight CLK–phase configurations.



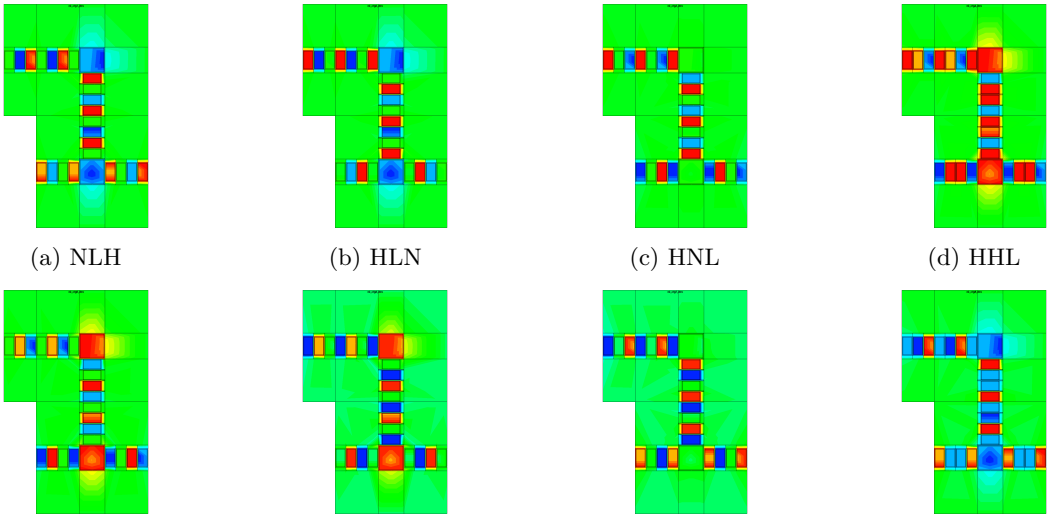
(a) NLH (b) HLN (c) HNL (d) HHL

(e) NHL (f) LHN (g) LNH (h) LLH

Figure 5.23: EP for a Bus–L–Bus structure.

5.4.5 Bus–L–T

In Fig.5.24 the electrostatic potential is shown for each of the eight CLK–phase configurations.



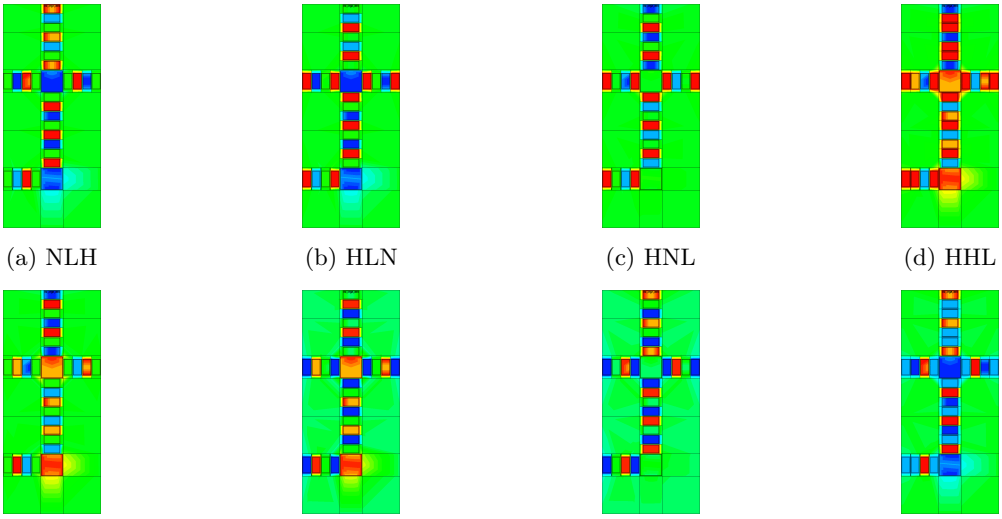
(a) NLH (b) HLN (c) HNL (d) HHL

(e) NHL (f) LHN (g) LNH (h) LLH

Figure 5.24: EP for a Bus–L–T structure.

5.4.6 L–MV–Bus

In Fig.5.25 the electrostatic potential is shown for each of the eight CLK–phase configurations.



(a) NLH (b) HLN (c) HNL (d) HHL

(e) NHL (f) LHN (g) LNH (h) LLH

Figure 5.25: EP for an L–MV–Bus structure.

Chapter 6

Conclusion and future perspectives

The number crunching done here paints a clear view of how the suggested MolFCN parts handle electric fields and move charge, starting from one basic unit all the way to tiny circuits. Taken together, the runs show the selected layered setup with electrodes can create the right field patterns needed to manage molecule alignment, yet still hold current leaks so low they hardly matter across every tested scenario. In the Standard Cell, electric field visuals plus cut-line graphs reveal that with a 0–10 V push, voltage shifts happen just where they should, between the upper and lower electrodes, across the working dielectric or empty gap. Around the middle of the groove, the field stays pretty steady, while bigger changes pop up near electrode corners or where materials meet. That pattern fits what's wanted for a basic plate-like setup, so molecules in the main zone feel a clean, predictable driving force. The overall current density bunching up close to boundaries, which shows the HfO_2 and air gaps block flow well, meaning most action comes from shifting fields, not real charge movement. Voltage views back this up: metal bits look like flat-energy zones, while non-conductors handle the full voltage shift as planned. The Cut-Y version acts just like the original. Even with its changed shape, the EF, TCD, and EP patterns show no odd peaks or sudden shifts, fields stay nearly even in the middle groove while currents drop near background noise levels outside edge zones. That means the way we set up the grid and materials for the base model holds up well under small design tweaks, which matters when arranging connections or adjusting spacing in actual circuits. As we shift to the three-cell setup, simulations start revealing how nearby units influence each other using a three-part timing signal. Electric field shapes highlight a step-by-step turn-on of H, then N, followed by L across the line, where peak intensity jumps from segment to segment matching the applied voltage order. Current density views stay weak and confined, suggesting that varying the clock in space doesn't break the stack's ability to block unwanted flow. Slices through the voltage landscape show distinct steps, flat areas near ($\pm$3 V) with sudden shifts at groove positions, confirming the layout holds the staggered voltages needed for guiding signals in one direction. The Majority Voter is tougher to handle because it needs four inputs plus a middle section working together smoothly. Instead of just one, two types were tested, one with an empty center, another with a cross-shaped gap, and both manage fields well enough for correct operation. Specifically, the cross-gap version gives cleaner, easier-to-manage field patterns near the center, helping guide the timing signal exactly where needed. Energy path graphs along input to output routes show the device passes on the strongest input state without messing up the clock's phase setup. Similar outcomes seen in T- and L-shaped wires prove this wiring method works fine around corners and splits, keeping electric performance like in straight runs. Lastly, circuits built automatically using the special layout software show

these traits work even in trickier setups. Instead of a rough three-part clock signal, picking an eight-step version helps simulation results skip sudden jumps (±3 V) , they now look like smooth steps. Whether testing Bus+T or L–MV–Bus - and every combo in between - the electric potential patterns stay true to basic building blocks, showing clear flat zones where devices operate and sharp but neat shifts at clock edges. That proves two things: the auto-conversion from circuit list to physical design works right, also the way clocks are set fits bigger MolFCN systems just fine. Fewer still, there's been zero focused heat modeling done. Every check happened at slow, steady conditions, just enough to test if the circuits worked right, while changes in heat across time got ignored completely. Because of that, looking into how devices warm up themselves plus related delays should guide what comes next.

In short, TCAD tests show the MolFCN cell idea works at the hardware level. The layer setup creates sharp, reliable electric fields with almost no leak current; when arranged in grids or splits, those traits stay stable even with timed signal shifts; built-in circuits run exactly like the theory predicts. These findings give a clear base for next steps, linking this TCAD method to tiny models of charge behavior, tuning shape and stuff to cut power needs, also pushing the auto-design system to handle bigger logic setups and match layouts with real-world test checks.

This study builds an initial clear link from tiny molecular devices to TCAD-driven layouts of MolFCN logic units and systems. A few paths open up ahead for follow-up exploration. A first step beyond current work involves physics-based and heat-aware models. Every simulation here stays steady-state, looking only at immediate electric response. Instead, tracking changes over time with combined electrical and thermal effects, like internal heating or real clock patterns, would help check durability, shifting power loss, and heat boundaries during extended use. Also, deeper atom-level details along with spread assessments,for instance, shape differences, substance traits, or placement errors, might show how well devices handle production flaws. So far, we've tested basic parts like cells, links, and voting setups; now it's time to build bigger pieces along with tiny working modules. A solid MolFCN inverter could be added next. In fact, a stable inversion step is key, since it needs to work with the suggested cell layout and timing method. That way, core logic parts like NAND or NOR gates become possible. On top of that, random combo circuits can then be built using just these tested base units. Doing this lets us watch how signals hold up, check how splits in wiring affect output, see where faults spread, plus test if the N-phase timing method still works well when pipelines get longer or connections split off in different directions. Lastly, the CAD process built here might grow into a broader design setup. Improvements could involve automatic conversion of logic descriptions into MolFCN setups, simple layout and wiring with timing limits, also pulling simplified behavior models from TCAD data for quicker circuit testing. Over time, these steps may help craft working prototypes, making the designs in this study potential picks for real-world MolFCN tests.

Appendix

Listing 1: SDE Code for Standard Cell

```
;; ============================================================
;; SENTARUS SDE - STANDARD CELL
;; SiO2 _ Dielettrico _ HfO2 _  Cornice Ti _ Contacts Au/Ti
;; ============================================================

(sde:clear)
;; ------------------------------
;; MAIN PARAMETER
;; ------------------------------
(sde:define-parameter "BottomDielectricWidth" 0.024000)
(sde:define-parameter "TrenchMiddleWidth" 0.004000)
(sde:define-parameter "TrenchWallWidth" (/ (- BottomDielectricWidth TrenchMiddleWidth) 2))
(sde:define-parameter "BottomDielectricThickness" 0.002000)
(sde:define-parameter "CellLength" 0.010000)
(sde:define-parameter "TrenchWallHeight" 0.003000)
(sde:define-parameter "AdhesionLayerThickness" 0.001000)
(sde:define-parameter "GoldLayerThickness" 0.002000)
(sde:define-parameter "ViaThickness" 0.001000)
(sde:define-parameter "PhasesNumber" 1)

;; ------------------------------
;; TOP GOLD + CORNER
;; ------------------------------
(sde:define-parameter "TopGoldWidthX" 0.016000)
(sde:define-parameter "TopGoldHeightY" 0.008000)
(sde:define-parameter "TopGoldThicknessZ" 0.004000)
(sde:define-parameter "TitaniumRimThickness" 0.000500)

;; ------------------------------
;; SECONDARY PARAMETER
;; ------------------------------
(sde:define-parameter "x_2ndWallStart" (+ TrenchWallWidth TrenchMiddleWidth))
(sde:define-parameter "z_BottomAdhesionLayerEnd" (+ BottomDielectricThickness
    AdhesionLayerThickness))
(sde:define-parameter "z_MiddleGoldEnd" (+ BottomDielectricThickness AdhesionLayerThickness
    GoldLayerThickness))
(sde:define-parameter "z_WallEnd" (+ z_MiddleGoldEnd TrenchWallHeight))
(sde:define-parameter "z_2ndGoldLayerEnd" (+ z_WallEnd TopGoldThicknessZ))
(sde:define-parameter "CenterX" (/ BottomDielectricWidth 2.0))
(sde:define-parameter "CenterY" (/ CellLength 2.0))

;; Bounding boxes
(sde:define-parameter "x_Au_min" (- CenterX (/ TopGoldWidthX 2.0)))
(sde:define-parameter "x_Au_max" (+ CenterX (/ TopGoldWidthX 2.0)))
(sde:define-parameter "y_Au_min" (- CenterY (/ TopGoldHeightY 2.0)))
(sde:define-parameter "y_Au_max" (+ CenterY (/ TopGoldHeightY 2.0)))
(sde:define-parameter "x_Rim_min" (- x_Au_min TitaniumRimThickness))
```

```
(sde:define-parameter "x_Rim_max" (+ x_Au_max TitaniumRimThickness))
(sde:define-parameter "y_Rim_min" (- y_Au_min TitaniumRimThickness))
(sde:define-parameter "y_Rim_max" (+ y_Au_max TitaniumRimThickness))

;; ============================================================
;; BOTTOM STRUCTURE
;; ============================================================
(sdegeo:create-cuboid (position 0 0 0)
 (position BottomDielectricWidth CellLength BottomDielectricThickness)
 "SiO2")

(sdegeo:create-cuboid (position 0 0 BottomDielectricThickness)
 (position BottomDielectricWidth CellLength z_BottomAdhesionLayerEnd)
 "Titanium")

(define BOTTOMCONTACT
 (sdegeo:create-cuboid (position 0 0 z_BottomAdhesionLayerEnd)
 (position BottomDielectricWidth CellLength z_MiddleGoldEnd)
 "Gold"))

;; ============================================================
;; CENTRAL STRUCTURE
;; ============================================================
(sdegeo:create-cuboid (position TrenchWallWidth 0 z_MiddleGoldEnd)
 (position x_2ndWallStart CellLength z_WallEnd)
 "Vacuum")

(define BODY_LEFT
 (sdegeo:create-cuboid (position 0 0 z_MiddleGoldEnd)
 (position TrenchWallWidth CellLength z_WallEnd)
 "HfO2"))

(define BODY_RIGHT
 (sdegeo:create-cuboid (position x_2ndWallStart 0 z_MiddleGoldEnd)
 (position BottomDielectricWidth CellLength z_WallEnd)
 "HfO2"))

;; ============================================================
;; TOP STRUCTURE
;; ============================================================
(define TOPCONTACT
 (sdegeo:create-cuboid (position x_Au_min y_Au_min z_WallEnd)
 (position x_Au_max y_Au_max z_2ndGoldLayerEnd)
 "Gold"))

;; Cornice Ti (4 lati)
(define RIM_LEFT (sdegeo:create-cuboid (position x_Rim_min y_Rim_min z_WallEnd)
 (position x_Au_min y_Rim_max z_2ndGoldLayerEnd)
 "Titanium"))
(define RIM_RIGHT (sdegeo:create-cuboid (position x_Au_max y_Rim_min z_WallEnd)
 (position x_Rim_max y_Rim_max z_2ndGoldLayerEnd)
 "Titanium"))
(define RIM_BOTTOM (sdegeo:create-cuboid (position x_Rim_min y_Rim_min z_WallEnd)
 (position x_Rim_max y_Au_min z_2ndGoldLayerEnd)
 "Titanium"))
(define RIM_TOP (sdegeo:create-cuboid (position x_Rim_min y_Au_max z_WallEnd)
 (position x_Rim_max y_Rim_max z_2ndGoldLayerEnd)
 "Titanium"))

(sdegeo:bool-unite (list RIM_LEFT RIM_RIGHT RIM_BOTTOM RIM_TOP))

;; ============================================================
;; TOP STRUCTURE - SiO2 AROUND Ti+Au
```

```
;; ============================================================
(define FILL_BOTTOM (sdegeo:create-cuboid (position 0 0 z_WallEnd)
 (position BottomDielectricWidth y_Rim_min z_2ndGoldLayerEnd)
 "SiO2"))
(define FILL_TOP (sdegeo:create-cuboid (position 0 y_Rim_max z_WallEnd)
 (position BottomDielectricWidth CellLength z_2ndGoldLayerEnd)
 "SiO2"))
(define FILL_LEFT (sdegeo:create-cuboid (position 0 y_Rim_min z_WallEnd)
 (position x_Rim_min y_Rim_max z_2ndGoldLayerEnd)
 "SiO2"))
(define FILL_RIGHT (sdegeo:create-cuboid (position x_Rim_max y_Rim_min z_WallEnd)
 (position BottomDielectricWidth y_Rim_max z_2ndGoldLayerEnd)
 "SiO2"))
(sdegeo:bool-unite (list FILL_BOTTOM FILL_TOP FILL_LEFT FILL_RIGHT))


;; ============================================================
;; CONTACTS
;; ============================================================

(sdegeo:define-contact-set "BottomContact" 4 (color:rgb 1 0 0) "##")
(sdegeo:set-current-contact-set "BottomContact")
(sdegeo:set-contact BOTTOMCONTACT "BottomContact")

(sdegeo:define-contact-set "TopContact" 4 (color:rgb 0 0 1) "##")
(sdegeo:set-current-contact-set "TopContact")
(sdegeo:set-contact TOPCONTACT "TopContact");;


;; ============================================================
;; MESH / REFINEMENT
;; ============================================================
(sdedr:define-refeval-window "InnerTrenchMesh" "Cuboid"
 (position TrenchWallWidth 0 z_BottomAdhesionLayerEnd)
 (position x_2ndWallStart CellLength z_2ndGoldLayerEnd))

(sdedr:define-refinement-size "TrenchRefinement" 0.001 0.005 0.001 0.001 0.005 0.001)
(sdedr:define-refinement-placement "RefinementPlacement_1" "TrenchRefinement"
 (list "window" "InnerTrenchMesh"))
(sdedr:define-refinement-function "TrenchRefinement" "MaxLenInt" "Gold" "Vacuum" 0.0001 "DoubleSide
     ")
(sdedr:define-refinement-function "TrenchRefinement" "MaxLenInt" "Gold" "Titanium" 0.0001)

;; ============================================================
;; BUILD + SAVE STRUCTURE
;; ============================================================
(sde:build-mesh "n@node@")
```

Listing 2: SDE Code for Standard Cell Cut-Y Version

```
;; ============================================================
;; SENTARUS SDE _ Standard cell (CUT Y version, flush_to_Ti)
;; SiO2 _ Dielectric   HfO2 _ Side bodies   Ti _ Rim   Au/Ti _ Contacts
;; ============================================================

(sde:clear)

;; ------------------------------
;; MAIN PARAMETERS
;; ------------------------------
(sde:define-parameter "BottomDielectricWidth"           0.024000)
(sde:define-parameter "TrenchMiddleWidth"       0.004000)
(sde:define-parameter "TrenchWallWidth"         (/ (- BottomDielectricWidth TrenchMiddleWidth) 2))
```

```
(sde:define-parameter "BottomDielectricThickness"  0.002000)
(sde:define-parameter "CellLength"              0.010000)
(sde:define-parameter "TrenchWallHeight"        0.003000)
(sde:define-parameter "AdhesionLayerThickness"  0.001000)
(sde:define-parameter "GoldLayerThickness"      0.002000)
(sde:define-parameter "ViaThickness"            0.001000)
(sde:define-parameter "PhasesNumber"            1)

;; --------------------------------
;; TOP GOLD + RIM
;; --------------------------------
(sde:define-parameter "TopGoldWidthX"           0.016000)
(sde:define-parameter "TopGoldHeightY"          0.008000)
(sde:define-parameter "TopGoldThicknessZ"       0.004000)
(sde:define-parameter "TitaniumRimThickness" 0.000500)

;; --------------------------------
;; SECONDARY PARAMETERS
;; --------------------------------
(sde:define-parameter "x_2ndWallStart"          (+ TrenchWallWidth TrenchMiddleWidth))
(sde:define-parameter "z_BottomAdhesionLayerEnd"  (+ BottomDielectricThickness
     AdhesionLayerThickness))
(sde:define-parameter "z_MiddleGoldEnd"         (+ BottomDielectricThickness AdhesionLayerThickness
      GoldLayerThickness))
(sde:define-parameter "z_WallEnd"               (+ z_MiddleGoldEnd TrenchWallHeight))
(sde:define-parameter "z_2ndGoldLayerEnd"       (+ z_WallEnd TopGoldThicknessZ))
(sde:define-parameter "CenterX"                 (/ BottomDielectricWidth 2.0))
(sde:define-parameter "CenterY"                 (/ CellLength 2.0))

;; Bounding boxes (top Au and Ti rim)
(sde:define-parameter "x_Au_min"  (- CenterX (/ TopGoldWidthX  2.0)))
(sde:define-parameter "x_Au_max"  (+ CenterX (/ TopGoldWidthX  2.0)))
(sde:define-parameter "y_Au_min"  (- CenterY (/ TopGoldHeightY 2.0)))
(sde:define-parameter "y_Au_max"  (+ CenterY (/ TopGoldHeightY 2.0)))
(sde:define-parameter "x_Rim_min" (- x_Au_min TitaniumRimThickness))
(sde:define-parameter "x_Rim_max" (+ x_Au_max TitaniumRimThickness))
(sde:define-parameter "y_Rim_min" (- y_Au_min TitaniumRimThickness))
(sde:define-parameter "y_Rim_max" (+ y_Au_max TitaniumRimThickness))

;; --------------------------------
;; ZX PLANE CUT DEFINITION
;; ALIGNED CUT WITH Ti RIM: YMaxCut = y_Rim_max
;; --------------------------------
(sde:define-parameter "YMaxCut" y_Rim_max)

;; ============================================================
;; LOWER STRUCTURE
;; ============================================================
(sdegeo:create-cuboid (position 0 0 0)
                      (position BottomDielectricWidth YMaxCut BottomDielectricThickness)
                      "SiO2")

(sdegeo:create-cuboid (position 0 0 BottomDielectricThickness)
                      (position BottomDielectricWidth YMaxCut z_BottomAdhesionLayerEnd)
                      "Titanium")

(define BOTTOMCONTACT
  (sdegeo:create-cuboid (position 0 0 z_BottomAdhesionLayerEnd)
                      (position BottomDielectricWidth YMaxCut z_MiddleGoldEnd)
                      "Gold"))

;; ============================================================
;; CENTRAL STRUCTURE
```

```
;; ============================================================
(sdegeo:create-cuboid (position TrenchWallWidth 0 z_MiddleGoldEnd)
                      (position x_2ndWallStart  YMaxCut z_WallEnd)
                      "Vacuum")

(define BODY_LEFT
  (sdegeo:create-cuboid (position 0 0 z_MiddleGoldEnd)
                      (position TrenchWallWidth YMaxCut z_WallEnd)
                      "HfO2"))

(define BODY_RIGHT
  (sdegeo:create-cuboid (position x_2ndWallStart 0 z_MiddleGoldEnd)
                      (position BottomDielectricWidth YMaxCut z_WallEnd)
                      "HfO2"))

;; ============================================================
;; TOP STRUCTURE
;; ============================================================
(define TOPCONTACT
  (sdegeo:create-cuboid (position x_Au_min y_Au_min z_WallEnd)
                      (position x_Au_max y_Au_max z_2ndGoldLayerEnd)
                      "Gold"))

;; Ti rim (4 sides)
(define RIM_LEFT
  (sdegeo:create-cuboid (position x_Rim_min y_Rim_min z_WallEnd)
                      (position x_Au_min  y_Rim_max z_2ndGoldLayerEnd)
                      "Titanium"))
(define RIM_RIGHT
  (sdegeo:create-cuboid (position x_Au_max  y_Rim_min z_WallEnd)
                      (position x_Rim_max y_Rim_max z_2ndGoldLayerEnd)
                      "Titanium"))
(define RIM_BOTTOM
  (sdegeo:create-cuboid (position x_Rim_min y_Rim_min z_WallEnd)
                      (position x_Rim_max y_Au_min  z_2ndGoldLayerEnd)
                      "Titanium"))
(define RIM_TOP
  (sdegeo:create-cuboid (position x_Rim_min y_Au_max  z_WallEnd)
                      (position x_Rim_max y_Rim_max z_2ndGoldLayerEnd)
                      "Titanium"))

(sdegeo:bool-unite (list RIM_LEFT RIM_RIGHT RIM_BOTTOM RIM_TOP))

;; ============================================================
;; TOP STRUCTURE _ SiO2 AROUND Ti+Au
;; ============================================================
(define FILL_BOTTOM
  (sdegeo:create-cuboid (position 0 0 z_WallEnd)
                      (position BottomDielectricWidth y_Rim_min z_2ndGoldLayerEnd)
                      "SiO2"))
(define FILL_LEFT
  (sdegeo:create-cuboid (position 0 y_Rim_min z_WallEnd)
                      (position x_Rim_min y_Rim_max z_2ndGoldLayerEnd)
                      "SiO2"))
(define FILL_RIGHT
  (sdegeo:create-cuboid (position x_Rim_max y_Rim_min z_WallEnd)
                      (position BottomDielectricWidth y_Rim_max z_2ndGoldLayerEnd)
                      "SiO2"))

(sdegeo:bool-unite (list FILL_BOTTOM FILL_LEFT FILL_RIGHT))

;; ============================================================
;; CONTACTS
```

```
;; ============================================================
(sdegeo:define-contact-set "BottomContact" 4 (color:rgb 1 0 0) "##")
(sdegeo:set-current-contact-set "BottomContact")
(sdegeo:set-contact BOTTOMCONTACT "BottomContact")

(sdegeo:define-contact-set "TopContact" 4 (color:rgb 0 0 1) "##")
(sdegeo:set-current-contact-set "TopContact")
(sdegeo:set-contact TOPCONTACT "TopContact")

;; ============================================================
;; MESH / REFINEMENT
;; ============================================================
(sdedr:define-refeval-window "InnerTrenchMesh" "Cuboid"
  (position TrenchWallWidth 0 z_BottomAdhesionLayerEnd)
  (position x_2ndWallStart  YMaxCut z_2ndGoldLayerEnd))

(sdedr:define-refinement-size "TrenchRefinement" 0.001 0.005 0.001 0.001 0.005 0.001)
(sdedr:define-refinement-placement "RefinementPlacement_1" "TrenchRefinement"
  (list "window" "InnerTrenchMesh"))
(sdedr:define-refinement-function "TrenchRefinement" "MaxLenInt" "Gold" "Vacuum" 0.0001 "DoubleSide
    ")
(sdedr:define-refinement-function "TrenchRefinement" "MaxLenInt" "Gold" "Titanium" 0.0001)

;; ============================================================
;; BUILD + SAVE STRUCTURE
;; ============================================================
(sde:build-mesh "n@node@")
```

Listing 3: SDevice Code for Standard Cell and Cut-Y Version

```
File {
 Grid = "n1_msh.tdr"
 Plot = "n@node@_device_des.tdr"
 Current = "n@node@_device_des.plt"
 Parameter = "sdevice.par"
}

Electrode {
 { Name = "BottomContact" Voltage = 0 }
 { Name = "TopContact" Voltage = 0 }
}

# ----------------------------
# PHYSICS
# ----------------------------
# MODELS FOR MATERIALS
Physics (Material="HfO2") {
 CondInsulator
}

Physics (Material="SiO2") {
 CondInsulator
}

# THERMIC CONTACT
Thermode {
 { Name = "BottomContact" Temperature = 300 SurfaceResistance = 1e-5 }
 { Name = "TopContact" Temperature = 300 SurfaceResistance = 1e-5 }
}

# ----------------------------
# PLOT
# ----------------------------
```

```
Plot {
 Potential
 ElectricField
 DielectricConstant
 Temperature

 # DIELECTRIC CURRENTS
 ConductionCurrentDensity/Vector
 DisplacementCurrentDensity/Vector

 # TOTAL CURRENT
 TotalCurrent/Vector

 # OTHERS
 SpaceCharge
 Potential Doping
 BandGap ElectronAffinity
 ConductionBandEnergy ValenceBandEnergy
}

Math {
 RelErrControl
 Extrapolate
}

# ----------------------------
# SOLVE
# ----------------------------
Solve
{
Coupled (Iterations= 100 LineSearchDamping= 1e-8) {Poisson}
Coupled{ Poisson Electron Hole Temperature Contact CondInsulator }
Plot(FilePrefix="n@node@_equilibrium")
quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
Goal {name= "TopContact" voltage = 10}
plot { range=(0, 1) intervals= 1}

){coupled { Poisson Electron Hole Temperature CondInsulator }}
}
```

Listing 4: SDE Code for Array 3-Phase

```
;; ============================================================
;; SENTARUS SDE  ARRAY Y with N=3 (cells 1-2 CUT, cell 3 STANDARD)
;; SiO2  Dielectric   HfO2  Side bodies   Ti  Rim   Au/Ti  Contacts
;; Units: micrometers ;; ============================================================

(sde:clear)

;; --------------------------------
;; MAIN PARAMETERS
;; --------------------------------
(sde:define-parameter "BottomDielectricWidth"      0.024000)
(sde:define-parameter "TrenchMiddleWidth"          0.004000)
(sde:define-parameter "TrenchWallWidth"            (/ (- BottomDielectricWidth TrenchMiddleWidth)
    2))
(sde:define-parameter "BottomDielectricThickness"  0.002000)
(sde:define-parameter "CellLength"                 0.010000)
(sde:define-parameter "TrenchWallHeight"           0.003000)
(sde:define-parameter "AdhesionLayerThickness"     0.001000)
(sde:define-parameter "GoldLayerThickness"         0.002000)
(sde:define-parameter "ViaThickness"               0.001000)
(sde:define-parameter "PhasesNumber"               1)
```

```
;; ------------------------------
;; TOP GOLD + RIM
;; ------------------------------
(sde:define-parameter "TopGoldWidthX"        0.016000)
(sde:define-parameter "TopGoldHeightY"       0.008000)
(sde:define-parameter "TopGoldThicknessZ"    0.004000)
(sde:define-parameter "TitaniumRimThickness" 0.000500)

;; ------------------------------
;; SECONDARY PARAMETERS
;; ------------------------------
(sde:define-parameter "x_2ndWallStart"            (+ TrenchWallWidth TrenchMiddleWidth))
(sde:define-parameter "z_BottomAdhesionLayerEnd"  (+ BottomDielectricThickness
    AdhesionLayerThickness))
(sde:define-parameter "z_MiddleGoldEnd"           (+ BottomDielectricThickness
    AdhesionLayerThickness GoldLayerThickness))
(sde:define-parameter "z_WallEnd"                 (+ z_MiddleGoldEnd TrenchWallHeight))
(sde:define-parameter "z_2ndGoldLayerEnd"         (+ z_WallEnd TopGoldThicknessZ))
(sde:define-parameter "CenterX"                   (/ BottomDielectricWidth 2.0))
(sde:define-parameter "CenterY"                   (/ CellLength 2.0))

;; Bounding boxes \(top Au and Ti rim\) _ local to the cell \(0..CellLength\)
(sde:define-parameter "x_Au_min"  (- CenterX (/ TopGoldWidthX  2.0)))
(sde:define-parameter "x_Au_max"  (+ CenterX (/ TopGoldWidthX  2.0)))
(sde:define-parameter "y_Au_min"  (- CenterY (/ TopGoldHeightY 2.0)))
(sde:define-parameter "y_Au_max"  (+ CenterY (/ TopGoldHeightY 2.0)))
(sde:define-parameter "x_Rim_min" (- x_Au_min TitaniumRimThickness))
(sde:define-parameter "x_Rim_max" (+ x_Au_max TitaniumRimThickness))
(sde:define-parameter "y_Rim_min" (- y_Au_min TitaniumRimThickness))
(sde:define-parameter "y_Rim_max" (+ y_Au_max TitaniumRimThickness))

;; ------------------------------
;; CELL TO CELL PITCH
;; ------------------------------
(define RimPackWidth (+ TopGoldHeightY (* 2.0 TitaniumRimThickness)))
(define Pitch (if (> CellLength RimPackWidth)
                  (min CellLength (* 0.5 (+ CellLength TopGoldHeightY (* 2.0 TitaniumRimThickness))
    ))
                  CellLength))

;; Offsets along Y for N=3
(define y0_1 0.0)
(define y0_2 (* 1.0 Pitch))
(define y0_3 (* 2.0 Pitch))

;; Helper functions for Y-shifted variables
(define (y-shift local y0) (+ local y0))

;; ============================================================
;; ================   CELL #1  CUT Y far   =================
;; ============================================================
(define YMaxCut_1 (y-shift y_Rim_max y0_1))        ;; cut flush with Ti rim
(define y_Au_min_1 (y-shift y_Au_min y0_1))
(define y_Au_max_1 (y-shift y_Au_max y0_1))
(define y_Rim_min_1 (y-shift y_Rim_min y0_1))
(define y_Rim_max_1 (y-shift y_Rim_max y0_1))

;; LOWER STRUCTURE
(sdegeo:create-cuboid (position 0        y0_1    0)
                      (position BottomDielectricWidth YMaxCut_1 BottomDielectricThickness)
                      "SiO2")
```

```
(sdegeo:create-cuboid (position 0        y0_1    BottomDielectricThickness)
                      (position BottomDielectricWidth YMaxCut_1 z_BottomAdhesionLayerEnd)
                      "Titanium")

(define BOTTOMCONTACT_1
  (sdegeo:create-cuboid (position 0        y0_1    z_BottomAdhesionLayerEnd)
                        (position BottomDielectricWidth YMaxCut_1 z_MiddleGoldEnd)
                        "Gold"))

;; MIDDLE STRUCTURE (Trench)
(sdegeo:create-cuboid (position TrenchWallWidth y0_1 z_MiddleGoldEnd)
                      (position x_2ndWallStart  YMaxCut_1 z_WallEnd)
                      "Vacuum")

(define BODY_LEFT_1
  (sdegeo:create-cuboid (position 0               y0_1    z_MiddleGoldEnd)
                        (position TrenchWallWidth YMaxCut_1 z_WallEnd)
                        "HfO2"))

(define BODY_RIGHT_1
  (sdegeo:create-cuboid (position x_2ndWallStart  y0_1    z_MiddleGoldEnd)
                        (position BottomDielectricWidth YMaxCut_1 z_WallEnd)
                        "HfO2"))

;; UPPER LAYER (Top Gold + Ti Rim)
(define TOPCONTACT_1
  (sdegeo:create-cuboid (position x_Au_min y_Au_min_1 z_WallEnd)
                        (position x_Au_max y_Au_max_1 z_2ndGoldLayerEnd)
                        "Gold"))

(define RIM_LEFT_1
  (sdegeo:create-cuboid (position x_Rim_min y_Rim_min_1 z_WallEnd)
                        (position x_Au_min  y_Rim_max_1 z_2ndGoldLayerEnd)
                        "Titanium"))
(define RIM_RIGHT_1
  (sdegeo:create-cuboid (position x_Au_max  y_Rim_min_1 z_WallEnd)
                        (position x_Rim_max y_Rim_max_1 z_2ndGoldLayerEnd)
                        "Titanium"))
(define RIM_BOTTOM_1
  (sdegeo:create-cuboid (position x_Rim_min y_Rim_min_1 z_WallEnd)
                        (position x_Rim_max y_Au_min_1  z_2ndGoldLayerEnd)
                        "Titanium"))
(define RIM_TOP_1
  (sdegeo:create-cuboid (position x_Rim_min y_Au_max_1  z_WallEnd)
                        (position x_Rim_max y_Rim_max_1 z_2ndGoldLayerEnd)
                        "Titanium"))

(sdegeo:bool-unite (list RIM_LEFT_1 RIM_RIGHT_1 RIM_BOTTOM_1 RIM_TOP_1))

;; SiO2 FILL around the rim (no FILL_TOP if zero thickness)
(define FILL_BOTTOM_1
  (sdegeo:create-cuboid (position 0                 y0_1        z_WallEnd)
                        (position BottomDielectricWidth y_Rim_min_1 z_2ndGoldLayerEnd)
                        "SiO2"))
(define FILL_LEFT_1
  (sdegeo:create-cuboid (position 0         y_Rim_min_1 z_WallEnd)
                        (position x_Rim_min y_Rim_max_1 z_2ndGoldLayerEnd)
                        "SiO2"))
(define FILL_RIGHT_1
  (sdegeo:create-cuboid (position x_Rim_max y_Rim_min_1 z_WallEnd)
                        (position BottomDielectricWidth y_Rim_max_1 z_2ndGoldLayerEnd)
                        "SiO2"))
```

```
(sdegeo:bool-unite (list FILL_BOTTOM_1 FILL_LEFT_1 FILL_RIGHT_1))

;; ============================================================
;; ================   CELL 2  CUT Y far   =================
;; ============================================================
(define YMaxCut_2 (y-shift y_Rim_max y0_2))        ;; cut flush with Ti rim
(define y_Au_min_2 (y-shift y_Au_min y0_2))
(define y_Au_max_2 (y-shift y_Au_max y0_2))
(define y_Rim_min_2 (y-shift y_Rim_min y0_2))
(define y_Rim_max_2 (y-shift y_Rim_max y0_2))

;; LOWER STRUCTURE
(sdegeo:create-cuboid (position 0        y0_2    0)
                      (position BottomDielectricWidth YMaxCut_2 BottomDielectricThickness)
                      "SiO2")

(sdegeo:create-cuboid (position 0        y0_2    BottomDielectricThickness)
                      (position BottomDielectricWidth YMaxCut_2 z_BottomAdhesionLayerEnd)
                      "Titanium")

(define BOTTOMCONTACT_2
  (sdegeo:create-cuboid (position 0        y0_2    z_BottomAdhesionLayerEnd)
                        (position BottomDielectricWidth YMaxCut_2 z_MiddleGoldEnd)
                        "Gold"))

;; MIDDLE STRUCTURE (Trench)
(sdegeo:create-cuboid (position TrenchWallWidth y0_2 z_MiddleGoldEnd)
                      (position x_2ndWallStart  YMaxCut_2 z_WallEnd)
                      "Vacuum")

(define BODY_LEFT_2
  (sdegeo:create-cuboid (position 0               y0_2    z_MiddleGoldEnd)
                        (position TrenchWallWidth YMaxCut_2 z_WallEnd)
                        "HfO2"))

(define BODY_RIGHT_2
  (sdegeo:create-cuboid (position x_2ndWallStart  y0_2    z_MiddleGoldEnd)
                        (position BottomDielectricWidth YMaxCut_2 z_WallEnd)
                        "HfO2"))

;; UPPER LAYER (Top Gold + Ti Rim)
(define TOPCONTACT_2
  (sdegeo:create-cuboid (position x_Au_min y_Au_min_2 z_WallEnd)
                        (position x_Au_max y_Au_max_2 z_2ndGoldLayerEnd)
                        "Gold"))

(define RIM_LEFT_2
  (sdegeo:create-cuboid (position x_Rim_min y_Rim_min_2 z_WallEnd)
                        (position x_Au_min  y_Rim_max_2 z_2ndGoldLayerEnd)
                        "Titanium"))
(define RIM_RIGHT_2
  (sdegeo:create-cuboid (position x_Au_max  y_Rim_min_2 z_WallEnd)
                        (position x_Rim_max y_Rim_max_2 z_2ndGoldLayerEnd)
                        "Titanium"))
(define RIM_BOTTOM_2
  (sdegeo:create-cuboid (position x_Rim_min y_Rim_min_2 z_WallEnd)
                        (position x_Rim_max y_Au_min_2  z_2ndGoldLayerEnd)
                        "Titanium"))
(define RIM_TOP_2
  (sdegeo:create-cuboid (position x_Rim_min y_Au_max_2  z_WallEnd)
                        (position x_Rim_max y_Rim_max_2 z_2ndGoldLayerEnd)
                        "Titanium"))
```

```
(sdegeo:bool-unite (list RIM_LEFT_2 RIM_RIGHT_2 RIM_BOTTOM_2 RIM_TOP_2))

;; SiO2 FILL around the rim (no FILL_TOP if zero thickness)
(define FILL_BOTTOM_2
  (sdegeo:create-cuboid (position 0                 y0_2        z_WallEnd)
                        (position BottomDielectricWidth y_Rim_min_2 z_2ndGoldLayerEnd)
                        "SiO2"))
(define FILL_LEFT_2
  (sdegeo:create-cuboid (position 0         y_Rim_min_2 z_WallEnd)
                        (position x_Rim_min y_Rim_max_2 z_2ndGoldLayerEnd)
                        "SiO2"))
(define FILL_RIGHT_2
  (sdegeo:create-cuboid (position x_Rim_max y_Rim_min_2 z_WallEnd)
                        (position BottomDielectricWidth y_Rim_max_2 z_2ndGoldLayerEnd)
                        "SiO2"))

(sdegeo:bool-unite (list FILL_BOTTOM_2 FILL_LEFT_2 FILL_RIGHT_2))

;; ============================================================
;; ================   CELL 3  STANDARD   =================
;; ============================================================
(define YMaxCut_3 (+ y0_3 CellLength))     ;; no cut
(define y_Au_min_3 (y-shift y_Au_min y0_3))
(define y_Au_max_3 (y-shift y_Au_max y0_3))
(define y_Rim_min_3 (y-shift y_Rim_min y0_3))
(define y_Rim_max_3 (y-shift y_Rim_max y0_3))

;; LOWER STRUCTURE
(sdegeo:create-cuboid (position 0        y0_3    0)
                      (position BottomDielectricWidth YMaxCut_3 BottomDielectricThickness)
                      "SiO2")

(sdegeo:create-cuboid (position 0        y0_3    BottomDielectricThickness)
                      (position BottomDielectricWidth YMaxCut_3 z_BottomAdhesionLayerEnd)
                      "Titanium")

(define BOTTOMCONTACT_3
  (sdegeo:create-cuboid (position 0        y0_3    z_BottomAdhesionLayerEnd)
                        (position BottomDielectricWidth YMaxCut_3 z_MiddleGoldEnd)
                        "Gold"))

;; MIDDLE STRUCTURE (Trench)
(sdegeo:create-cuboid (position TrenchWallWidth y0_3 z_MiddleGoldEnd)
                      (position x_2ndWallStart  YMaxCut_3 z_WallEnd)
                      "Vacuum")

(define BODY_LEFT_3
  (sdegeo:create-cuboid (position 0               y0_3    z_MiddleGoldEnd)
                        (position TrenchWallWidth YMaxCut_3 z_WallEnd)
                        "HfO2"))

(define BODY_RIGHT_3
  (sdegeo:create-cuboid (position x_2ndWallStart  y0_3    z_MiddleGoldEnd)
                        (position BottomDielectricWidth YMaxCut_3 z_WallEnd)
                        "HfO2"))

;; UPPER LAYER (Top Gold + Ti Rim)
(define TOPCONTACT_3
  (sdegeo:create-cuboid (position x_Au_min y_Au_min_3 z_WallEnd)
                        (position x_Au_max y_Au_max_3 z_2ndGoldLayerEnd)
                        "Gold"))

(define RIM_LEFT_3
```

```
  (sdegeo:create-cuboid (position x_Rim_min y_Rim_min_3 z_WallEnd)
                        (position x_Au_min  y_Rim_max_3 z_2ndGoldLayerEnd)
                        "Titanium"))
(define RIM_RIGHT_3
  (sdegeo:create-cuboid (position x_Au_max  y_Rim_min_3 z_WallEnd)
                        (position x_Rim_max y_Rim_max_3 z_2ndGoldLayerEnd)
                        "Titanium"))
(define RIM_BOTTOM_3
  (sdegeo:create-cuboid (position x_Rim_min y_Rim_min_3 z_WallEnd)
                        (position x_Rim_max y_Au_min_3  z_2ndGoldLayerEnd)
                        "Titanium"))
(define RIM_TOP_3
  (sdegeo:create-cuboid (position x_Rim_min y_Au_max_3  z_WallEnd)
                        (position x_Rim_max y_Rim_max_3 z_2ndGoldLayerEnd)
                        "Titanium"))

(sdegeo:bool-unite (list RIM_LEFT_3 RIM_RIGHT_3 RIM_BOTTOM_3 RIM_TOP_3))

;; SiO2 FILL around the rim (STANDARD with TOP)
(define FILL_BOTTOM_3
  (sdegeo:create-cuboid (position 0                 y0_3        z_WallEnd)
                        (position BottomDielectricWidth y_Rim_min_3 z_2ndGoldLayerEnd)
                        "SiO2"))
(define FILL_LEFT_3
  (sdegeo:create-cuboid (position 0         y_Rim_min_3 z_WallEnd)
                        (position x_Rim_min y_Rim_max_3 z_2ndGoldLayerEnd)
                        "SiO2"))
(define FILL_RIGHT_3
  (sdegeo:create-cuboid (position x_Rim_max y_Rim_min_3 z_WallEnd)
                        (position BottomDielectricWidth y_Rim_max_3 z_2ndGoldLayerEnd)
                        "SiO2"))
(define FILL_TOP_3
  (sdegeo:create-cuboid (position 0                 y_Rim_max_3 z_WallEnd)
                        (position BottomDielectricWidth YMaxCut_3 z_2ndGoldLayerEnd)
                        "SiO2"))

(sdegeo:bool-unite (list FILL_BOTTOM_3 FILL_LEFT_3 FILL_RIGHT_3 FILL_TOP_3))

;; ============================================================
;; CONTACTS one set for each top contact
;; ============================================================
(sdegeo:define-contact-set "BottomContact" 4 (color:rgb 1 0 0) "##")
(sdegeo:set-current-contact-set "BottomContact")
(sdegeo:set-contact BOTTOMCONTACT_1 "BottomContact")
(sdegeo:set-contact BOTTOMCONTACT_2 "BottomContact")
(sdegeo:set-contact BOTTOMCONTACT_3 "BottomContact")
(sdegeo:define-contact-set "TopContact_1" 4 (color:rgb 0 0 1) "##")
(sdegeo:set-current-contact-set "TopContact_1")
(sdegeo:set-contact TOPCONTACT_1 "TopContact_1")
(sdegeo:define-contact-set "TopContact_2" 4 (color:rgb 0 0 1) "##")
(sdegeo:set-current-contact-set "TopContact_2")
(sdegeo:set-contact TOPCONTACT_2 "TopContact_2")
(sdegeo:define-contact-set "TopContact_3" 4 (color:rgb 0 0 1) "##")
(sdegeo:set-current-contact-set "TopContact_3")
(sdegeo:set-contact TOPCONTACT_3 "TopContact_3")

;; ============================================================
;; MESH / REFINEMENT each cell within its window
;; ============================================================
(sdedr:define-refeval-window "InnerTrenchMesh_1" "Cuboid"
  (position TrenchWallWidth y0_1 z_BottomAdhesionLayerEnd)
  (position x_2ndWallStart  YMaxCut_1 z_2ndGoldLayerEnd))
```

```
(sdedr:define-refeval-window "InnerTrenchMesh_2" "Cuboid"
  (position TrenchWallWidth y0_2 z_BottomAdhesionLayerEnd)
  (position x_2ndWallStart  YMaxCut_2 z_2ndGoldLayerEnd))

(sdedr:define-refeval-window "InnerTrenchMesh_3" "Cuboid"
  (position TrenchWallWidth y0_3 z_BottomAdhesionLayerEnd)
  (position x_2ndWallStart  YMaxCut_3 z_2ndGoldLayerEnd))

(sdedr:define-refinement-size "TrenchRefinement" 0.001 0.005 0.001 0.001 0.005 0.001)
(sdedr:define-refinement-placement "Refine_1" "TrenchRefinement" (list "window" "InnerTrenchMesh_1"
    ))
(sdedr:define-refinement-placement "Refine_2" "TrenchRefinement" (list "window" "InnerTrenchMesh_2"
    ))
(sdedr:define-refinement-placement "Refine_3" "TrenchRefinement" (list "window" "InnerTrenchMesh_3"
    ))

(sdedr:define-refinement-function "TrenchRefinement" "MaxLenInt" "Gold"    "Vacuum"   0.0001 "
    DoubleSide")
(sdedr:define-refinement-function "TrenchRefinement" "MaxLenInt" "Gold"    "Titanium" 0.0001)

;; ============================================================
;; BUILD + SAVE STRUCTURE
;; ============================================================
(sde:build-mesh "n@node@")
```

Listing 5: SDE Code for MV Vacuum

```
;; ======================================================================
;; SENTARUS SDE _ Majority_Voter Cross \(N=4\) + Per_cell Named Contacts
;; Units: micrometers
;; Top contacts defined:
;;   south_1..4, north_1..4, west_1..4, east_1..4, and center X.
;; Index: 1 = far end, 4 = near the center.
;; ======================================================================

(sde:clear)

; ------------------------------
; MAIN PARAMETERS
; ------------------------------
(sde:define-parameter "BottomDielectricWidth"   0.024000)
(sde:define-parameter "CellLength"                      0.010000)
(sde:define-parameter "BottomDielectricThickness"       0.002000)
(sde:define-parameter "AdhesionLayerThickness"          0.001000)
(sde:define-parameter "GoldLayerThickness"              0.002000)
(sde:define-parameter "TrenchMiddleWidth"               0.004000)
(sde:define-parameter "TrenchWallHeight"                0.003000)
(sde:define-parameter "TopGoldWidthX"                   0.016000)
(sde:define-parameter "TopGoldHeightY"          0.008000)
(sde:define-parameter "TopGoldThicknessZ"               0.004000)
(sde:define-parameter "TitaniumRimThickness"            0.000500)
(sde:define-parameter "CenterCapSide"                   0.022000)
(sde:define-parameter "x0"                              0.000000)
(sde:define-parameter "y0"                              0.000000)
(sde:define-parameter "z0"                              0.000000)
(define Ncells 4)
(sde:define-parameter "BlockMaterial"           "HfO2")

; ------------------------------
; DERIVED & CONSTANTS
; ------------------------------
(sde:define-parameter "TrenchWallWidth"
  (/ (- BottomDielectricWidth TrenchMiddleWidth) 2.0))
```

```
(define H_tot (+ BottomDielectricThickness AdhesionLayerThickness GoldLayerThickness
    TrenchWallHeight TopGoldThicknessZ))
(define RimPackWidth (+ TopGoldHeightY (* 2.0 TitaniumRimThickness)))
(define Pitch (if (> CellLength RimPackWidth)
               (min CellLength (* 0.5 (+ CellLength TopGoldHeightY (* 2.0 TitaniumRimThickness))))
               CellLength))

; --------------------------------
; HELPERS
; --------------------------------
(define (mk-cuboid name mat x y z dx dy dz)
  (sdegeo:create-cuboid (position x y z)
                        (position (+ x dx) (+ y dy) (+ z dz))
                        mat))

; --------------------------------
; CONTACT SETS
; --------------------------------
(sdegeo:define-contact-set "BottomContact" 4 (color:rgb 1 0 0) "##")

(sdegeo:define-contact-set "TopContact_south_1" 4 (color:rgb 0 0 1) "##")
(sdegeo:define-contact-set "TopContact_south_2" 4 (color:rgb 0 0 1) "##")
(sdegeo:define-contact-set "TopContact_south_3" 4 (color:rgb 0 0 1) "##")
(sdegeo:define-contact-set "TopContact_south_4" 4 (color:rgb 0 0 1) "##")

(sdegeo:define-contact-set "TopContact_north_1" 4 (color:rgb 0 0 1) "##")
(sdegeo:define-contact-set "TopContact_north_2" 4 (color:rgb 0 0 1) "##")
(sdegeo:define-contact-set "TopContact_north_3" 4 (color:rgb 0 0 1) "##")
(sdegeo:define-contact-set "TopContact_north_4" 4 (color:rgb 0 0 1) "##")

(sdegeo:define-contact-set "TopContact_west_1" 4 (color:rgb 0 0 1) "##")
(sdegeo:define-contact-set "TopContact_west_2" 4 (color:rgb 0 0 1) "##")
(sdegeo:define-contact-set "TopContact_west_3" 4 (color:rgb 0 0 1) "##")
(sdegeo:define-contact-set "TopContact_west_4" 4 (color:rgb 0 0 1) "##")

(sdegeo:define-contact-set "TopContact_east_1" 4 (color:rgb 0 0 1) "##")
(sdegeo:define-contact-set "TopContact_east_2" 4 (color:rgb 0 0 1) "##")
(sdegeo:define-contact-set "TopContact_east_3" 4 (color:rgb 0 0 1) "##")
(sdegeo:define-contact-set "TopContact_east_4" 4 (color:rgb 0 0 1) "##")

(sdegeo:define-contact-set "TopContact_X" 4 (color:rgb 0 0 1) "##")

; --------------------------------
; CELL +Y (with named top contact)
; --------------------------------
(define (CellY prefix x y z Wcell isCut cname)
  (mk-cuboid (string-append prefix "_SiO2") "SiO2" x y z BottomDielectricWidth Wcell
    BottomDielectricThickness)
  (mk-cuboid (string-append prefix "_TiB") "Titanium" x y (+ z BottomDielectricThickness)
    BottomDielectricWidth Wcell AdhesionLayerThickness)
  (define AUB (mk-cuboid (string-append prefix "_AuB") "Gold" x y (+ z BottomDielectricThickness
    AdhesionLayerThickness) BottomDielectricWidth Wcell GoldLayerThickness))
  (sdegeo:set-current-contact-set "BottomContact")
  (sdegeo:set-contact AUB "BottomContact")

  (define ztop (+ z BottomDielectricThickness AdhesionLayerThickness GoldLayerThickness))
  (define Lvac (max 0.0 (min TrenchMiddleWidth BottomDielectricWidth)))
  (define Lleft  (/ (- BottomDielectricWidth Lvac) 2.0))
  (define Lright (- BottomDielectricWidth (+ Lleft Lvac)))
  (if (> Lleft 0.0)  (mk-cuboid (string-append prefix "_HfO2_L") "HfO2" x y ztop Lleft Wcell
    TrenchWallHeight))
```

```
  (if (> Lvac 0.0)   (mk-cuboid (string-append prefix "_Vac")    "Vacuum" (+ x Lleft) y ztop Lvac
    Wcell TrenchWallHeight))
  (if (> Lright 0.0) (mk-cuboid (string-append prefix "_HfO2_R") "HfO2" (+ x Lleft Lvac) y ztop
    Lright Wcell TrenchWallHeight))

  (define zcap (+ ztop TrenchWallHeight))
  (define xcap (+ x (/ (- BottomDielectricWidth TopGoldWidthX) 2.0)))
  (define ycap (if isCut (+ y (- Wcell (+ TopGoldHeightY TitaniumRimThickness))) (+ y (/ (-
    CellLength TopGoldHeightY) 2.0))))
  (define AUT (mk-cuboid (string-append prefix "_AuT") "Gold" xcap ycap zcap TopGoldWidthX
    TopGoldHeightY TopGoldThicknessZ))
  (sdegeo:set-current-contact-set cname)
  (sdegeo:set-contact AUT cname)

  (mk-cuboid (string-append prefix "_TiFxL") "Titanium" (- xcap TitaniumRimThickness) ycap zcap
    TitaniumRimThickness TopGoldHeightY TopGoldThicknessZ)
  (mk-cuboid (string-append prefix "_TiFxR") "Titanium" (+ xcap TopGoldWidthX) ycap zcap
    TitaniumRimThickness TopGoldHeightY TopGoldThicknessZ)
  (mk-cuboid (string-append prefix "_TiFyF") "Titanium" (- xcap TitaniumRimThickness) (- ycap
    TitaniumRimThickness) zcap (+ TopGoldWidthX (* 2.0 TitaniumRimThickness)) TitaniumRimThickness
     TopGoldThicknessZ)
  (mk-cuboid (string-append prefix "_TiFyB") "Titanium" (- xcap TitaniumRimThickness) (+ ycap
    TopGoldHeightY) zcap (+ TopGoldWidthX (* 2.0 TitaniumRimThickness)) TitaniumRimThickness
    TopGoldThicknessZ)

  (define xin_min (- xcap TitaniumRimThickness))
  (define xin_max (+ xcap TopGoldWidthX  TitaniumRimThickness))
  (define yin_min (- ycap TitaniumRimThickness))
  (define yin_max (+ ycap TopGoldHeightY TitaniumRimThickness))
  (define xb_min x) (define xb_max (+ x BottomDielectricWidth))
  (define yb_min y) (define yb_max (+ y Wcell))
  (define Tleft  (max 0.0 (- xin_min xb_min)))
  (define Tright (max 0.0 (- xb_max xin_max)))
  (define Tfront (max 0.0 (- yin_min yb_min)))
  (define Tback  (max 0.0 (- yb_max yin_max)))
  (define dxcore (max 0.0 (- xin_max xin_min)))
  (if (> Tleft 0.0)  (mk-cuboid (string-append prefix "_SiO2FrL") "SiO2" xb_min yb_min zcap Tleft
    Wcell TopGoldThicknessZ))
  (if (> Tright 0.0) (mk-cuboid (string-append prefix "_SiO2FrR") "SiO2" xin_max yb_min zcap Tright
     Wcell TopGoldThicknessZ))
  (if (and (> Tfront 0.0) (> dxcore 0.0)) (mk-cuboid (string-append prefix "_SiO2FrF") "SiO2"
    xin_min yb_min zcap dxcore Tfront TopGoldThicknessZ))
  (if (and (> Tback 0.0)  (> dxcore 0.0)) (mk-cuboid (string-append prefix "_SiO2FrB") "SiO2"
    xin_min yin_max zcap dxcore Tback TopGoldThicknessZ))
)

; -------------------------------
; CELL +X (with named top contact)
; -------------------------------
(define (CellX prefix x y z WcellX isCut cname)
  (mk-cuboid (string-append prefix "_SiO2") "SiO2" x y z WcellX BottomDielectricWidth
    BottomDielectricThickness)
  (mk-cuboid (string-append prefix "_TiB") "Titanium" x y (+ z BottomDielectricThickness) WcellX
    BottomDielectricWidth AdhesionLayerThickness)
  (define AUB (mk-cuboid (string-append prefix "_AuB") "Gold" x y (+ z BottomDielectricThickness
    AdhesionLayerThickness) WcellX BottomDielectricWidth GoldLayerThickness))
  (sdegeo:set-current-contact-set "BottomContact")
  (sdegeo:set-contact AUB "BottomContact")

  (define ztop (+ z BottomDielectricThickness AdhesionLayerThickness GoldLayerThickness))
  (define Ly_vac (max 0.0 (min TrenchMiddleWidth BottomDielectricWidth)))
  (define Ly_left  (/ (- BottomDielectricWidth Ly_vac) 2.0))
  (define Ly_right (- BottomDielectricWidth (+ Ly_left Ly_vac)))
```

```
  (if (> Ly_left 0.0)  (mk-cuboid (string-append prefix "_HfO2_L") "HfO2" x y ztop WcellX Ly_left
    TrenchWallHeight))
  (if (> Ly_vac 0.0)   (mk-cuboid (string-append prefix "_Vac")    "Vacuum" x (+ y Ly_left) ztop
    WcellX Ly_vac TrenchWallHeight))
  (if (> Ly_right 0.0) (mk-cuboid (string-append prefix "_HfO2_R") "HfO2" x (+ y Ly_left Ly_vac)
    ztop WcellX Ly_right TrenchWallHeight))

  (define zcap (+ ztop TrenchWallHeight))
  (define xcap (if isCut (+ x (- WcellX (+ TopGoldHeightY TitaniumRimThickness))) (+ x (/ (-
    CellLength TopGoldHeightY) 2.0))))
  (define ycap (+ y (/ (- BottomDielectricWidth TopGoldWidthX) 2.0)))
  (define AUT (mk-cuboid (string-append prefix "_AuT") "Gold" xcap ycap zcap TopGoldHeightY
    TopGoldWidthX TopGoldThicknessZ))
  (sdegeo:set-current-contact-set cname)
  (sdegeo:set-contact AUT cname)

  (mk-cuboid (string-append prefix "_TiFxL") "Titanium" (- xcap TitaniumRimThickness) ycap zcap
    TitaniumRimThickness TopGoldWidthX TopGoldThicknessZ)
  (mk-cuboid (string-append prefix "_TiFxR") "Titanium" (+ xcap TopGoldHeightY) ycap zcap
    TitaniumRimThickness TopGoldWidthX TopGoldThicknessZ)
  (mk-cuboid (string-append prefix "_TiFyF") "Titanium" (- xcap TitaniumRimThickness) (- ycap
    TitaniumRimThickness) zcap (+ TopGoldHeightY (* 2.0 TitaniumRimThickness))
    TitaniumRimThickness TopGoldThicknessZ)
  (mk-cuboid (string-append prefix "_TiFyB") "Titanium" (- xcap TitaniumRimThickness) (+ ycap
    TopGoldWidthX) zcap (+ TopGoldHeightY (* 2.0 TitaniumRimThickness)) TitaniumRimThickness
    TopGoldThicknessZ)

  (define xin_min (- xcap TitaniumRimThickness))
  (define xin_max (+ xcap TopGoldHeightY TitaniumRimThickness))
  (define yin_min (- ycap TitaniumRimThickness))
  (define yin_max (+ ycap TopGoldWidthX  TitaniumRimThickness))
  (define xb_min x) (define xb_max (+ x WcellX))
  (define yb_min y) (define yb_max (+ y BottomDielectricWidth))
  (define Tleft  (max 0.0 (- xin_min xb_min)))
  (define Tright (max 0.0 (- xb_max xin_max)))
  (define Tfront (max 0.0 (- yin_min yb_min)))
  (define Tback  (max 0.0 (- yb_max yin_max)))
  (define dxcore (max 0.0 (- xin_max xin_min)))
  (if (> Tleft 0.0)  (mk-cuboid (string-append prefix "_SiO2FrL") "SiO2" xb_min yb_min zcap Tleft
    BottomDielectricWidth TopGoldThicknessZ))
  (if (> Tright 0.0) (mk-cuboid (string-append prefix "_SiO2FrR") "SiO2" xin_max yb_min zcap Tright
     BottomDielectricWidth TopGoldThicknessZ))
  (if (and (> Tfront 0.0) (> dxcore 0.0)) (mk-cuboid (string-append prefix "_SiO2FrF") "SiO2"
    xin_min yb_min zcap dxcore Tfront TopGoldThicknessZ))
  (if (and (> Tback 0.0)  (> dxcore 0.0)) (mk-cuboid (string-append prefix "_SiO2FrB") "SiO2"
    xin_min yin_max zcap dxcore Tback TopGoldThicknessZ))
)

; --------------------------------
; CORNER FILLERS (4 layers)
; --------------------------------
(define (CornerLayers prefix x y z span)
  (define zc z)
  (mk-cuboid (string-append prefix "_L1_SiO2") "SiO2" x y zc span span (+ BottomDielectricThickness
     AdhesionLayerThickness))
  (set! zc (+ zc (+ BottomDielectricThickness AdhesionLayerThickness)))
  (define GOLD_FILL (mk-cuboid (string-append prefix "_L2_Au") "Gold" x y zc span span
    GoldLayerThickness))
  (sdegeo:set-current-contact-set "BottomContact")
  (sdegeo:set-contact GOLD_FILL "BottomContact")
  (set! zc (+ zc GoldLayerThickness))
  (mk-cuboid (string-append prefix "_L3_HfO2") "HfO2" x y zc span span TrenchWallHeight)
  (set! zc (+ zc TrenchWallHeight))
```

```
  (mk-cuboid (string-append prefix "_L4_SiO2") "SiO2" x y zc span span TopGoldThicknessZ))

;; ======================================================================
;; GEOMETRY BUILD (N=4)
;; ======================================================================

;; SOUTH branch (Y-): cut..cut..std near center
(define y_cursor y0)
(CellY "Ypre1" x0 y_cursor z0 Pitch #t "TopContact_south_1") (set! y_cursor (+ y_cursor Pitch))
(CellY "Ypre2" x0 y_cursor z0 Pitch #t "TopContact_south_2") (set! y_cursor (+ y_cursor Pitch))
(CellY "Ypre3" x0 y_cursor z0 Pitch #t "TopContact_south_3") (set! y_cursor (+ y_cursor Pitch))
(CellY "Ypre4" x0 y_cursor z0 CellLength #f "TopContact_south_4") (set! y_cursor (+ y_cursor
    CellLength))

; ================ CENTER BLOCK _ layered stack + TopContact_X
(define xB x0)
(define yB y_cursor)

(mk-cuboid "BLOCK_SiO2" "SiO2" xB yB z0 BottomDielectricWidth BottomDielectricWidth
    BottomDielectricThickness)
(mk-cuboid "BLOCK_TiB" "Titanium" xB yB (+ z0 BottomDielectricThickness) BottomDielectricWidth
    BottomDielectricWidth AdhesionLayerThickness)
(define AUB_CENTER (mk-cuboid "BLOCK_AuB" "Gold" xB yB (+ z0 BottomDielectricThickness
    AdhesionLayerThickness) BottomDielectricWidth BottomDielectricWidth GoldLayerThickness))
(sdegeo:set-current-contact-set "BottomContact")
(sdegeo:set-contact AUB_CENTER "BottomContact")
(define z_topBottom (+ z0 BottomDielectricThickness AdhesionLayerThickness GoldLayerThickness))
(mk-cuboid "BLOCK_Vac" "Vacuum" xB yB z_topBottom BottomDielectricWidth BottomDielectricWidth
    TrenchWallHeight)
(define zcapC (+ z_topBottom TrenchWallHeight))
(define capSide (min CenterCapSide BottomDielectricWidth))
(define xcapC (+ xB (/ (- BottomDielectricWidth capSide) 2.0)))
(define ycapC (+ yB (/ (- BottomDielectricWidth capSide) 2.0)))
(define AuC (mk-cuboid "BLOCK_AuT" "Gold" xcapC ycapC zcapC capSide capSide TopGoldThicknessZ))
(sdegeo:set-current-contact-set "TopContact_X")
(sdegeo:set-contact AuC "TopContact_X")
(mk-cuboid "BLOCK_TiL" "Titanium" (- xcapC TitaniumRimThickness) ycapC zcapC TitaniumRimThickness
    capSide TopGoldThicknessZ)
(mk-cuboid "BLOCK_TiR" "Titanium" (+ xcapC capSide) ycapC zcapC TitaniumRimThickness capSide
    TopGoldThicknessZ)
(mk-cuboid "BLOCK_TiF" "Titanium" (- xcapC TitaniumRimThickness) (- ycapC TitaniumRimThickness)
    zcapC (+ capSide (* 2.0 TitaniumRimThickness)) TitaniumRimThickness TopGoldThicknessZ)
(mk-cuboid "BLOCK_TiB" "Titanium" (- xcapC TitaniumRimThickness) (+ ycapC capSide) zcapC (+ capSide
     (* 2.0 TitaniumRimThickness)) TitaniumRimThickness TopGoldThicknessZ)
(define xb_min xB) (define xb_max (+ xB BottomDielectricWidth))
(define yb_min yB) (define yb_max (+ yB BottomDielectricWidth))
(define xin_min (- xcapC TitaniumRimThickness))
(define xin_max (+ xcapC capSide  TitaniumRimThickness))
(define yin_min (- ycapC TitaniumRimThickness))
(define yin_max (+ ycapC capSide  TitaniumRimThickness))
(define TleftC  (max 0.0 (- xin_min xb_min)))
(define TrightC (max 0.0 (- xb_max xin_max)))
(define TfrontC (max 0.0 (- yin_min yb_min)))
(define TbackC  (max 0.0 (- yb_max yin_max)))
(define dxcoreC (max 0.0 (- xin_max xin_min)))
(if (> TleftC 0.0)  (mk-cuboid "BLOCK_SiO2FrL" "SiO2" xb_min yb_min zcapC TleftC
    BottomDielectricWidth TopGoldThicknessZ))
(if (> TrightC 0.0) (mk-cuboid "BLOCK_SiO2FrR" "SiO2" xin_max yb_min zcapC TrightC
    BottomDielectricWidth TopGoldThicknessZ))
(if (and (> TfrontC 0.0) (> dxcoreC 0.0)) (mk-cuboid "BLOCK_SiO2FrF" "SiO2" xin_min yb_min zcapC
    dxcoreC TfrontC TopGoldThicknessZ))
(if (and (> TbackC 0.0)  (> dxcoreC 0.0)) (mk-cuboid "BLOCK_SiO2FrB" "SiO2" xin_min yin_max zcapC
    dxcoreC TbackC TopGoldThicknessZ))
```

```
; NORTH branch (Y+): std near center, then cuts
(set! y_cursor (+ yB BottomDielectricWidth))
(CellY "Ypost1" x0 y_cursor z0 CellLength #f "TopContact_north_4") (set! y_cursor (+ y_cursor
    CellLength))
(CellY "Ypost2" x0 y_cursor z0 Pitch #t "TopContact_north_3") (set! y_cursor (+ y_cursor Pitch))
(CellY "Ypost3" x0 y_cursor z0 Pitch #t "TopContact_north_2") (set! y_cursor (+ y_cursor Pitch))
(CellY "Ypost4" x0 y_cursor z0 Pitch #t "TopContact_north_1") (set! y_cursor (+ y_cursor Pitch))

;; EAST branch (X+): std near center, then cuts
(define xB2 (+ xB BottomDielectricWidth))
(define x_cursor xB2)
(CellX "Xpos1" x_cursor yB z0 CellLength #f "TopContact_east_1") (set! x_cursor (+ x_cursor
    CellLength))
(CellX "Xpos2" x_cursor yB z0 Pitch #t "TopContact_east_2") (set! x_cursor (+ x_cursor Pitch))
(CellX "Xpos3" x_cursor yB z0 Pitch #t "TopContact_east_3") (set! x_cursor (+ x_cursor Pitch))
(CellX "Xpos4" x_cursor yB z0 Pitch #t "TopContact_east_4") (set! x_cursor (+ x_cursor Pitch))

;; WEST branch (X-): std near center, then cuts to the left
(set! x_cursor xB)
(define x_origin (- x_cursor CellLength))
(CellX "Xneg4" x_origin yB z0 CellLength #f "TopContact_west_4") (set! x_cursor x_origin)
(set! x_origin (- x_cursor Pitch))
(CellX "Xneg3" x_origin yB z0 Pitch #t "TopContact_west_3") (set! x_cursor x_origin)
(set! x_origin (- x_cursor Pitch))
(CellX "Xneg2" x_origin yB z0 Pitch #t "TopContact_west_2") (set! x_cursor x_origin)
(set! x_origin (- x_cursor Pitch))
(CellX "Xneg1" x_origin yB z0 Pitch #t "TopContact_west_1") (set! x_cursor x_origin)

(define yB2 (+ yB BottomDielectricWidth))
(define arm_span (+ CellLength (* (- Ncells 1) Pitch)))
(CornerLayers "F_NE" xB2 yB2 z0 arm_span)
(CornerLayers "F_NW" (- xB arm_span) yB2 z0 arm_span)
(CornerLayers "F_SW" (- xB arm_span) (- yB arm_span) z0 arm_span)
(CornerLayers "F_SE" xB2 (- yB arm_span) z0 arm_span)

; ========================================================================
; MESH REFINEMENT
; ========================================================================
(sdedr:define-refinement-function "RFn" "MaxLenInt" "Gold" "Vacuum"   0.0001 "DoubleSide")
(sdedr:define-refinement-function "RFn" "MaxLenInt" "Gold" "Titanium" 0.0001)

; ========================================================================
; BUILD
; ========================================================================
(sde:build-mesh "n@node@")
```

Listing 6: SDevice Code for MV Vacuum

```
File {
  Grid          = "n1_msh.tdr"
  Plot          = "n@node@_clock_des.tdr"
  Current   = "n@node@_clock_des.plt"
  Parameter = "sdevice.par"
}


Electrode {

*BOTTOMCONTACT

  { Name = "BottomContact"                     Voltage = 0.0 }
```

```
*INPUTS SEQUENCES TOPCONTACT

  { Name = "TopContact_south_1"   Voltage = 0.0 }
  { Name = "TopContact_south_2"   Voltage = 0.0 }
  { Name = "TopContact_south_3"   Voltage = 0.0 }
  { Name = "TopContact_south_4"   Voltage = 0.0 }

  { Name = "TopContact_north_1"   Voltage = 0.0 }
  { Name = "TopContact_north_2"   Voltage = 0.0 }
  { Name = "TopContact_north_3"   Voltage = 0.0 }
  { Name = "TopContact_north_4"   Voltage = 0.0 }

  { Name = "TopContact_west_1"   Voltage = 0.0 }
  { Name = "TopContact_west_2"   Voltage = 0.0 }
  { Name = "TopContact_west_3"   Voltage = 0.0 }
  { Name = "TopContact_west_4"   Voltage = 0.0 }

*CENTER CELL TOPCONTACT

  { Name = "TopContact_X"       Voltage = 0.0 }


*OUTPUT TOPCONTACT

  { Name = "TopContact_east_1"   Voltage = 0.0 }
  { Name = "TopContact_east_2"   Voltage = 0.0 }
  { Name = "TopContact_east_3"   Voltage = 0.0 }
  { Name = "TopContact_east_4"   Voltage = 0.0 }

}


# ----------------------------
# PHYSICS
# ----------------------------
Physics (Material="HfO2") {
  CondInsulator
}

Physics (Material="SiO2") {
  CondInsulator
}

Thermode {

  { Name = "TopContact_south_1"  Temperature = 300 SurfaceResistance = 1e-5 }
  { Name = "TopContact_south_2"  Temperature = 300 SurfaceResistance = 1e-5 }
  { Name = "TopContact_south_3"  Temperature = 300 SurfaceResistance = 1e-5 }
  { Name = "TopContact_south_4"  Temperature = 300 SurfaceResistance = 1e-5 }

  { Name = "TopContact_north_1"  Temperature = 300 SurfaceResistance = 1e-5 }
  { Name = "TopContact_north_2"  Temperature = 300 SurfaceResistance = 1e-5 }
  { Name = "TopContact_north_3"  Temperature = 300 SurfaceResistance = 1e-5 }
  { Name = "TopContact_north_4"  Temperature = 300 SurfaceResistance = 1e-5 }

  { Name = "TopContact_west_1"  Temperature = 300 SurfaceResistance = 1e-5 }
  { Name = "TopContact_west_2"  Temperature = 300 SurfaceResistance = 1e-5 }
  { Name = "TopContact_west_3"  Temperature = 300 SurfaceResistance = 1e-5 }
  { Name = "TopContact_west_4"  Temperature = 300 SurfaceResistance = 1e-5 }

  { Name = "TopContact_X"                      Temperature = 300 SurfaceResistance = 1e-5 }
```

```
  { Name = "TopContact_east_1"  Temperature = 300 SurfaceResistance = 1e-5 }
  { Name = "TopContact_east_2"  Temperature = 300 SurfaceResistance = 1e-5 }
  { Name = "TopContact_east_3"  Temperature = 300 SurfaceResistance = 1e-5 }
  { Name = "TopContact_east_4"  Temperature = 300 SurfaceResistance = 1e-5 }

  { Name = "BottomContact"                      Temperature = 300 SurfaceResistance = 1e-5 }
}

# ----------------------------
# PLOT
# ----------------------------
Plot {
  Potential
  ElectricField
  DielectricConstant
  Temperature
  ConductionCurrentDensity/Vector
  DisplacementCurrentDensity/Vector
  TotalCurrent/Vector
  SpaceCharge
  Potential Doping
  BandGap ElectronAffinity
  ConductionBandEnergy ValenceBandEnergy
}

Math {
  RelErrControl
  Extrapolate
}

Solve {
  Coupled (Iterations= 100 LineSearchDamping= 1e-8) {Poisson}
  Coupled{ Poisson Temperature Contact CondInsulator }
  Plot(FilePrefix="n@node@_equilibrium")

  # ======================================================================
  # ==== CONFIG (cfg1) with startPhase=S; inputs 1..N, center N+1, east N+2..N+1+N ====
  quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4

  Goal{ name = "BottomContact"                  voltage = 0.0 }

  Goal{ name = "TopContact_south_1"     voltage = 3.0 }
  Goal{ name = "TopContact_south_2"     voltage = 0.0 }
  Goal{ name = "TopContact_south_3"     voltage = -3.0 }
  Goal{ name = "TopContact_south_4"     voltage = 3.0 }

  Goal{ name = "TopContact_north_1"     voltage = 3.0 }
  Goal{ name = "TopContact_north_2"     voltage = 0.0 }
  Goal{ name = "TopContact_north_3"     voltage = -3.0 }
  Goal{ name = "TopContact_north_4"     voltage = 3.0 }

  Goal{ name = "TopContact_west_1"      voltage = 3.0 }
  Goal{ name = "TopContact_west_2"      voltage = 0.0 }
  Goal{ name = "TopContact_west_3"      voltage = -3.0 }
  Goal{ name = "TopContact_west_4"      voltage = 3.0 }

  Goal{ name = "TopContact_X"                   voltage = 0.0 }

  Goal{ name = "TopContact_east_1"      voltage = -3.0 }
  Goal{ name = "TopContact_east_2"      voltage = 3.0 }
  Goal{ name = "TopContact_east_3"      voltage = 0.0 }
  Goal{ name = "TopContact_east_4"      voltage = -3.0 }
```

```
      plot { range=(0, 1) intervals= 1}
){coupled { Poisson Temperature CondInsulator }}

Plot(FilePrefix="n@node@_cfg1")

# ---- return to 0 V on all top contacts
quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
Goal{ name = "BottomContact"                  voltage = 0.0 }

Goal{ name = "TopContact_south_1"     voltage = 0.0 }
Goal{ name = "TopContact_south_2"     voltage = 0.0 }
Goal{ name = "TopContact_south_3"     voltage = 0.0 }
Goal{ name = "TopContact_south_4"     voltage = 0.0 }

Goal{ name = "TopContact_north_1"     voltage = 0.0 }
Goal{ name = "TopContact_north_2"     voltage = 0.0 }
Goal{ name = "TopContact_north_3"     voltage = 0.0 }
Goal{ name = "TopContact_north_4"     voltage = 0.0 }

Goal{ name = "TopContact_west_1"      voltage = 0.0 }
Goal{ name = "TopContact_west_2"      voltage = 0.0 }
Goal{ name = "TopContact_west_3"      voltage = 0.0 }
Goal{ name = "TopContact_west_4"      voltage = 0.0 }

Goal{ name = "TopContact_X"                   voltage = 0.0 }

Goal{ name = "TopContact_east_1"      voltage = 0.0 }
Goal{ name = "TopContact_east_2"      voltage = 0.0 }
Goal{ name = "TopContact_east_3"      voltage = 0.0 }
Goal{ name = "TopContact_east_4"      voltage = 0.0 }

      plot { range=(0, 1) intervals= 1}
){coupled { Poisson Temperature CondInsulator }}

# ======================================================================
# ==== CONFIG (cfg2) with startPhase=R; inputs 1..N, center N+1, east N+2..N+1+N ====
quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4

Goal{ name = "BottomContact"                  voltage = 0.0 }

Goal{ name = "TopContact_south_1"     voltage = -3.0 }
Goal{ name = "TopContact_south_2"     voltage = 3.0 }
Goal{ name = "TopContact_south_3"     voltage = 0.0 }
Goal{ name = "TopContact_south_4"     voltage = -3.0 }

Goal{ name = "TopContact_north_1"     voltage = -3.0 }
Goal{ name = "TopContact_north_2"     voltage = 3.0 }
Goal{ name = "TopContact_north_3"     voltage = 0.0 }
Goal{ name = "TopContact_north_4"     voltage = -3.0 }

Goal{ name = "TopContact_west_1"      voltage = -3.0 }
Goal{ name = "TopContact_west_2"      voltage = 3.0 }
Goal{ name = "TopContact_west_3"      voltage = 0.0 }
Goal{ name = "TopContact_west_4"      voltage = -3.0 }

Goal{ name = "TopContact_X"                   voltage = 3.0 }

Goal{ name = "TopContact_east_1"      voltage = 0.0 }
Goal{ name = "TopContact_east_2"      voltage = -3.0 }
Goal{ name = "TopContact_east_3"      voltage = 3.0 }
Goal{ name = "TopContact_east_4"      voltage = 0.0 }

      plot { range=(0, 1) intervals= 1}
```

```
){coupled { Poisson Temperature CondInsulator }}

Plot(FilePrefix="n@node@_cfg2")

# ---- return to 0 V on all top contacts
quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4
Goal{ name = "BottomContact"                  voltage = 0.0 }

Goal{ name = "TopContact_south_1"     voltage = 0.0 }
Goal{ name = "TopContact_south_2"     voltage = 0.0 }
Goal{ name = "TopContact_south_3"     voltage = 0.0 }
Goal{ name = "TopContact_south_4"     voltage = 0.0 }

Goal{ name = "TopContact_north_1"     voltage = 0.0 }
Goal{ name = "TopContact_north_2"     voltage = 0.0 }
Goal{ name = "TopContact_north_3"     voltage = 0.0 }
Goal{ name = "TopContact_north_4"     voltage = 0.0 }

Goal{ name = "TopContact_west_1"      voltage = 0.0 }
Goal{ name = "TopContact_west_2"      voltage = 0.0 }
Goal{ name = "TopContact_west_3"      voltage = 0.0 }
Goal{ name = "TopContact_west_4"      voltage = 0.0 }

Goal{ name = "TopContact_X"                   voltage = 0.0 }

Goal{ name = "TopContact_east_1"      voltage = 0.0 }
Goal{ name = "TopContact_east_2"      voltage = 0.0 }
Goal{ name = "TopContact_east_3"      voltage = 0.0 }
Goal{ name = "TopContact_east_4"      voltage = 0.0 }

      plot { range=(0, 1) intervals= 1}
){coupled { Poisson Temperature CondInsulator }}

# ======================================================================
# ==== CONFIG (cfg3) with startPhase=H; inputs 1..N, center N+1, east N+2..N+1+N ====
quasistationary (InitialStep = 0.01 Increment = 1.5 MaxStep = 0.05 MinStep=1e-4

Goal{ name = "BottomContact"                  voltage = 0.0 }

Goal{ name = "TopContact_south_1"     voltage = 0.0 }
Goal{ name = "TopContact_south_2"     voltage = -3.0 }
Goal{ name = "TopContact_south_3"     voltage = 3.0 }
Goal{ name = "TopContact_south_4"     voltage = 0.0 }

Goal{ name = "TopContact_north_1"     voltage = 0.0 }
Goal{ name = "TopContact_north_2"     voltage = -3.0 }
Goal{ name = "TopContact_north_3"     voltage = 3.0 }
Goal{ name = "TopContact_north_4"     voltage = 0.0 }

Goal{ name = "TopContact_west_1"      voltage = 0.0 }
Goal{ name = "TopContact_west_2"      voltage = -3.0 }
Goal{ name = "TopContact_west_3"      voltage = 3.0 }
Goal{ name = "TopContact_west_4"      voltage = 0.0 }

Goal{ name = "TopContact_X"                   voltage = -3.0 }

Goal{ name = "TopContact_east_1"      voltage = 3.0 }
Goal{ name = "TopContact_east_2"      voltage = 0.0 }
Goal{ name = "TopContact_east_3"      voltage = -3.0 }
Goal{ name = "TopContact_east_4"      voltage = 3.0 }

      plot { range=(0, 1) intervals= 1}
){coupled { Poisson Temperature CondInsulator }}
```

```
  Plot(FilePrefix="n@node@_cfg3")

}
```

Listing 7: SDE: MV CrossVacuum Entire Center Block variation from MV Vacuum type [15]

```
; ================= CENTER BLOCK _ HfO2 mid + Cross vacuum + TopContact_X
(define xB x0)
(define yB y_cursor)

(mk-cuboid "BLOCK_SiO2" "SiO2" xB yB z0 BottomDielectricWidth BottomDielectricWidth
     BottomDielectricThickness)
(mk-cuboid "BLOCK_TiB" "Titanium" xB yB (+ z0 BottomDielectricThickness) BottomDielectricWidth
     BottomDielectricWidth AdhesionLayerThickness)
(define AUB_CENTER (mk-cuboid "BLOCK_AuB" "Gold" xB yB (+ z0 BottomDielectricThickness
     AdhesionLayerThickness) BottomDielectricWidth BottomDielectricWidth GoldLayerThickness))
(sdegeo:set-current-contact-set "BottomContact")
(sdegeo:set-contact AUB_CENTER "BottomContact")
(define z_topBottom (+ z0 BottomDielectricThickness AdhesionLayerThickness GoldLayerThickness))
(mk-cuboid "BLOCK_mid" "HfO2" xB yB z_topBottom BottomDielectricWidth BottomDielectricWidth
     TrenchWallHeight)
(define Wy_h (min TrenchMiddleWidth BottomDielectricWidth))
(define y_h (+ yB (/ (- BottomDielectricWidth Wy_h) 2.0)))
(mk-cuboid "BLOCK_vacH" "Vacuum" xB y_h z_topBottom BottomDielectricWidth Wy_h TrenchWallHeight)
(define Lx_v (min TrenchMiddleWidth BottomDielectricWidth))
(define x_v (+ xB (/ (- BottomDielectricWidth Lx_v) 2.0)))
(mk-cuboid "BLOCK_vacV" "Vacuum" x_v yB z_topBottom Lx_v BottomDielectricWidth TrenchWallHeight)
(define zcapC (+ z_topBottom TrenchWallHeight))
(define capSide (min CenterCapSide BottomDielectricWidth))
(define xcapC (+ xB (/ (- BottomDielectricWidth capSide) 2.0)))
(define ycapC (+ yB (/ (- BottomDielectricWidth capSide) 2.0)))
(define AuC (mk-cuboid "BLOCK_AuT" "Gold" xcapC ycapC zcapC capSide capSide TopGoldThicknessZ))
(sdegeo:set-current-contact-set "TopContact_X")
(sdegeo:set-contact AuC "TopContact_X")
(mk-cuboid "BLOCK_TiL" "Titanium" (- xcapC TitaniumRimThickness) ycapC zcapC TitaniumRimThickness
     capSide TopGoldThicknessZ)
(mk-cuboid "BLOCK_TiR" "Titanium" (+ xcapC capSide) ycapC zcapC TitaniumRimThickness capSide
     TopGoldThicknessZ)
(mk-cuboid "BLOCK_TiF" "Titanium" (- xcapC TitaniumRimThickness) (- ycapC TitaniumRimThickness)
     zcapC (+ capSide (* 2.0 TitaniumRimThickness)) TitaniumRimThickness TopGoldThicknessZ)
(mk-cuboid "BLOCK_TiB" "Titanium" (- xcapC TitaniumRimThickness) (+ ycapC capSide) zcapC (+ capSide
      (* 2.0 TitaniumRimThickness)) TitaniumRimThickness TopGoldThicknessZ)
(define xb_min xB) (define xb_max (+ xB BottomDielectricWidth))
(define yb_min yB) (define yb_max (+ yB BottomDielectricWidth))
(define xin_min (- xcapC TitaniumRimThickness))
(define xin_max (+ xcapC capSide  TitaniumRimThickness))
(define yin_min (- ycapC TitaniumRimThickness))
(define yin_max (+ ycapC capSide  TitaniumRimThickness))
(define TleftC  (max 0.0 (- xin_min xb_min)))
(define TrightC (max 0.0 (- xb_max xin_max)))
(define TfrontC (max 0.0 (- yin_min yb_min)))
(define TbackC  (max 0.0 (- yb_max yin_max)))
(define dxcoreC (max 0.0 (- xin_max xin_min)))
(if (> TleftC 0.0)  (mk-cuboid "BLOCK_SiO2FrL" "SiO2" xb_min yb_min zcapC TleftC
     BottomDielectricWidth TopGoldThicknessZ))
(if (> TrightC 0.0) (mk-cuboid "BLOCK_SiO2FrR" "SiO2" xin_max yb_min zcapC TrightC
     BottomDielectricWidth TopGoldThicknessZ))
(if (and (> TfrontC 0.0) (> dxcoreC 0.0)) (mk-cuboid "BLOCK_SiO2FrF" "SiO2" xin_min yb_min zcapC
     dxcoreC TfrontC TopGoldThicknessZ))
(if (and (> TbackC 0.0)  (> dxcoreC 0.0)) (mk-cuboid "BLOCK_SiO2FrB" "SiO2" xin_min yin_max zcapC
     dxcoreC TbackC TopGoldThicknessZ))
```

Bibliography

[1] C. S. Lent, B. Isaksen, and M. Lieberman, "Molecular Quantum-Dot Cellular Automata," *Journal of the American Chemical Society*, vol. 125, no. 4, pp. 1056–1063, 2003, doi: 10.1021/ja026856g.

[2] M. Graziano, R. Wang, M. Ruo Roch, Y. Ardesi, F. Riente, and G. Piccinini, "Characterisation of a bis-ferrocene molecular QCA wire on a non-ideal gold surface," *Micro & Nano Letters*, vol. 14, no. 1, pp. 22–27, 2019, doi: 10.1049/mnl.2018.5201.

[3] W. Hu, K. Sarveswaran, M. Lieberman, and G. H. Bernstein, "Sub-10 nm electron beam lithography using cold development of poly(methyl methacrylate)," *Journal of Vacuum Science & Technology B*, vol. 22, no. 4, pp. 1711–1716, 2004, doi: 10.1116/1.1763897.

[4] A. Loubat, L.-M. Lacroix, A. Robert, M. Impéror-Clerc, R. Poteau, L. Maron, R. Arenal, B. Pansu, and G. Viau, "Ultrathin Gold Nanowires: Soft-Templating versus Liquid Phase Synthesis, a Quantitative Study," *The Journal of Physical Chemistry C*, vol. 119, no. 8, pp. 4422–4430, 2015, doi: 10.1021/acs.jpcc.5b00242.

[5] *The International Technology Roadmap for Semiconductors 2.0: 2015 Edition — Emerging Research Materials (ERM)*, ITRS, San Jose, CA, 2015, Roadmap report, ch. *Emerging Research Materials (ERM)*.

[6] R. Listo, F. Ravera, G. Beretta, Y. Ardesi, G. Piccinini, and M. Graziano, " Unveiling charge dynamics in molecular field-coupled nanocomputing," in *Proc. 2024 IEEE 24th International Conference on Nanotechnology (NANO)*, Jul. 2024, pp. 424–429, doi: 10.1109/NANO61704.2024.10668388.

[7] J. Timler and C. S. Lent, "Power gain and dissipation in quantum-dot cellular automata," *Journal of Applied Physics*, vol. 91, no. 2, pp. 823–831, Jan. 2002, doi: 10.1063/1.1421217.

[8] M. Baldo, *Introduction to Nanoelectronics: Complete Course Notes (6.701/6.719)*, MIT OpenCourseWare, May 2011, OCW compilation of lecture notes.

[9] E. P. Blair, E. Yost, and C. S. Lent, "Power dissipation in clocking wires for clocked molecular quantum-dot cellular automata," *Journal of Computational Electronics*, vol. 9, pp. 49–55, Jun. 2010, doi: 10.1007/s10825-010-0308-1.

[10] S. Datta, *Quantum Transport: Atom to Transistor*. Cambridge, UK: Cambridge University Press, 2005, ISBN: 9780521631457.

[11] Y. Ardesi, U. Garlando, F. Riente, G. Beretta, G. Piccinini, and M. Graziano, "Taming molecular field-coupling for nanocomputing design," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 19, no. 1, pp. 1–24, Jan. 2022.

[12] M. Graziano and G. Piccinini, *Nanoelectronics Systems — Lecture Notes, Version V1.3 (Draft)*, Course notes, Politecnico di Torino, Sep. 2024, Draft V1.3 dated 23 Sep 2024.

[13] P.-S. Yang, P.-H. Cheng, C. R. Kao, and M.-J. Chen, "Novel self-shrinking mask for sub-3 nm pattern fabrication — Supplementary Information," Supplementary Information to *Scientific Reports* **6**:29625, 2016. (Complete article metadata to be confirmed in main paper.)

[14] M. Graziano, "Molecular QCA: AmeraReady manuscript/notes," unpublished manuscript, n.d. (Internal document; complete title/year/venue when available.)

[15] Synopsys, Inc. (2017). *Sentaurus™ Structure Editor User Guide* (Version N-2017.09). Synopsys, Inc., Mountain View, CA. Available at: `https://www.synopsys.com`. Accessed: November 13, 2025

[16] Synopsys, *Sentaurus Device User Guide*, Version S-2021.09, Synopsys Inc., Mountain View, CA, 2021.

[17] *50 Years of Moore's Law*, Intel Corporation, Santa Clara, CA, 2015, corporate retrospective.

[18] J. A. Hutchby, G. I. Bourianoff, V. V. Zhirnov, and J. E. Brewer, "Extending the road beyond CMOS," *IEEE Circuits and Devices Magazine*, vol. 18, no. 2, pp. 28–41, Mar. 2002.

[19] D. E. Nikonov and I. A. Young, "Overview of beyond-CMOS devices and a uniform methodology for their benchmarking," arXiv:1302.0244 [cond-mat.mes-hall], Sep. 2012. (Also published in *Proceedings of the IEEE*, vol. 101, no. 12, pp. 2498–2533, Dec. 2013.)

[20] K. Bernstein, R. K. Cavin, W. Porod, A. Seabaugh, and J. Welser, "Device and architecture outlook for beyond CMOS switches," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2169–2184, Dec. 2010.

[21] F. Ellinger, M. Claus, M. Schröter, and C. Carta, "Review of advanced and beyond CMOS FET technologies for radio frequency circuit design," in *Proc. SBMO/IEEE MTT-S International Microwave & Optoelectronics Conference (IMOC)*, Oct. 2011, pp. 347–351.

[22] A. Mackus, A. Bol, and W. Kessels, "The use of atomic layer deposition in advanced nanopatterning," *Nanoscale*, vol. 6, no. 19, pp. 10941–10960, 2014, doi: 10.1039/C4NR01954G.

[23] R. Singh and R. K. Ulrich, "High and low dielectric constant materials," *The Electrochemical Society Interface*, vol. 8, no. 2, pp. 26–30, 1999.

[24] M. Vacca, M. Graziano, and M. Zamboni, "Nanomagnetic logic microprocessor: Hierarchical power model," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 8, pp. 1410–1420, Aug. 2013 (first available online 2012), doi: 10.1109/TVLSI.2012.2213105.

[25] D. Fischer and A. Kersch, "The effect of dopants on the dielectric constant of HfO_2 and ZrO_2 from first principles," *Applied Physics Letters*, vol. 92, no. 1, p. 012908, Jan. 2008, doi: 10.1063/1.2828458.

[26] C. S. Lent, P. D. Tougaw, W. Porod, G. H. Bernstein (1993). "Quantum cellular automata", *Nanotechnology* **4**(1), 49–57. Available at: `https://doi.org/10.1088/0957-4484/4/1/004`. Accessed: November 19, 2025

[27] R. Wang, A. Pulimeno, M. R. Roch, V. Cauda, G. Piccinini, M. Graziano (2016). "Effect of a clock system on bis-ferrocene molecular QCA", *IEEE Trans. Nanotechnol.* **15**(4), 574–582. Available at: `https://doi.org/10.1109/TNANO.2016.2554120`. Accessed: November 19, 2025

[28] L. Zoli (2010). "Active bis-ferrocene molecules as unit for molecular computation", PhD Dissertation, Politecnico di Torino, Turin, Italy.
Available at: `https://iris.polito.it/handle/11583/2369819`. Accessed: November 19, 2025

[29] V. Arima, M. Iurlo, L. Zoli, A. Kumar, M. Fabani, P. Greco, M. A. Rampi, R. Giannelli, F. Paolucci, L. Marcaccio, C. A. Bortolotti, M. S. Magno, A. Vanossi, M. Gazzano, P. G. Mineo, A. G. Ficarra, A. Credi, G. Maruccio (2012). "Toward quantum-dot cellular automata units: thiolated-carbazole linked bisferrocenes", *Nanoscale* **4**(3), 813–823.
Available at: `https://doi.org/10.1039/C1NR11397K`. Accessed: November 19, 2025

[30] A. Pulimeno, M. Graziano, A. Sanginario, V. Cauda, R. Wang, G. Piccinini (2013). "Bis-ferrocene molecular QCA wire: ab-initio simulations of fabrication driven fault tolerance", *IEEE Trans. Nanotechnol.* **12**(4), 498–507.
Available at: `https://doi.org/10.1109/TNANO.2013.2261972`. Accessed: November 19, 2025

[31] G. Beretta (2020). "Study of field-coupled nanocomputing based on molecules for neural systems", Master's thesis, Politecnico di Torino, Torino, Italy.
Available at: `https://webthesis.biblio.polito.it/14703/`. Accessed: November 19, 2025

[32] F. Riente (2016). "Design methods and tools for nanocomputing: from silicon nanoarrays to nano magnetic logic", Master's thesis, Politecnico di Torino, Torino, Italy.
Available at: `https://webthesis.biblio.polito.it/3781/`. Accessed: November 19, 2025

[33] Ravera, Federico and Ardesi, Yuri and Piccinini, Gianluca and Graziano, Mariagrazia, "Technology-Aware Simulation for Prototyping Molecular Field-Coupled Nanocomputing," *IEEE Transactions on Nanotechnology*, vol. 23, pp. 521–528, 2024. doi: 10.1109/TNANO.2024.3415790.

[34] A. Pulimeno, M. Graziano, C. Abrardi, D. Demarchi, and G. Piccinini, "Molecular QCA: A write-in system based on electric fields," in *Proc. 4th IEEE International NanoElectronics Conference (INEC)*, Jun. 2011, pp. 1–2.

[35] Y. Ardesi, G. Turvani, M. Graziano, and G. Piccinini, "Scerpa simulation of clocked molecular field-coupling nanocomputing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 3, pp. 558–567, Mar. 2021, doi: 10.1109/TVLSI.2020.3043109.

[36] IEEE International Roadmap for Devices and Systems (IRDS), "Lithography and Patterning," Institute of Electrical and Electronics Engineers, 2023. [Online]. Available: `https://doi.org/10.60627/C13Z-V363`. Accessed: Mar. 6, 2025.