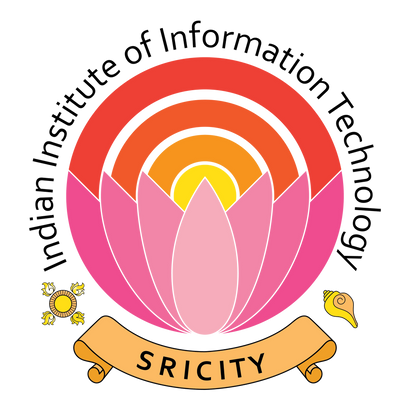
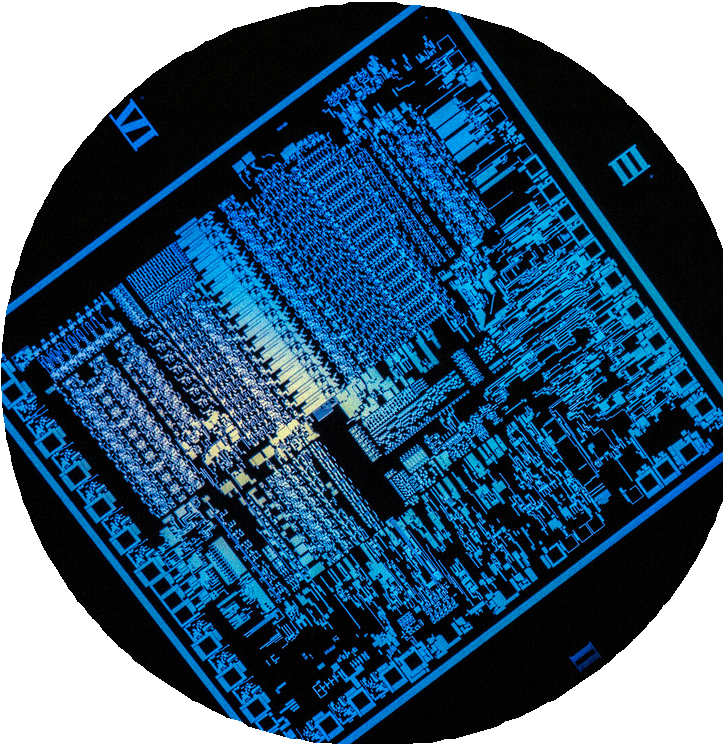
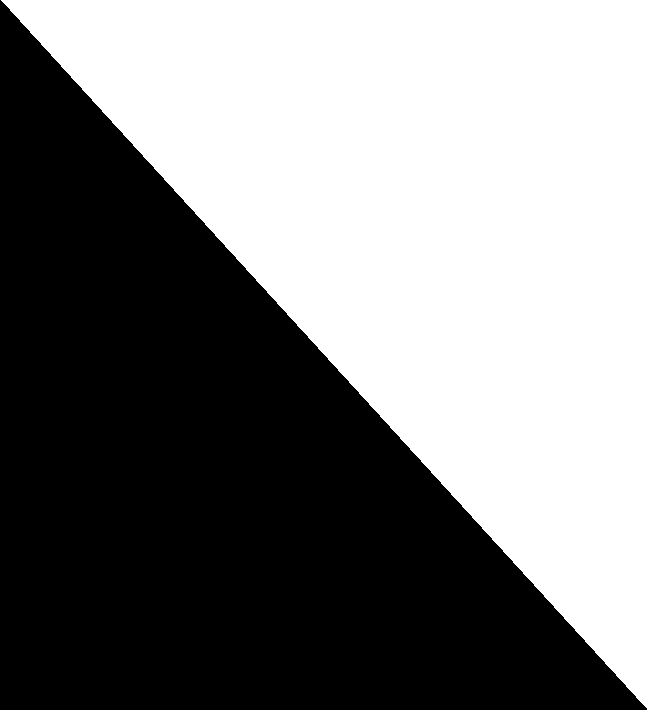
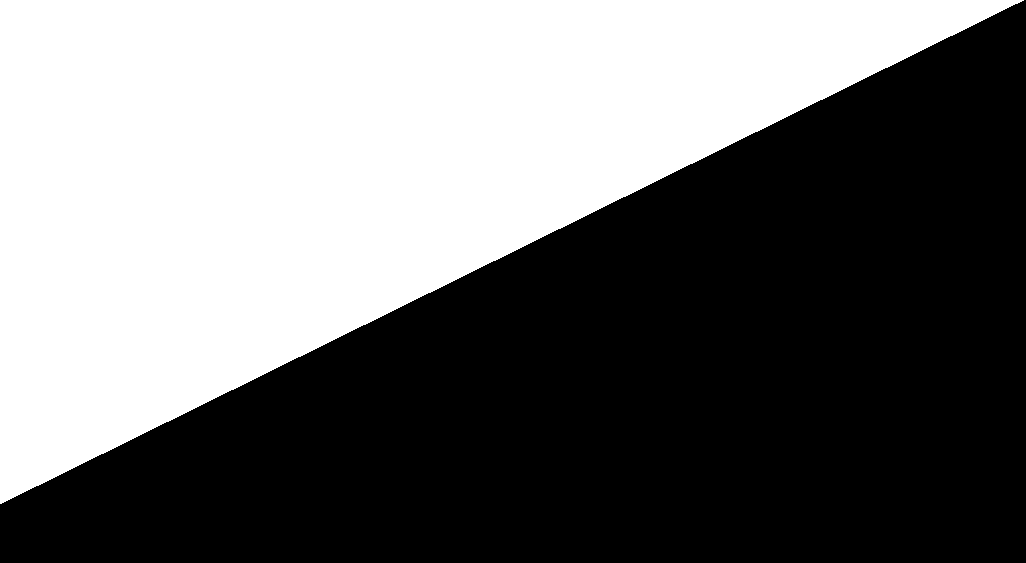
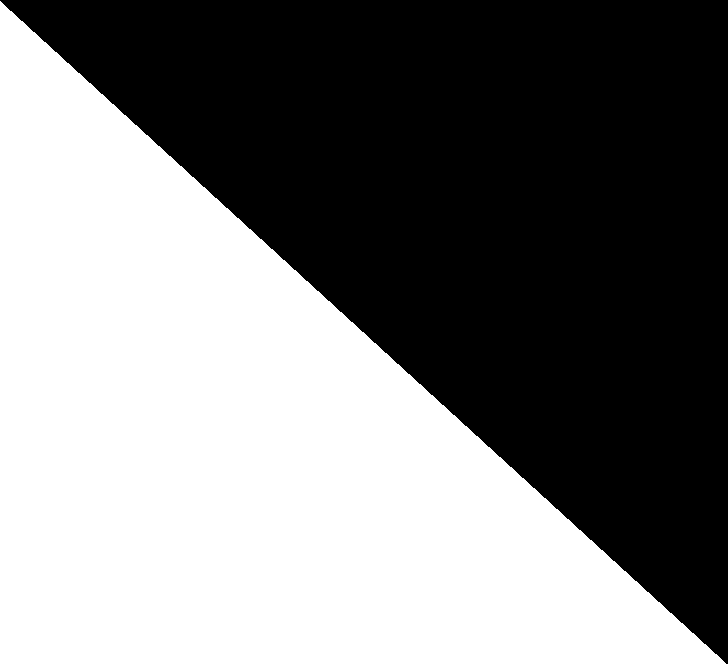
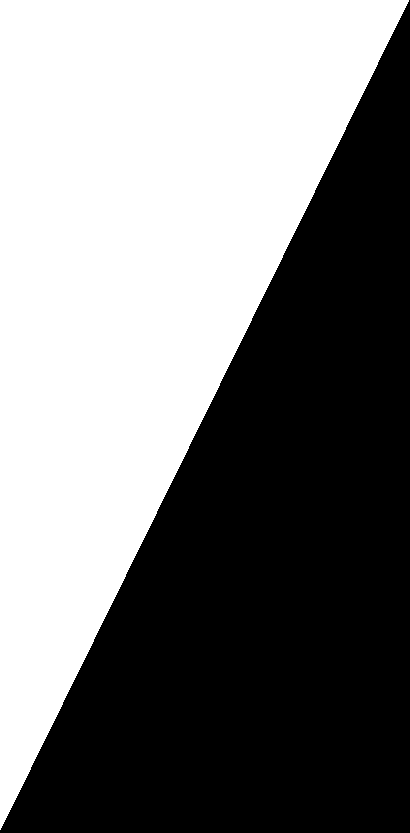
**Indian Institute of Information Technology Sri City, Chittoor**



(An Institute of National Importance under an Act of Parliament)

VLSI

**MONSOON – 2024 COURSE PROJECT**

**YASHWANTH S S20220020323 UG-3 ECE**

**Email:** [**yashwanth.s22@iiits.in**](mailto:yashwanth.s22@iiits.in)

**Quantum-Dot Cellular Automata Layout Generator**

**And Cross-bar Architecture Circuits**

**Abstract:**

Quantum-dot Cellular Automata (QCA) is gaining recognition as a viable replacement for CMOS transistor technology, with the potential to create logic circuits utilizing quantum devices such as quantum dots or single-domain nanomagnets. Nonetheless, designing and implementing QCA circuits demands specialized tools due to their distinct properties. While tools such as QCADesigner are available for manual layout and simulation, there is a shortage of automated tools for generating QCA layouts. To address this gap, we introduce the QCA-Layout Generator (QCA-LG), which is integrated into a broader design process for QCA technology. QCA-LG can accept standard formats from commonly used synthesis tools and automatically produce layouts that comply with the standards set by QCADesigner. This capability enables the automatic conversion of logical circuits defined in VHDL into layouts, which can subsequently be refined manually and simulated using QCADesigner. Additionally, we present a new design methodology specifically developed for QCA circuits, grounded in universal design principles for QCA and a generic QCA crossbar structure. This approach offers a systematic method for tailoring the crossbar architecture to realize any required logic function. By implementing this methodology, we illustrate the efficacy, accessibility, and cohesion of the QCA circuit design processes for both combinational and sequential circuits.

**Introduction:**

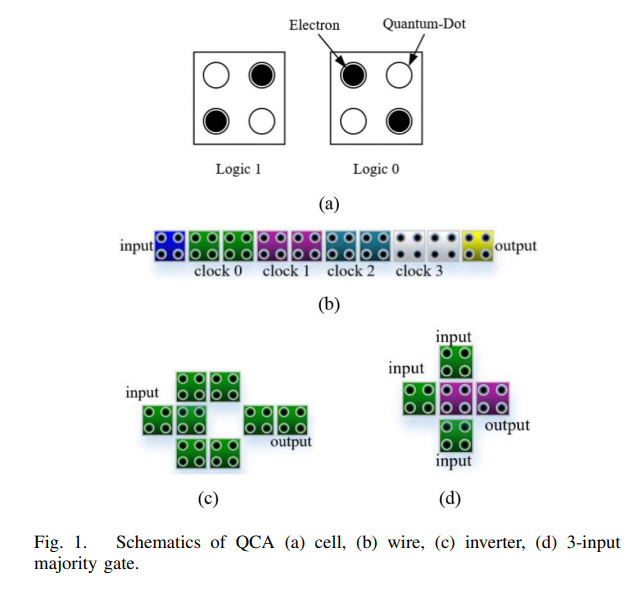
In the foreseeable future, the existing CMOS technology might struggle to keep pace with the rapid expansion of the technology. As an alternative, Quantum-dot Cellular Automata (QCA) presents a promising solution, given its adaptability across various physical systems. Quantum dots, in particular, exhibit significant promise due to their speed of operation and integration density. This project zeroes in on QCA realized with quantum dots, akin to QCA Designer. Although software tools have been established to assist with the design of QCA circuits and layouts, the current approaches for mapping logic to majority gates are not optimal. To remedy this, the paper introduces the QCA-LG tool, intended to automate the generation of layouts for QCA combinational circuits. This tool effectively maps logic circuits to QCA gates and produces the corresponding layouts, while supporting widely-used netlist formats in addition to the layout format of QCA Designer. This development enables the visualization and physical simulation of QCA layouts for the first time.

Moreover, we present a comprehensive methodology for the reliable and efficient design of QCA circuits utilizing the proposed architecture. This approach takes into account the inputs and outputs, as well as the programming lines of the crossbar architecture, to implement Boolean functions and standard QCA circuits.

**Background:**

Similar to how traditional circuits utilize transistors, QCA cells serve as the fundamental components in QCA-based circuits. Each QCA cell is comprised of four quantum dots positioned at the corners of a square configuration, which is illustrated schematically in Fig. 1(a). Within each cell, there are two additional free electrons that can move between the quantum dots. Due to Coulombic interactions, two distinct polarizations—defined by the positions of these electrons—can be established. The extra free electrons in each cell occupy opposite sites as a result of these interactions, leading to two potential ground states with polarizations of ‘1’ and ‘−1’, which represent the binary states ‘0’ and ‘1’, respectively. These two states of a QCA cell are depicted in Fig. 1(a). The clocking mechanism typically employed in QCA designs consists of a four-stage clock system, which minimizes signal metastability and guarantees proper signal propagation throughout the circuit.

A QCA wire is formed by a sequence of adjacent cells, as shown in Figure 1(b); this wire contains four types of clocked QCA cells. The primary components of any QCA circuit are the inverter and the majority gate. The fundamental logic gates in QCA technology include the inverter and the 3-input majority gate. As illustrated in Fig. 1(c), the inverter functionally alters the input signal's value. Additionally, Fig. 1(d) presents a schematic of the 3-input majority gate, which performs the logic function M (a, b, c) = ab + ac + bc. It is evident from this function that by fixing one of the inputs to ‘1’ or ‘0’, it can be easily transformed into a 2-input OR or AND gate, respectively. In QCA technology, all logic operations are executed through combinations of the inverter and the 3-input majority gate.

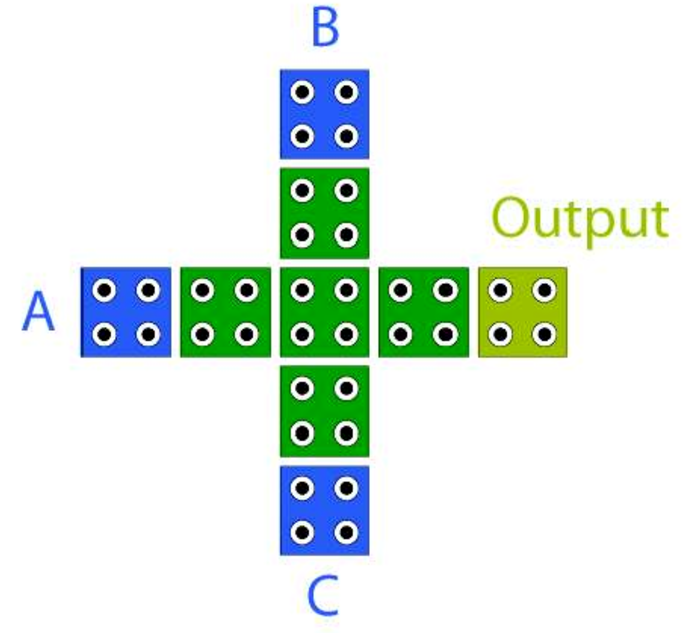
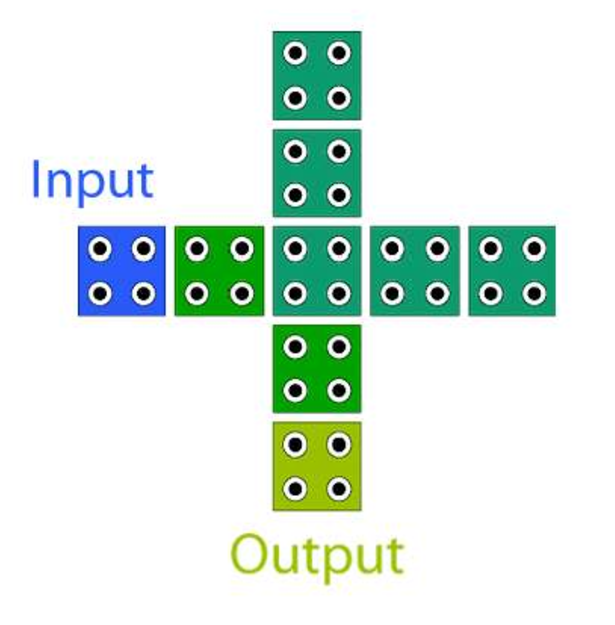
****

**Overview of Crossbar Method:**

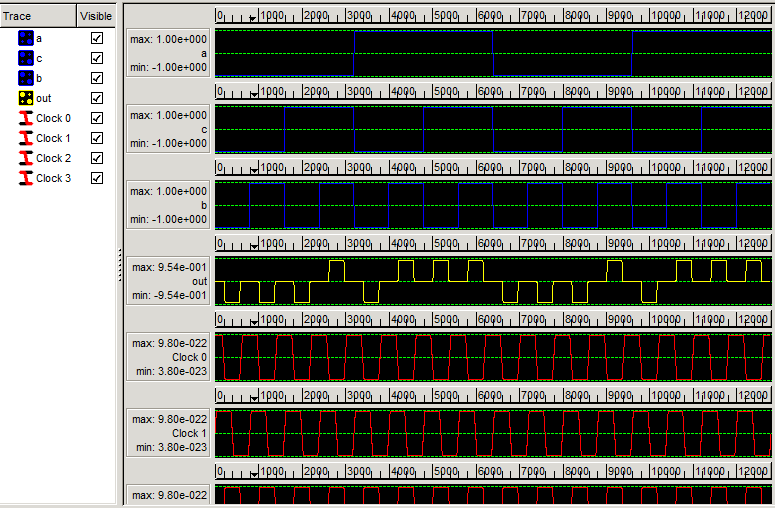
This paper proposes an automated methodology for designing combinational and sequential circuits using a programmable QCA crossbar architecture. The methodology addresses key design challenges, particularly the issue of compatibility between different QCA circuits and the significant interconnection overhead, which often exceeds the size of the circuits themselves. These issues arise from the absence of universal design rules. The proposed methodology leverages the fundamental design rules of the programmable QCA crossbar architecture and introduces universal QCA structural blocks to design combinational logic circuits. Additionally, it resolves signal timing and robustness issues by handling clock zone partitioning.

For sequential logic circuits, the methodology enhances the basic set of QCA structural blocks with a memory element block. This enables the design of memory cells on the programmable QCA crossbar architecture, which are essential for the development of the methodology. These memory cells support the creation of 2n-bit memories and provide programmability. This allows the same QCA circuit to function as either a memory cell or a processing unit. By exploiting the features of the programmable QCA crossbar architecture, both memory and logic blocks are integrated into the same crossbar to design sequential logic circuits.

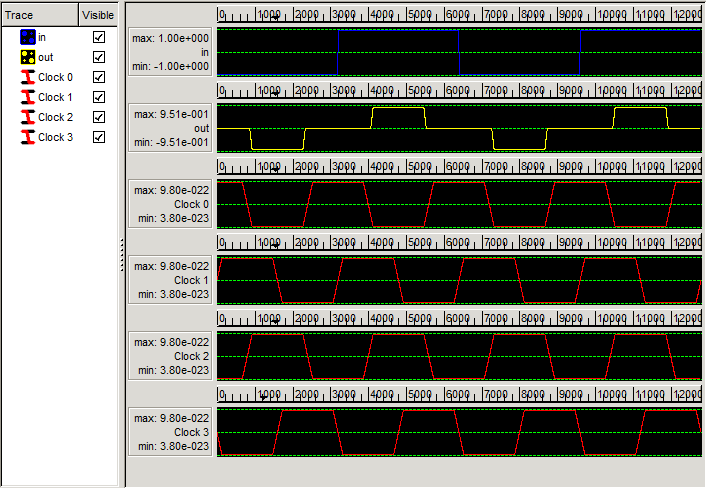
The methodology focuses on automated QCA design and does not address clocking schemes. Therefore, it should not be compared with previously introduced clocking schemes, as fixed distribution clocking offers several advantages for the design and fabrication of QCA circuits.

****  ****

1. **(b)**



**(c)**

****

**(d)**

**Fig 2. a) Crossbar Majority Gate, b) Crossbar NOT Gate (inverter)**

**c) Crossbar Majority Gate Simulation Results**

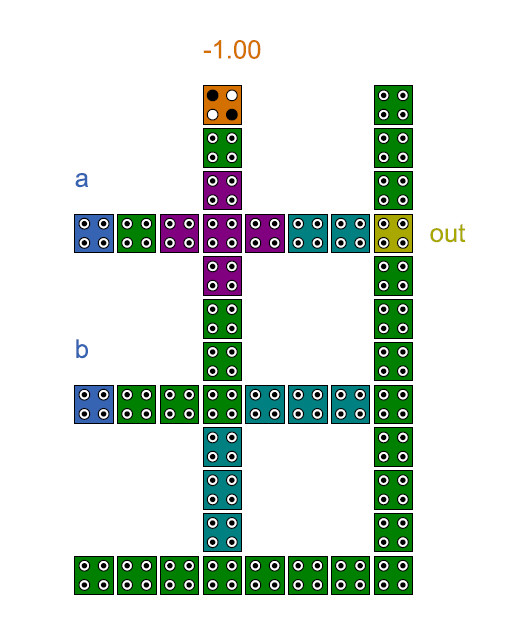
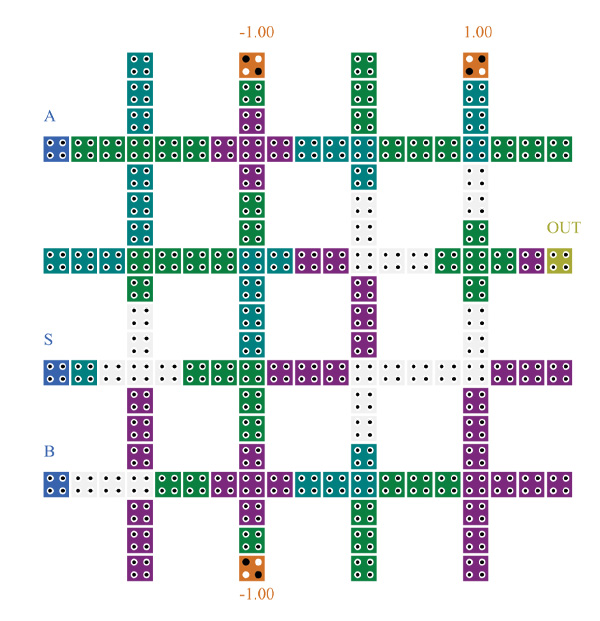
**d) Crossbar Invertor Simulation Results**

In the suggested design methodology, data flows from the left side to the right side of the circuit, facilitated by effective management of clock zone partitioning. This partitioning is essential in the design of QCA circuits, as it guarantees both stability and regulated information flow. The most suitable clocking technique for this task is adiabatic switching, which modifies the state of electrons within each quantum-dot cell. Electrons can be transitioned to either a neutral state or to one of two distinct logic states, influenced by the polarization of adjacent cells. Within each clock zone, all quantum-dot cells are managed by a unified clock signal, allowing seamless information transfer to adjacent cells in the following clock zone. This phased management promotes synchronized and effective data transmission throughout the QCA circuit.

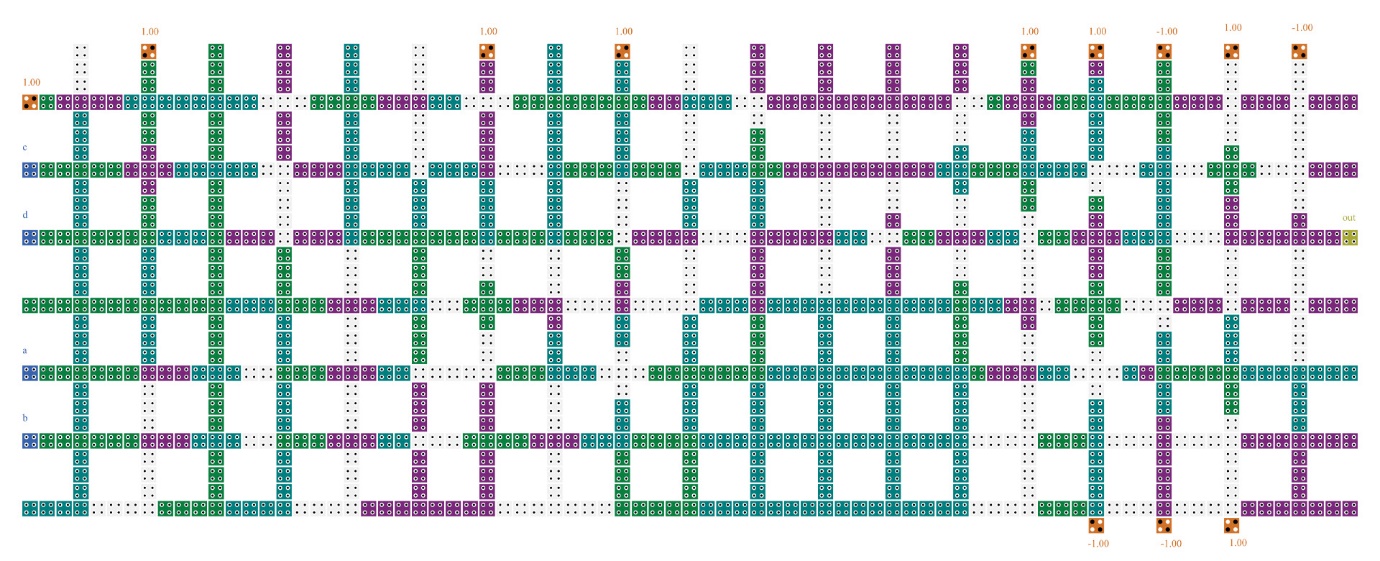
In this approach, blocks are positioned within the crossbar area, with their placements influenced by the quantity inputs they possess. These blocks fall into two distinct groups: those that carry out fundamental logic functions and those facilitate signal routing. This methodology also guarantees uniform delays across all blocks, which aids in synchronizing the circuit and reduces delays associated with signal routing.

In essence, the methodology reconfigures logic functions for QCA application, divides circuits into levels and sub-level, and strategically places on the crossbar using programming lines from top and bottom. This organized strategy fosters the creation of reliable efficient QCA circuits with minimal delays, tackling compatibility challenges and offering a comprehensive solution for the automated design of QCA circuits.

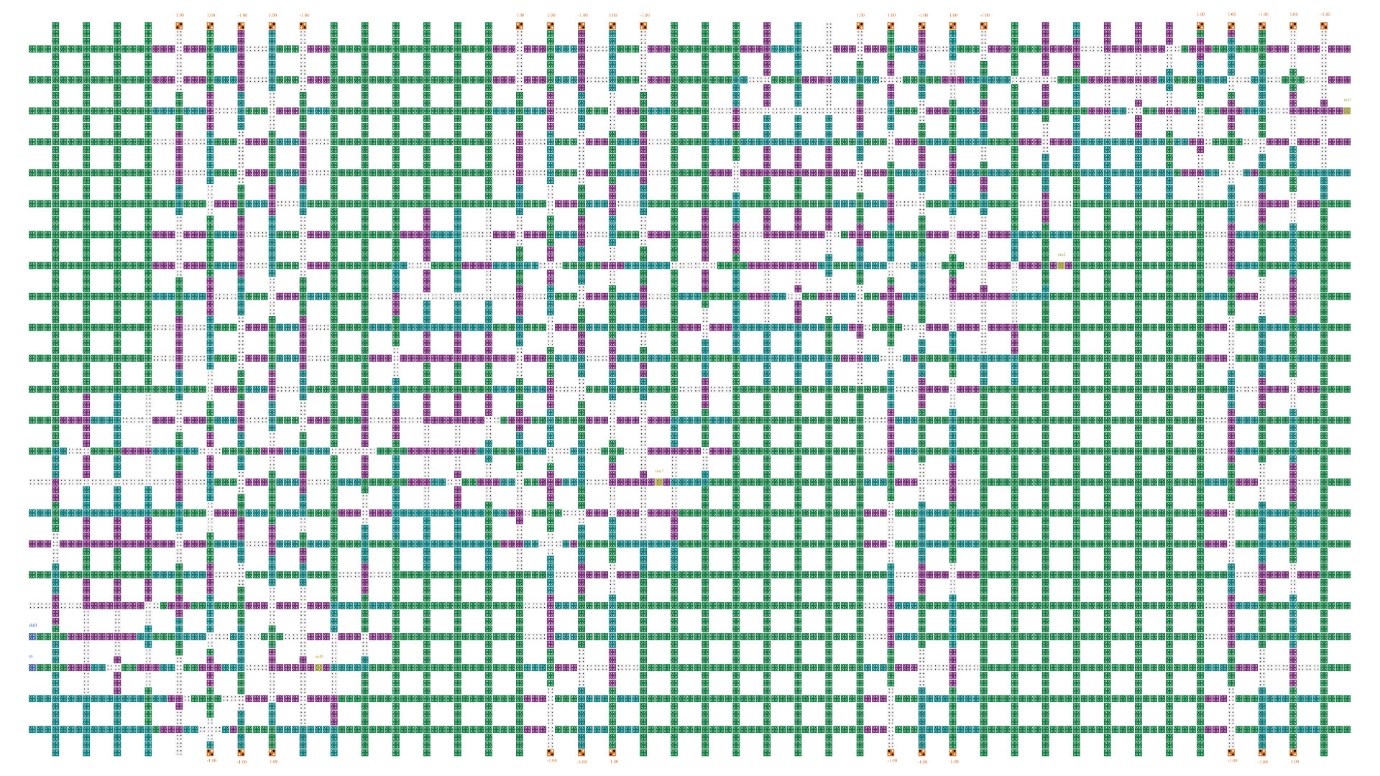
**Application of Design Methodology:**

**** 

1. **(b)**



**(c)**

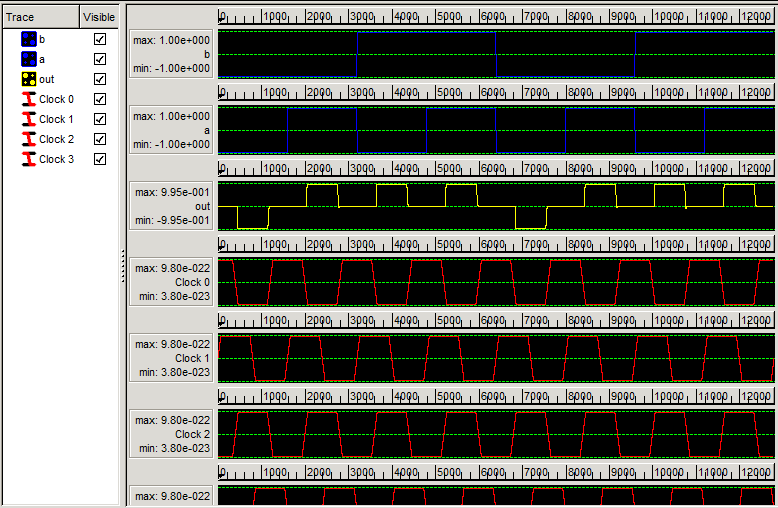


**(d)**

**Fig 3. a) 2-input block evaluating a.b b) QCA 2:1 Multiplexer**

**c) a + b + (c+d) QCA circuit with memory element**

**d) 4-bit shift register QCA Circuit**

****

**(a)**

****

**(b)**

****

**(c)**

****

**(d)**

**Fig 4. a) 2-input block evaluating a.b simulation results**

**b) QCA 2:1 Multiplexer simulation results**

**c) a + b + (c+d) QCA circuit with memory element simulation results**

**d) 4-bit shift register QCA Circuit simulation results**

**AUTOMATED QCA LAYOUTS GENERATION:**

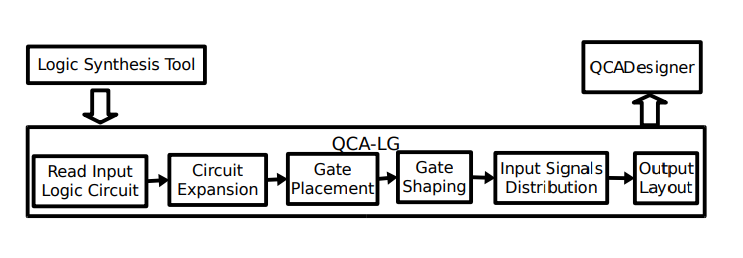
The QCA Logic Generator (LG) tool, developed in C++, generates QCA blocks based on the input provided. It accepts .lsi, .gate, and .vhd files as input and produces .qca files. While the tool operates via a Command Prompt User Interface (UI), its simplicity ensures user-friendliness.

However, only a limited number of circuits have been successfully generated using this tool so far, highlighting a significant area for improvement.

Based on the circuit design methodology, we enhanced this tool to automate QCA circuit design. Users input the desired logic function, and the tool generates a **.qca** file compatible with QCADesigner. The synthesis process includes:

1. **Block Selection**: Analyze the logic function and select the required QCA blocks.
2. **Position Definition**: Divide the circuit into levels/sub-levels and position blocks based on feedforward logic.
3. **Wire Routing**: Add binary wires to connect block outputs to subsequent inputs, progressing from primary inputs to outputs.
4. **Clocking**: Divide the circuit into clock zones from left to right, ensuring stability and adherence to constraints.
5. **Cell Placement**: Place quantum-dot cells at crossbar positions and clock zones as determined in previous steps.

This tool enables efficient and automated QCA circuit layout generation.



To generate a netlist based only on the components supported by the design, we've created libraries containing just those components. These libraries include majority gates, NOT gates, and 2-input AND, OR, NAND, and NOR gates. Internally, the circuits are represented as directed graphs and stored in a hash table, where each object represents a gate in the circuit or a primary input.

**Generation of Input Files for QCA LG Generator:**

To generate the required **.lsi**, **.vhd**, and **.gate** files for the QCA LG Generator, specific tools and workflows are necessary:

**1. Generating .lsi Files**

* **Tools Used**: Synopsys Synthesis Tool (Base Paper) or Yosys (Alternative).
* **Challenges with Synopsys**:  
  The base paper utilized the Synopsys Synthesis Tool, which is commercially available and lacks a student license option.
* **Alternative Solution - Yosys**:  
  Through extensive research, I identified **Yosys**, an open-source synthesis tool, as an alternative. Yosys can synthesize **Verilog (.v)** files and export the Netlist in formats like **.blif** and **Verilog Netlist**.
* **Conversion to .lsi**:  
  Using a Python-based external converter, we can transform the **.v** file generated by Yosys into the required **.lsi** format.

**2. Generating .gate Files**

* **Tools Used**: MVSIS (and its successor, SIS).
* **Workflow**:
  + Input the **.blif** file (exported from Yosys) into MVSIS.
  + Use MVSIS to generate the **.gate** file.
* **Challenges with MVSIS**:  
  MVSIS has undergone updates, and the usage of the **qca.genlib** library is no longer functional. This creates limitations in generating proper **.gate** files.

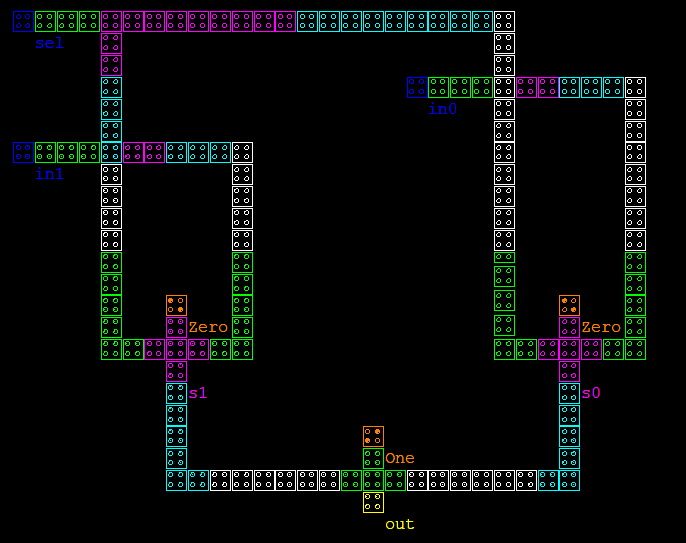
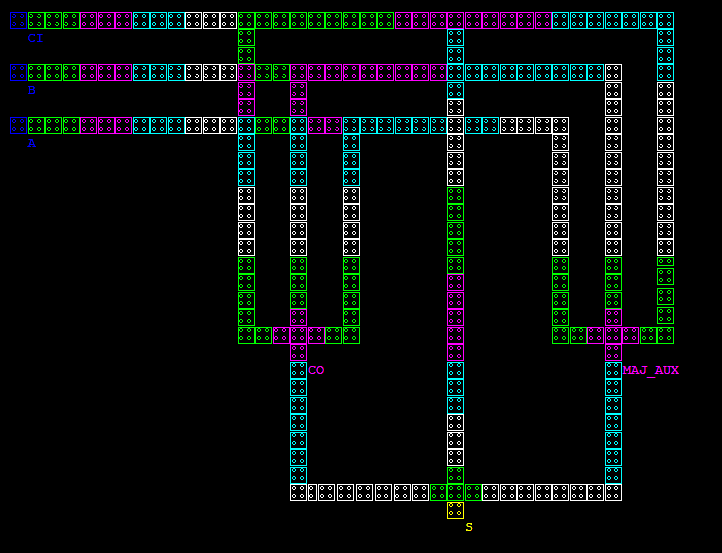
**3. Python Script Development**

* To streamline the process, I developed a Python script capable of generating **.lsi**, **.vhd**, and **.gate** files.
* **Current Limitations**:
  + The generated files have errors and inconsistencies.
  + The script is currently limited to handling **single-gate equations**.

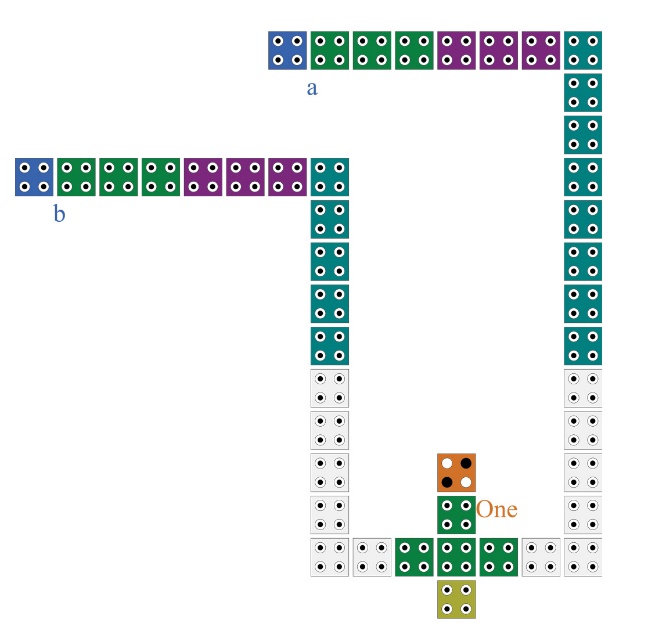
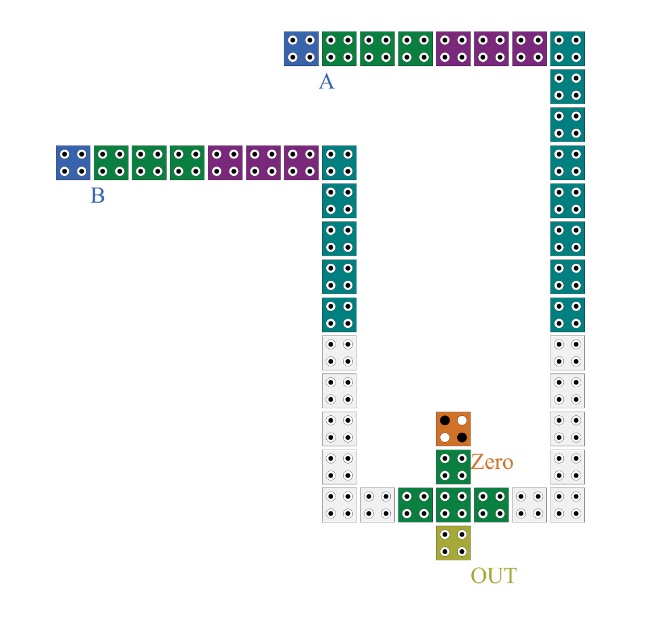
**4. Documentation and Installation**

* I created a detailed instructional video covering the installation and usage of **Yosys** and **MVSIS** tools, including workarounds for compatibility issues.

**Some of the Automated QCA Circuits:**

** **

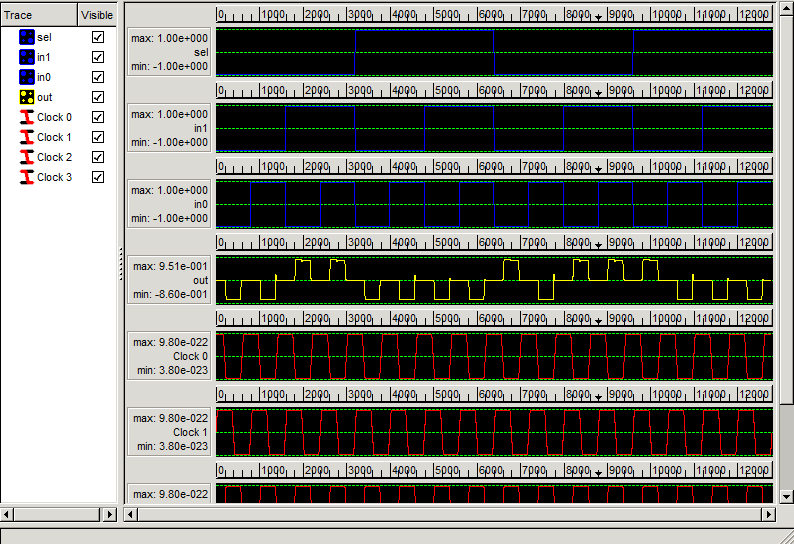
* + 1. **(b)**

****

**(c) (d)**

**Fig 5. a) Automated Mux Circuit b) Automated Adder Circuit**

**c) Automated AND Circuit d) Automated OR Circuit**

****

**(a)**

****

**(b)**

****

**(c)**

****

**(d)**

**Fig 6. a) Automated Mux Circuit Simulation Results**

**b) Automated Adder Circuit Simulation Results**

**c) Automated AND Circuit Simulation Results with errors and inconsistencies**

**d) Automated OR Circuit Simulation Results with errors and inconsistencies**

**Conclusion:**

This paper addresses the absence of universal design methodologies and architectures for Quantum-dot Cellular Automata (QCA) technology, which has hindered the development of software design tools for large QCA circuits. To overcome this, the authors propose an automated design methodology utilizing a generic programmable QCA cell crossbar architecture to implement any Boolean logic function. The effectiveness of this approach is validated through the design of various Boolean logic circuits and extended to sequential circuits, with physical simulations performed using the QCADesigner tool.

The QCA-LG tool is an integral component of this methodology, capable of generating layouts for combinational circuits in QCA. It accepts standard netlist formats such as LSI and Gate to produce QCADesigner-compatible layouts. However, the tool’s current scope is limited to small circuits. Challenges remain in optimizing gate placement for medium and large circuits that include more complex components beyond majority gates, highlighting the need for further research to improve efficiency and scalability.

To support this process, alternative tools like **Yosys**, an open-source synthesis tool, were explored to overcome the limitations of commercial software like **Synopsys**. Yosys synthesizes Verilog files to Netlist formats like **.blif**, which can be used to generate **.lsi** and **.gate** files through additional workflows. While the base paper uses **MVSIS** for generating **.gate** files, it has been found to have limitations, especially with the **qca.genlib** functionality, which does not work as expected in the updated version of MVSIS. Although attempts were made to find an alternative to MVSIS, no fully effective solution was identified.

While progress has been made in automating the generation of these files, the workflow still faces challenges in compatibility, output accuracy, and handling complex logic functions. Further refinements in tool integration, script functionality, and gate placement optimization are necessary to enhance the generation process and expand the applicability of the QCA-LG tool. Additionally, addressing the limitations of **MVSIS** or identifying a more reliable alternative remains a key area for future research.

**Software or Tools Used:**

**QCADesigner:** [**https://github.com/kwalus/QCADesigner/blob/master/release/QCADesigner-2.0.3-setup-gtk.exe**](https://github.com/kwalus/QCADesigner/blob/master/release/QCADesigner-2.0.3-setup-gtk.exe)

**QCA-LG:** [**https://github.com/TiagoTT/qca-lg**](https://github.com/TiagoTT/qca-lg)

**Yosys:** [**https://github.com/YosysHQ/yosys**](https://github.com/YosysHQ/yosys)

**MVSIS:** [**https://ptolemy.berkeley.edu/projects/embedded/mvsis/software.html**](https://ptolemy.berkeley.edu/projects/embedded/mvsis/software.html)

**References:**

[1] C. Lent, P. Tougaw, W. Porod, and G. Bernstein, “Quantum cellular automata,” Nanotechnology, vol. 4, no. 1, pp. 49–57, 1993.

[2] Orestis Liolis, Vassilios A.Mardiris, Ioannis G. Karafyllidis, Sorin Cotofana and Georgios CH. Sirakoulis, “Methodology for Automated Design of Quantum-Dot Cellular Automata Circuits”, vol 4, pp. 162-171, 2023.

[3] K. Walus, V. Dimitrov, G. Jullien, and W. Miller, “QCADesigner: A CAD tool for an emerging nano-technology,” Micronet Annual Workshop, 2003, web: <http://www.qcadesigner.ca>.

[4] Tiago Teodosio and Leonel Sousa, “QCA-LG: A tool for the automatic layout generation of QCA combinational circuits”, web: <http://web.ist.utl.pt/ttt/qca-lg/index.html>

[5] C. Wolf, J. Glaser. Yosys – “A Free Verilog Synthesis Suite” In *Proceedings of Austrochip 2013*, web: <https://yosyshq.net/yosys/>