Designing Cellular Automata Structures using Quantum-dot Cellular Automata

Mayur Bubna, Subhra Mazumdar, Sudip Roy and Rajib Mall
Department of Computer Sc. & Engineering
Indian Institute of Technology, Kharagpur-721 302, India
{ mayur, subra , sudip, rajib }@ cse.iitkgp.ernet.in

***Abstract-* Quantum-dot Cellular Automata (QCA) is a promising, emerging nanotechnology based on single electron effects in quantum dots and molecules. While many logic implementations based on QCA devices have been proposed in literature [6, 7, 8], the inherent cellular structure of QCA cells make it a natural candidate for Cellular Automata (CA) implementation. CA offers regularity and modularity to the design which helps to mitigate the susceptibility of QCA cells to manufacturing defects. Also, pipelining is an inherent property of QCA computation which is essential for CA operation. This work is first reported work to the best of our knowledge, towards realization of CA structures using QCA logic cells. We give detailed schematic of some typical CA rules implemented using QCA. Also, QCA implementation of a Programmable Cellular Automata (PCA) has been developed. A pseudo-random sequence generator is developed using the proposed PCA.**

Keywords: Quantum-dot Cellular Automata, logic design, cellular automata.

I. INTRODUCTION

Nanotechnology is an emerging area of interest which offers alternative design technologies like carbon nanotubes, quantum dot structures, molecular devices and microfluidic biochips. Quantum-Dot Cellular Automata (QCA) [4,10] is an emerging paradigm which allows operating frequencies in range of THz and device integration densities about 900 times more than the current end of CMOS scaling limits, which is not possible in current CMOS technologies. It has been predicted as one of the future nanotechnologies in Semiconductor Industries Association's International Roadmap for Semiconductors (ITRS) [3]. QCA encodes information in the configuration of electrons within the QCA cell, and relies on charge interactions to enable the transmission and processing of information. Logical operations and data movement are accomplished via Coulombic interaction between neighbouring QCA cells rather than electric current flow. QCA circuits could be clocked at extremely high frequencies (1-10 THz), potentially leading to circuits with densities that are one or two orders of magnitude beyond what end-of-the-curve CMOS can provide [1], and dissipate very little power [2].

Many logic devices based on QCA gates have been proposed in literature [6,7,8]. All of them try to map the QCA cells to realize CMOS based structures like AND, OR etc. For example Niemer [7] have proposed a 4 bit-slice ALU designed using QCA cells. But QCA cells are clocked elements and offer inherent pipelining. Cellular architectures are more suitable to QCA because of inherent pipeline structure which CA structures offer. Also, regular layout of CA structures offers higher integration densities with lower power consumption. Further, various kinds of manufacturing defects occur in QCA manufacturing technology processes. The basic defects include displacement errors, rotated cells, missing cells, fixed cells (due to stray charge), bad clocking wires, and bad circuit input/output [5,9]. Cellular Automata offers modular layout which enhances manufacturability.

We give a model of one dimensional, null-boundary, linear CA using Rule 60 and give schematics and QCA layout implementations. Also, we give a model of Programmable Cellular Automata (PCA) which can be modified to accommodate a variety of complementary, additive, null-boundary CA rules by passing appropriate control inputs. The proposed PCA has been modified to show a four cell with rule 90/150/165/105 implemented using QCA based gates and latches. Detailed schematic and layout have been provided for all the implementations. This work is the first approach to the best of our knowledge which utilizes the cellular structure of QCA cells to devise Cellular Automata structures realizing various state transition rules. The proposed model can be easily modified to implement other one dimensional as well as two dimensional Cellular Automata structures.

II. QUANTUM-DOT CELLULAR AUTOMATA

We briefly describe the various QCA gates and computation mechanism using QCA cells.

A. *Basic QCA cell:*

QCA and the QCA cell was first introduced by Prof. C. S. Lent at the University of Notre Dame [3]. QCA information processing is based on the Coulombic interactions between many identical QCA cells.

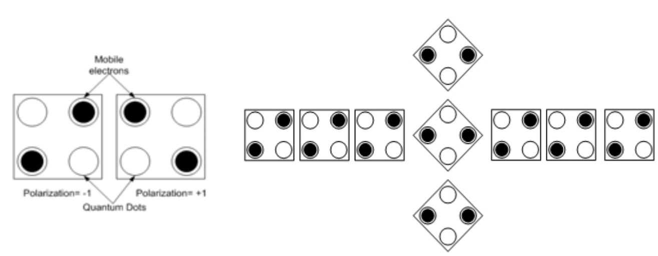

Fig.1. (a) Polarizations of QCA cell and (b) Two types of QCA wires.

Each QCA cell is constructed using four electronic sites or dots coupled through quantum mechanical tunneling barriers. The electronic sites represent locations that a mobile electron can occupy. The cells contain two mobile electrons which repel each other as a result of their mutual Coulombic repulsion, and, in the ground state, tend to occupy the diagonal sites of the cell. These lead to two polarizations of a QCA cell, denoted as +1 and -1 respectively. Binary information can be encoded in the polarization of electrons in each QCA cell. Thus, logic 0 and logic 1 are encoded in polarization -1 and +1 respectively. Fig. 1(a) shows the two possible polarizations of a QCA cell. QCA wires can be either made up of 90^0 cells or 45^0 cells. 45^0 cells are used for coplanar wire crossings (Fig. 1(b)).

B. QCA Computation:

Unlike standard technologies, where metallic interconnects are used to connect transistors together, QCA cells act as both the switching device as well as interconnects. This difference has a significant impact on optimizing QCA computing architectures and the latency of the circuits. QCA computation proceeds by the orientation of cells based on the polarization of the neighbouring cells. The basic gates for computation in QCA are the majority gate 'M' and the inverter. The majority gate computes the function M=AB+BC+CA and outputs the majority value of its three inputs. By fixing the input polarization of one input cell to -1 or +1, i.e. logic value 0 and 1 respectively, AND and OR gates can be computed. Fig.2 (a) and 2(b) shows a Majority and an inverter gate.

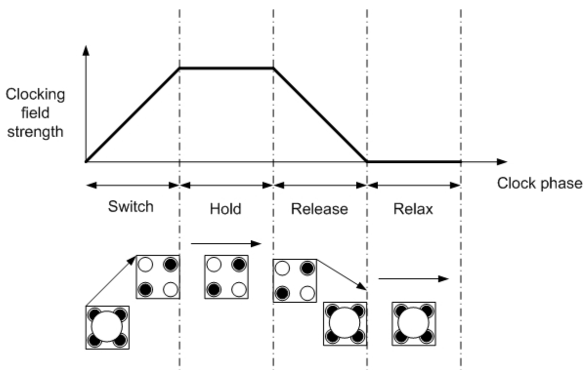

Fig.3. The four clocking phases and the assosciated QCA cell polarizations.

C. QCA clocking:

The clock in QCA is multi-phased. Individual QCA cells are not timed separately. A group of QCA cells can be divided into sub-groups that offer the advantage of multi-phase clocking and pipelining. For each sub-group of QCA cells, a single potential modulates the inter-dot barriers in all of the cells. This clocking scheme allows one sub-group of cells to perform a certain calculation, have its state frozen by the raising of its interdot barriers, and have the output of that sub-group of cells act as the input to a successor sub-group (i.e. clocking sub-group 1 can act as input to clocking sub-group 2). During the calculation phase, the successor group is kept in an unpolarized state so it does not influence

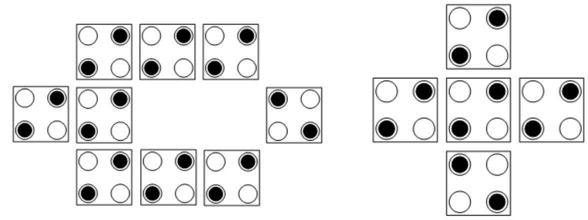

Fig.2. (a) a Majority gate (b) an inverter

the computation. Each of the four clocking sub-groups corresponds to one of four different clocking phases. Neighboring sub-groups of QCA cells concurrently receive neighboring clocking phases.

Fig. 3 shows the four clocking phases and the assosciated polarization of electrons in these phases. During the first clock phase, the *switch phase*, QCA cells begin unpolarized and their interdot potential barriers are low. The barriers are then raised during this phase and the QCA cells become polarized according to the state of their driver (i.e. their input cell). It is in this clock phase that the actual computation (or switching) occurs. By the end of this clock phase, barriers are high enough to suppress any electron tunneling and cell states are fixed. During the second clock phase, the *hold phase*, barriers are held high so the outputs of the subgroup can be used as inputs to the next stage. In the third clock phase, the *release phase*, barriers are lowered and cells are allowed to relax to an unpolarized state. Finally, during the fourth clock phase, the *relax phase*, cell barriers remain lowered and cells remain in an unpolarized state [4].

III. CELLULAR AUTOMATA

Cellular Automata was proposed by J. von Neumann [11] as a cellular space with self-producing configurations involving 5-neighbourhood cells, each having 29 states. Various applications of Cellular Automata have been proposed in literature, for example modeling growth processes, image processing, computer architectures, language recognizers, error correcting codes etc. Many applications of CA based on local neighbourhood have been utilized for various VLSI applications like synthesis of testable finite-state machines, pseudoassosciative memory, Built-in Self Test (BIST) etc.[12]

A. CA state transition rules:

The following notations have been used to characterize CA transition rules:

i : The position of a cell in the one-dimensional array;

t : the time step;

$Q_i(t)$: the output of the ith cell at the tth time step; and

$Q_i(t+1)$: the output state of ith cell at the $(t+1)$th time step;

The next-state function (transition) for a three-neighborhood CA cell can be expressed as follows:

$Q_i(t+1) = f\,[Q_i(t), Q_{i+1}(t), Q_{i-1}(t)]$

where, f denotes the local transition function realized with a combinational logic, and is known as the "rule" of the CA.

N_s:	111	110	101	100	011	010	001	000	
Next state:	0	0	1	1	1	1	0	0	(rule 60)
Next state:	0	1	0	1	1	0	1	0	(rule 90)
Next state:	1	0	0	1	0	1	1	0	(rule 150)
Next state:	1	0	1	0	0	1	0	1	(rule 165)
Nexr state:	0	1	1	0	1	0	0	1	(rule 105)

Table 1: Rule notation for a CA

If the next-state function of a two-state, three neighbourhood cell is expressed in the form of a truth table, then the decimal equivalent of the output is called the rule number for the cell [12].

In Table 1, $\mathbf{N_s}$ denote the neighbourhood state of a cell. The top row gives all possible eight states of the three neighbouring cells (the left neighbour of the *i*th cell, the *i*th cell itself, and its right neighbour) at the time instant *t*. The second and third rows give the corresponding states of the *i*th cell at the time instant *t+1* for the two illustrative CA rules. The corresponding combinational logic for the above rules can be specified as:

Rule 60: $Q_i(t+1) = Q_{i-1}(t) \oplus Q_i(t)$

Rule 90: $Q_i(t+1) = Q_{i-1}(t) \oplus Q_{i+!}(t)$ etc.

where $\oplus$ denotes XOR (that is, addition modulo-2). For example in Fig. 4, a single CA cell is depicted whose output depends on the state values from it's left and right neighbors, and depending on the switch open or closed computes the rule 90 or 150.

B. Group CA characterization:

A group of CA cells form a Group CA. The *n*-bit global state of a CA at time *t* can be denoted as vector $S^{(t)}$. The states of a CA during each discrete time step is successively sampled to form a stream < $S^{(0)}$, $S^{(1)}$, $S^{(2)}$, …….>. This approach qualifies the CA as an iterative PRNG. Here we consider only linear/additive CA rules, more specifically only rules 90, 150, 165 and 105 shown in Table 1. We can then define a state transition matrix for a CA, denoted as **T**. It is an *n*-by-*n* square matrix, with each row representing the state transition neighborhood dependencies for each cell, an entry of "1" means dependency and "0" otherwise. The next global state vector is then calculated uniquely by

$$S^{(t+1)} = \mathbf{T}.\, S^{(t)}$$

For example, if the initial state of a 4 cell CA is $S^{(0)} = [\ 1\ 1\ 0\ 0\ ]^T$, then the next state will be $S^{(1)} = \mathrm{T}.\, S^{(0)} = [1\ 1\ 1\ 0]^T$, with the state transition matrix **T** defined as:

$$\mathbf{T} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

All arithmetic is performed over *GF*(2). Here we only consider CA with null boundary conditions where the leftmost/rightmost CA cells receive a fixed "0" input from the leftmost and rightmost "supposed" neighbour respectively.

C. Programmable Cellular Automata (PCA):

Programmable Cellular Automata (PCA) allows spatial and temporal variation in the state transition rules within a CA, according to some external control scheme. This equates to dynamically changing the state transition matrix T. Through an appropriate selection of state transition rules and the control signal wirings, a number of rules can be programmed into the operation of the PCA. Fig. 5 shows an example PCA cell with various control input signals. Depending on the values of control signals, the PCA cells realize either 2-input or 3-input, complementary or non-complementary CA rules.

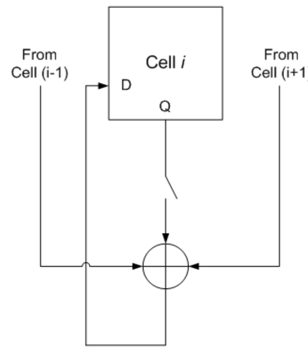

Fig. 4. A simple CA cell showing rule 90/150

The combination logic defining the state transition behaviour is the 3-input XOR gate which receives inputs from the state values of left neighbor, right neighbor and the cell itself. The inputs are not directly fed into the XOR gate but are multiplexed using switches as shown. This enables the PCA cell to be initialized to any desired state by suitably choosing the control input to the switches and the initialization signals. The output is finally fed back to a D Flip-Flop which stores the state of a cell for computation in the next clock cycle.

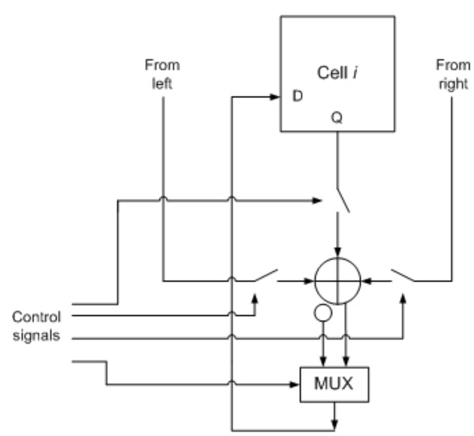

Fig. 5. A PCA cell with various control signals

IV. QCA IMPLEMENTATION OF CA

A. *QCA implementation of Rule 60 and PCA:*

The schematic of a single CA cell for Rule 60 is shown in Fig. 6. **Input1** and **Input2** are used for initializing the CA cell to a particular value. **S1** and **S2** are the select inputs to the 2-to-1 MUX. **S1** (or **S2**) selects whether the inputs to the 2-input XOR gate is the initialization value, **Input1** (**Input2**) or feedback value $\mathbf{Q_{i-1}(t)}$ ($\mathbf{Q_i(t)}$). **D0**, **D1** etc. are latches which are made of QCA cells and is equivalent to wires in CMOS logic. Thus, when **S1=1**, **S2=1** circuit behaves in initialization mode, and CA is initialized with input values. When **S1=0**, **S2=0**, the circuit behaves in operation mode with feedback and the states of the cells evolve according to Rule 60. Since QCA operates in accordance with an external clock, the storage element in a CA is realized by the feedback latch in the circuit. The latency of the circuit is 2T, where T is the one clock period consisting of four clock phases. This CA cell can be replicated to generate the whole CA structure. Since the CA is a null boundary CA, the leftmost and rightmost group CA cell shall always have left input and right input values permanently fixed to 0. The layout of circuit generated using QCADesigner [8] is shown in Fig. 7. The coloured regions indicate the QCA cells divided into four clocking phases.

Next we implement a 4-cell PCA with rule set 90/150/165/105. Fig. 8 gives the schematic of a single PCA cell which can be configured to various rules based on the inputs to the MUX. In this case, the combinational logic is a 3-input XOR gate which based on the value of signal **B**, implements complementary rule pairs 150/105 (**B=0**) and

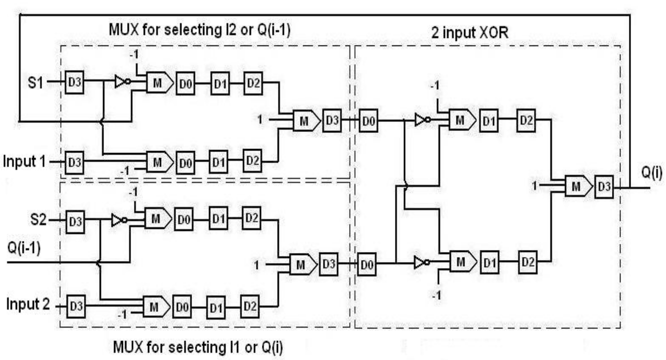

Fig. 6. QCA Schematic of a PCA cell with rule 60

90/165 (**B=1**). **Select0** and **Select1** are select inputs to a MUX, which are used to initialize the XOR gate or set the PCA cell for operation in feedback mode, as explained for Rule 60 CA cell above. **Input0** and **Input1** are used to initialize the CA cells. **A** input selects which of the complementary rules is implemented for each CA rule pair. For example, for **Select0=0**, **Select1=0**, and **B=0**, setting **A=0**, selects rule 90, while setting **A=1**, CA implements the rule 150. Table 2 shows the various possible PCA cell configurations. Fig. 9 shows the full layouts for 4 cells of an example PCA cell generated using QCADesigner tool [8].

Switch B	Switch A	Rule implemented
0	0	90
0	1	150
1	0	165
1	1	105

Table 2: Control switch configurations for PCA

B. *QCA implementation of a Pseudo-random Number Generator:*

The evolution of states of PCA cell behave like a pseudo-random number generator. When the states of the group of CA cells, < $S^{(0)}$, $S^{(1)}$, $S^{(2)}$, …….>, during each time step is sampled, it forms a pseudo-random number stream. It was shown in [12] that an exhaustive LFSR (linear feedback shift register) and an exhaustive CA are isomorphic to each other. The assosciated characteristic polynomial can be obtained by constructing the matrix [T]+x[I], where [I] is the identity matrix and taking its determinant. For the PCA rule 90/150/165/105, the assosciated characteristic polynomial was found out to be x^4-$3x^2$+1. Thus, this PCA behaves like an LFSR. The phase shift properties of this pseudo-number sequence has been studied extensively in [12]. By initializing the PCA with different seed values, different types of pseudo-random sequences can be generated.

V. CONCLUSION

In this paper, we have designed various types of linear and additive CA rules using QCA cells. The design of a Programmable CA (PCA) was shown using QCA cells and detailed layout and schematics were generated using QCA cells. PCA was configured to a pseudo-random number generator and shown to be equivalent to an LFSR. Various other kinds of applications of CA based on QCA can be constructed. Future work includes designing control structures using QCA like testable FSM design, BIST etc. Also, reversible implementations of QCA structures [13] should be implemented which reduces the switching power dissipation not possible in irreversible implementations.

REFERENCES

[1] M.T. Niemier, “The Effects of a New Technology on the Design, Organization, and Architectures of Computing Systems,” Ph.D. Dissertation.

[2] J. Timler and C.S. Lent, “Power gain and dissipation in quantum-dot cellular automata”, J. of App. Phys., 91, 2002, p.823-831.

[3] International Technology Roadmap for Semiconductors, International Technology Roadmap for Semiconductors (ITRS) 2004, http://public.itrs.net, 2004.

[4] Tougaw, P. and Lent, C., Logical devices implemented using quantum cellular-automata. Journal of Applied Physics. v75. 1818-1825.

[5] G. Schulhof, K. Walus, G. A. Jullien, “Simulation of random cell displacements in QCA”, Journal of Emerging Technologies, Vol. 3, No.1, Article no. 2, 2007.

[6] Walus, Konrad, Graham A. Jullien, “Design tools for an emerging SoC technology : Quantum-dot cellular automata”, Proceedings of IEEE, Vol. 94, Issue 6, pp. 1225-1244.

[7] Designing Digital Systems in Quantum Cellular Automata M.T.Niemier, Master’s Thesis, University of Notre Dame (2004)

[8] K. Walus, T. Dysart, G.A. Jullien, R.A. Budiman, “QCADesigner: A Rapid Design and Simulation Tool for Quantum-Dot Cellular Automata”, IEEE Transactions on Nanotechnology, vol. 3, no. 1, pp. 26-31, March 2004.

[9] Timothy J. Dysart. "Defect Properties and Design Tools for Quantum Dot Cellular Automata." Master's Thesis, 2005.

[10] G.L. Snider, A.O. Orlov, R.K. Kummamuru, R. Ramasubramaniam, I. Amlani, G.H. Bernstein, C.S. Lent, J.L. Merz and P. Wolfgang, “Quantum-dot cellular automata: introduction and experimental overview”, Proceedings of the First IEEE Conference on NANOTECHNOLOGY pp. 465-470 (2001)

[11] S. Wolfram, “Statistical Mechanics for Cellular Automata”, *Rev. Mod. Phys.*, Vol. 55, July 1983, pp. 601-644.

[12] Additive Cellular Automata, Theory and Applicatons-Volume 1. P. P. Chaudhuri, D. R. Chowdhury, S. Nandi, S. Chattopadhyay. IEEE Press.

[13] C. S. Lent, Sarah E. Frost, P. M. Kogge, Reversible computation with QCA,Proc. Of 2nd Conf. on Computing Frontiers, pp.403, 2005.

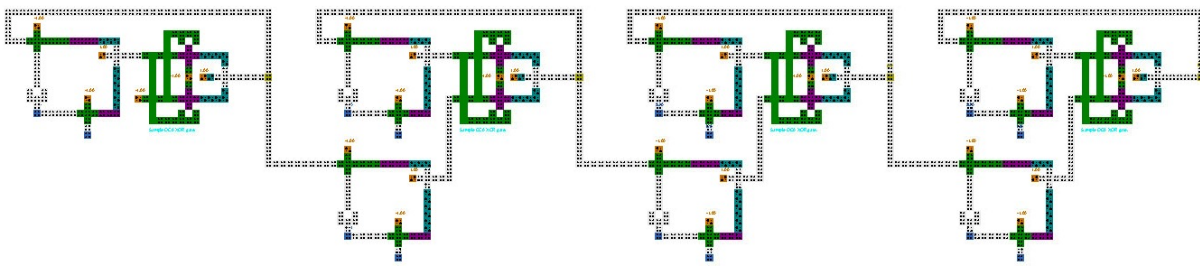

Fig. 7. Detailed layout of Rule 60.

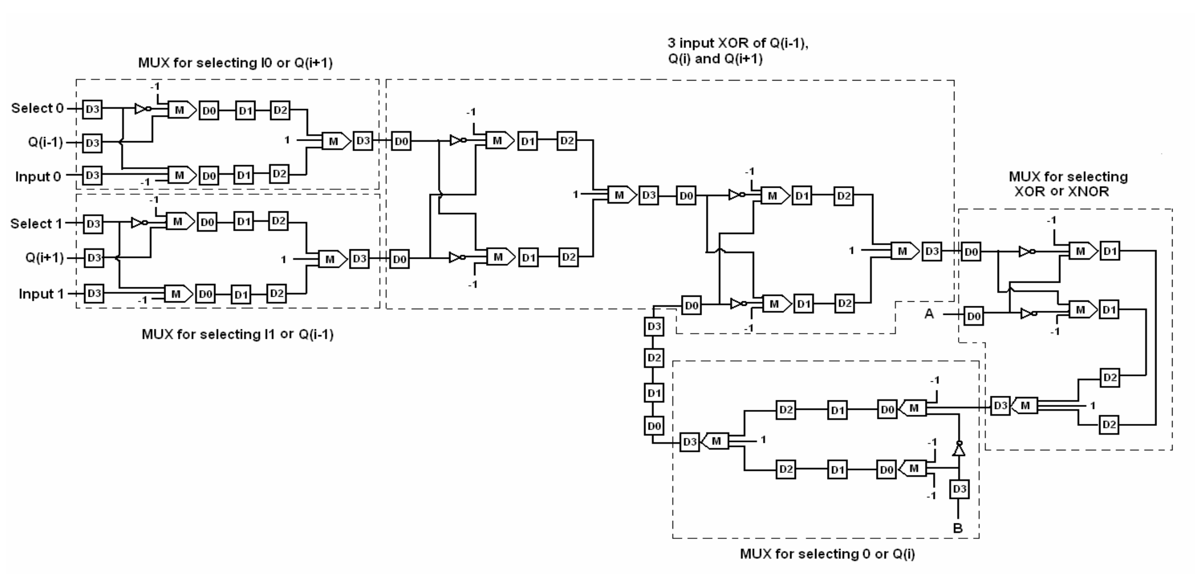

Fig. 8. Schematic of a PCA cell with rule 90/150/165/105

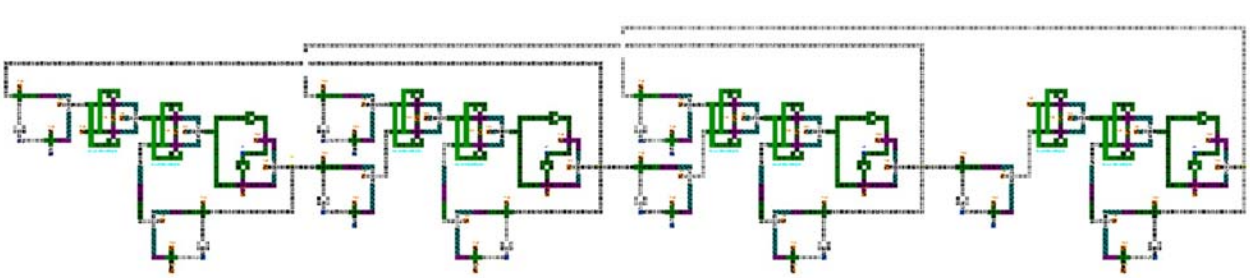

Fig. 9. Detailed layout of Rule 90/150/165/105