Modeling quantum dot devices in Cell-DEVS environment

Yuri Boiko and Gabriel Wainer
Carleton University
1125 Colonel By Drive,
Ottawa, ON, K1S 5B6 CANADA
yuri.boiko@rocketmail.com , gwainer@sce.carleton.ca

Keywords: Discrete event simulation, Cell-DEVS, quantum dot, quantum automata, cellular automata, XOR gates, majority vote gates, quantum wire

Abstract
Simulation of selected elements of Brain Machine, specifically those based on quantum dot technology and defined by quantum cellular automata, is demonstrated in Cell-DEVS environment employing CD++ toolkit. Models of quantum dot cellular devices are implemented within Cell-DEVS formal definitions. For Cell-DEVS modeling the following quantum dot based devices have been selected: quantum XOR, quantum wire and majority vote gates. Operation of models of these devices is visualized by CD Modeler animation as well as draw-log tools within Cell-DEVS environment. Test rules for functionality verification of the models are presented and verified to validate the models.

1. INTRODUCTION

Modeling and simulation of the various elements of the Brain Machine are gaining of the simulation and design research communities [1, 2], as new technological approach, such as nano-technology, is gaining its recognition as foundation of new generation of transistor-free electronics of the years to come. The benefits of the modeling and simulation techniques include efficient optimization of the designs of sophisticated quantum dot based nano-technological devices, implementation of and experimentation with of which is challenging task by itself both technologically and price-wise. DEVS (Discrete Event System Specification) methodology gained deserved recognition in modeling complex artificial systems [3, 4] by offering hierarchical approach of building complex systems from elementary blocks of sub-models, which in turn can be atomic (lowest level) or coupled (intermediate level) models by itself. The extension of DEVS formalism to the systems, which can be described by cellular automata approach, is Cell-DEVS. Herewith the implementations of the quantum dot based devices models are demonstrated using Cell-DEVS formal specifications. Herewith the CD++ toolkit is used for programming the models.

2. Cell-DEVS PROBLEM IMPLEMENTATION

2.1 Quantum Automata for modeling of quantum dot devices

In the present study Cell-DEVS formalism is employed to model operation of basic quantum dot devices, which are described by quantum cellular automata. For Cell-DEVS modeling the following quantum dot based devices have been selected: quantum XOR, quantum wire and majority vote gates. Demonstrated herewith is operation of models of these devices in Cell-DEVS environment employing CD++ tool kit. Therefore, the real system of choice here is Cellular Automata implementation of Quantum Logic Circuits, which are used in the current state-of-the-art Quantum Computing and Artificial Brain Machines. Quantum Cellular Automata (QCA) refers to a model of quantum computation, which has been devised in analogy to conventional models of cellular automata introduced by von Neumann. Important for a model of quantum cellular automata is that it should be universal for quantum computation [5, 6] (i.e. that it can efficiently simulate quantum Turing machines, or equivalently, uniform families of quantum circuits). Additional restrictions are that quantum cellular automata should be reversible and/or local unitary, and have an easily determined global transition function from the rule for updating individual cells [5]. Examples of the quantum devices can be found in [5 - 12].

2.2 Structures of Quantum Cellular Automata in Cell-DEVS

The subject of this project is implementation of the Cellular Automata for logic device based quantum dot cells. Each quantum cell contains two electrons which interact with the neighbors Coulombically. Each cell is designed having two perpendicular axes of preferential alignment of the pair of electrons in it. Such states of preferential alignment are used to encode binary bits of 0 and 1. Employing such cells, various quantum devices can be built for the digital circuits to operate at nanometer scale, which hold a promise of faster speed and reduced size. Shown below are the cellular automata's for XOR quantum gates (Fig.1) and for the 3 inputs delayed Majority Vote Gates (Fig.2).

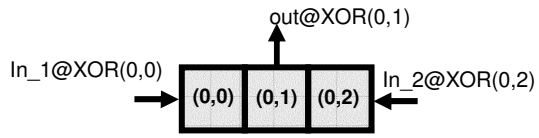

Figure 1. Quantum Cellular Automata for XOR gates.

In the cellular field there are several zones of inactive cells, which are not involved in the operation of the device, but rather are affecting indexes of the operational cells. Among operational cells there are input cells, transport cells, calculation cell and output cell.

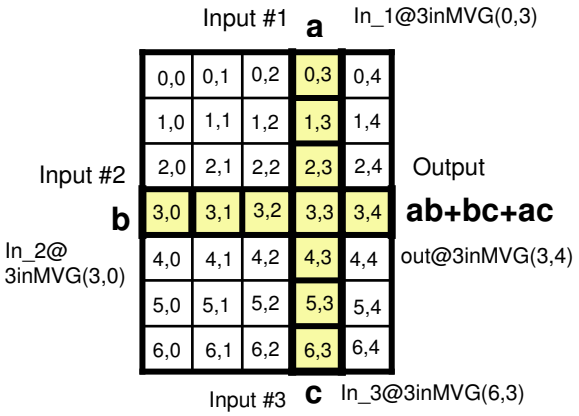

Figure 2. Quantum Cellular Automata for 3 inputs Delayed Majority Vote Gate device. Inputs are a, b and c. Output function is ab+bc+ac (modulo 2).

2.3 Properties of Quantum Cellular Automata devices (Cell-DEVS)

The properties of the models of XOR and delayed Majority Vote Gates had been verified using the "black box" testing method. Initially both models were tested by setting all inputs to the value of 1 by the following lines in the program for respective devices:

(1) (for XOR gate)

```
initialvalue : 0
initialrowvalue :  0   101
```

(2) (for 3 inputs Majority Vote Gates)

```
initialvalue : 0
initialrowvalue :  0   00010
initialrowvalue :  3   10000
initialrowvalue :  6   00010
```

Under such conditions the output of 1 is expected in both models. Obtaining 1 at the output implies that that model is likely to be functional with other external inputs. After that the external input had been introduced via external events file **.ev*. For XOR gates the rule was that output "-1" is if both inputs are equal (i.e. both to "1" or to "-1") and output "1" otherwise and calculation as well as clearing time constants set both at 100 msec:

```
[calculus-rule]
rule : -1 100 { (0,-1)=-1 and (0,1)=-1 }
rule : 1 100 { (0,-1)=1 and (0,1)=-1 }
rule : 1 100 { (0,-1)=-1 and (0,1)=1 }
rule : -1 100 { (0,-1)=1 and (0,1)=1 }
rule : 0 100 { t }
```

Calculation rules for 3 inputs Majority Vote gates were set to make the output function to be ab+bc+ac in modulo 2 and containing "-1" instead of "0" , which is needed to avoid confusion between neutral state of the cell and signal "0" (as also was the case for XOR gates):

```
[calculus-rule]
rule : 1 10 { (-1,0)=1 and (0,-1)=1 and (1,0)=1 }
rule : 1 10 { (-1,0)=1 and (0,-1)=1 and (1,0)=-1 }
rule : 1 10 { (-1,0)=1 and (0,-1)=-1 and (1,0)=1 }
rule : 1 10 { (-1,0)=-1 and (0,-1)=1 and (1,0)=1 }
rule : -1 10 { (-1,0)=1 and (0,-1)=-1 and (1,0)=-1 }
rule : -1 10 { (-1,0)=-1 and (0,-1)=1 and (1,0)=-1 }
rule : -1 10 { (-1,0)=-1 and (0,-1)=-1 and (1,0)=1 }
rule : -1 10 { (-1,0)=-1 and (0,-1)=-1 and (1,0)=-1 }
rule : 0 10 { t }
```

where calculation and clearing time constants are both set at 10 msec value.

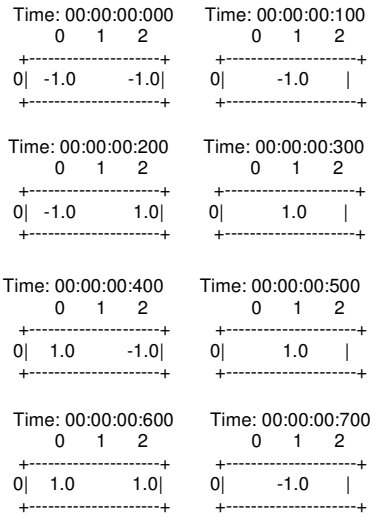

Figure 3. Operation of quantum automata XOR gates in drawlog tool of CD++.

For both models the test cases were introduced by adding different combinations of inputs to the event file (*.ev*), run the simulation (*.scp*) and check whether the outputs in the output file (*.out*) are what was expected.

The properties of the XOR gate model can be formulated as a following set of rules, which are confirmed by the data presented in Table 1 (enumeration starts from xxiii as continuation of the initial work presented in [13]):

(xxiii) signal inputs and signal outputs are connected with XOR relations;

(xxiv) delay for signal output is equal to the programmed processing time;

(xxv) each signal output is followed by clearing "0" output after additional delay of programmed clearing time.

Rule (xxiii) is confirmed in Table 1 by correspondence with the XOR relations. The rule (xxiv) is also followed as the delay for all output signals are equal to the programmed value of 100 msec. And the last rule (xxv) is confirmed by the fact, that each signal is followed by the clearing output of "0" after programmed value of clearing time (also 100 msec).

Drawlog tool in CD++ allows for graphic view of signal propagation, shown in Fig.3. Similarly, animation tool of CD++ Modeler allow visualization of the automata XOR gates operation, as it is shown in Fig.4.

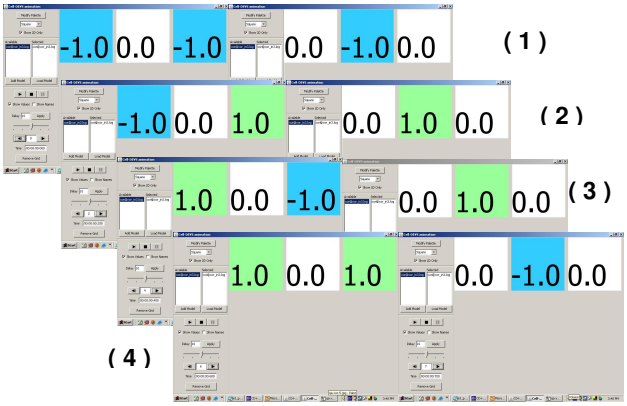

Figure 4. Visualization of the operation of quantum automata XOR gates in CD++ Modeler animation tool. Lines (1)-(4) represent various input-output combinations.

The above consideration conclusively confirms functionality and validity of the quantum automata of XOR gates.
Similar consideration has been given to the properties of the quantum cellular automata of 3 inputs Majority Vote Gates shown in Fig.2

The properties of the Delayed Majority Vote Gates model can be formulated as a following set of rules, confirmation of which is sought in Table 2:

(xxvi) signal inputs and signal outputs are connected with Majority Vote relations;

(xxvii) delay for signal output is equal to the number of steps multiplied by the programmed processing time at each step;

(xxviii) each signal output is followed by clearing "0" output after additional delay of programmed output clearing time.

Cases (1)-(4) in Table 2 provide majority of "1" at the input, so expected output value is "1" and the obtained value is also "1", as it is seen in the Table 2 in accord with the rule (xxvi). It is also seen that the output is delayed by 40 msec ,

Table 1. Test results for comparing inputs and output mapping to the expected values of the XOR gates.

#	Initiating Events (the cause)	Output result, confirming the rules
1	00:00:00:000 in1 -1 00:00:00:000 in2 -1	00:00:00:100 out -1 00:00:00:200 out 0
2	00:00:00:200 in1 -1	00:00:00:300 out 1 00:00:00:400 out 0
3	00:00:00:400 in1 1 00:00:00:400 in2 -1	00:00:00:500 out 1 00:00:00:600 out 0
4	00:00:00:600 in1 1 00:00:00:600 in2 1	00:00:00:700 out -1 00:00:00:800 out 0

Table 2. Test results for comparing inputs and output mapping to the expected values of the Delayed Majority Vote Gates.

#	Initiating Events (the cause)	Output result, confirming the rules
1	00:00:00:000 in1 1 00:00:00:000 in2 1 00:00:00:000 in3 1	00:00:00:040 out 1 00:00:00:041 out 0
2	00:00:00:100 in1 1 00:00:00:100 in2 1 00:00:00:100 in3 -1	00:00:00:140 out 1 00:00:00:141 out 0
3	00:00:00:200 in1 1 00:00:00:200 in2 -1 00:00:00:200 in3 1	00:00:00:240 out 1 00:00:00:241 out 0
4	00:00:00:300 in1 -1 00:00:00:300 in2 1 00:00:00:300 in3 1	00:00:00:340 out 1 00:00:00:341 out 0
5	00:00:00:400 in1 1 00:00:00:400 in2 -1 00:00:00:400 in3 -1	00:00:00:440 out -1 00:00:00:441 out 0
6	00:00:00:500 in1 -1 00:00:00:500 in2 1 00:00:00:500 in3 -1	00:00:00:540 out -1 00:00:00:541 out 0
7	00:00:00:600 in1 -1 00:00:00:600 in2 -1 00:00:00:600 in3 1	00:00:00:640 out -1 00:00:00:641 out 0
8	00:00:00:700 in1 -1 00:00:00:700 in2 -1 00:00:00:700 in3 -1	00:00:00:740 out -1 00:00:00:741 out 0

which is due to total of four cells distance between the input and output cells, each cell having the delay time of 10 msec, thus confirming rule (xxvii). There is also expected "0" signal 1 msec after output of "1" – this is due to reset of the output cell value to 0, which is programmed
to occur with 1 msec delay as the rule (xxviii) requires. Respectively, cases (5) - (8) provide majority of "-1" at the input, so expected output value is "-1" and the obtained value is also "-1", as it is also confirmed in the Table 10 in accord with the rule (xxvi). Again, the output is delayed by 40 msec, which is due to total of four cells distance between the input and output cells, each cell having the delay time of

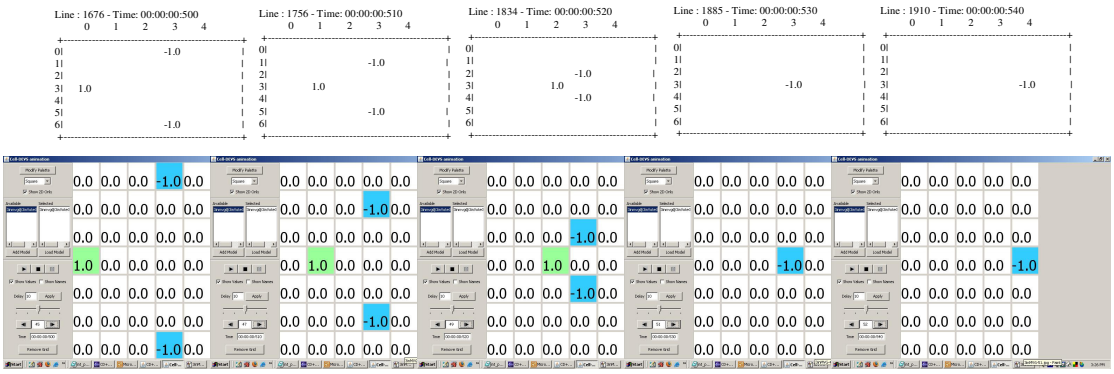

Figure 5. Comparison of visualization of signal propagation with drawlog tool of CD++ (top line) and animation tool (bottom line) in CD++ Modeler toolkit. (Input condition of case 6 of Table 2).

10 msec (rule (xxvii)). There is also expected "0" signal 1 msec after output of "-1" as was the case above for the same reason of resetting the output cell value to 0 with 1 msec delay (rule (xxviii)).

Animation example of the signal propagation in the Delayed Majority Vote Gates model is shown in Fig.5 produced by CD++ Modeler. For comparison on the top in Fig.5 shown is visualization with drawlog tool.

3. CONCLUSIONS

The following conclusions can be drawn form the above considerations.

1. CD++ toolkit is demonstrated as a suitable environment for simulation and visualization of the operation of quantum cellular automata type of nano-devices under Cell-DEVS formalism.
2. The models of quantum dot based XOR gates, quantum wires and majority vote gates are successfully implemented in Cell-DEVS formal definitions.
3. Hierarchy of the quantum cellular automata models has been successfully validated via establishing test rules and conducting the tests.

References:

[1] Zeigler, B.P., The brain-machine disanalogy revisited, *BioSystems,* Vol. 64, pp. 127-140. (2002).

[2] Obeid, I. Wolf, P.D, "Evaluation of spike-detection algorithms for a brain-machine interface application",- Biomedical Engineering, IEEE Transactions on, Volume 51, Issue 6, page(s) 905- 911, June 2004.

[3] G.Wainer, N. Giambiasi. "Timed Cell-DEVS: modelling and simulation of cell spaces ". Invited paper for the book: Discrete Event Modeling & Simulation: Enabling Future Technologies. Springer- Verlag, 2001.

[4] G. Wainer. "Applying Cell-DEVS Methodology for Modeling the Environment". In Simulation, Transactions of the SCS. Vol. 82, No. 10, 635-660. October 2006.

[5] Neto, O.P.V.; Pacheco, M.A.C.; Barbosa, C.R.H., "Neural Network Simulation and Evolutionary Synthesis of QCA Circuits",- Transactions on Computers, Volume 56, Issue 2, Feb. 2007 Page(s):191 - 201

[6] Buller A (2003b) Reversible Cascades and 3D Cellular Logic Machine, Technical Report TR-0012, ATR Human Information Science Laboratories, Kyoto.

[7] W.J. Townsend, JA Abraham - Nanotechnology, 2004. 4th IEEE Conference on, Complex gate implementations for quantum dot cellular automata, 2004 www.cerc.utexas.edu/~whitney/pdfs/nano04.pdf

[8] J. Watrous, "On one-dimensional quantum cellular automata", *Proc. 36th FOCS*, 1995: pp. 528–537.

[9] Aubrey Jaffer, "4-Neighbor 3-State Universal Cellular Automaton", Copyright 1974, 2002, 2004. http://swiss.csail.mit.edu/~jaffer/Cell/CAN4S3

[10] C.S. Lent, "Bypassing the Transistor Paradigm," Science, pp. 1597-1599, June 2000.

[11] Neto, O.P.V.; Pacheco, M.A.C.; Barbosa, C.R.H.; "Neural Network Simulation and Evolutionary Synthesis of QCA Circuits",- Transactions on Computers, Volume 56, Issue 2, Feb. 2007 Page(s):191 – 201.

[12] Bajec, I. L. Mraz, M., "Multi-Valued Logic Based on Quantum-Dot Cellular Automata",- International J., Unconventional Computing, vol.3; issue 4, pp. 311-322, 2007.

[13] Y. Boiko and G. Wainer, "Modeling Spiking Neural Terminal in DEVS",- 2008 Spring Simulation Multiconference (SpringSim'08 Poster Session, paper #30).