Ternary Quantum-Dot Cellular Automata

Matevž Baloh[1], Jani Bevk[1], Tadej Borovšak[2], and Tjaž Brelih[1]

[1]University of Ljubljana, Faculty of Computer and Information Science
[2]University of Ljubljana, Faculty of Mathematics and Physics

ABSTRACT

The ternary quantum-dot cellular automata promise to be an interesting nano-scale computing platform for implementing multi-valued logic. In this work we investigate the behaviour of basic constructs such as lines and simple logic gates, implemented as cellular automata on various architectures. Additionally, we present some new tools that make it easier to design and test (simulate) ternary quantum-dot cellular automata.

Introduction

It seems that our current, silicon-based, binary computing may be nearing its speed and size limits. It is becoming increasingly difficult to create faster silicon-based processors. Because of this, several alternative platforms for processing are being researched. One of the promising ones is quantum-dot cellular automata (QCA) due to its low power consumption, small size and high speed of operation[1].

The basic building block of QCA is a cell. A QCA cell is made from several quantum dots and two mobile electrons, which can either be located at a quantum dot or the tunnel between dots. The most basic version of binary QCA cells contain four dots, one in each corner of a square cell. Two mobile electrons can now form two stable polarizations: an electron in the upper-left and lower-right dots, or in the upper-right and lower-left dots. One represents logical *truth*, the other logical *false*. These polarizations are shown in figure 1.

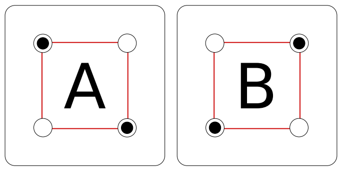


Figure 1. Stable polarizations of binary QCA cell.

QCA circuits are built by placing QCA cells in specific arrangements. If two cells are right next to each other, the same repulsive forces which cause cells to assume a stable state also cause the two adjacent cells to assume the same or opposite states, depending on their relative positioning. This means that correctly positioned QCA cells can be used to both transfer information and process it.

An electron's repulsion effect will however be subject to interference and other effects, which accumulate on a longer line of QCA cells, potentially causing incorrect results. To solve this, a clock system is used. Each clock cycle is made up of four phases: switch, hold, release and relax. These govern whether or not electrons can tunnel between dots in the QCA cell. During the hold phase, electrons cannot move from the dot they are in. During the relax phase, electrons can move freely. During the switch and release phases the tunnels are closing and opening, respectively. The clock stabilizes the QCA circuit and prevents accumulation of various effects which could produce incorrect processing or unstable states.

QCA cells can thus have a defined clock offset. These can theoretically be set for each individual QCA cell, however it is more reasonable to have a pre-defined pattern of clock phases, which are a part of the underlying board. One such pattern is the USE scheme[2], where the board is divided into small square areas (5×5), and the clock offset is defined for the entire area in a uniform pattern, which can be seen in figure 2.

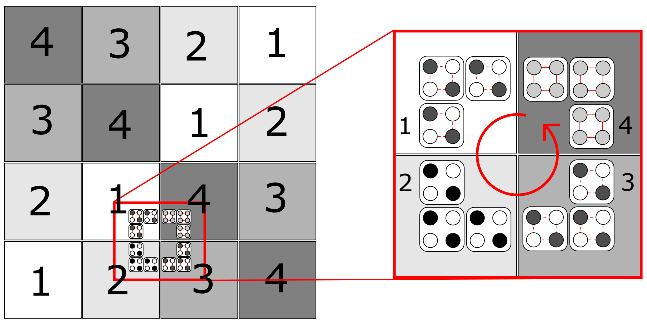


Figure 2. USE Clock pattern. The number represents the clock phase at a given point in time. In the next phase, numbers increase by one and 4 becomes 1.

Ternary QCAs

Ternary QCA (tQCA) cells are an expanded version of binary QCA cells, which can assume one of four stable polarizations: A, B, C and D, shown in figure 3. Those states represent three logical values: -1, 0 and 1 which correspond to values in ternary system. Logical value -1 is represented by state A, logical 0 is represented by states C and D and logical 1 is represented by state B. By convention, state D should not appear as an input or output state[3].

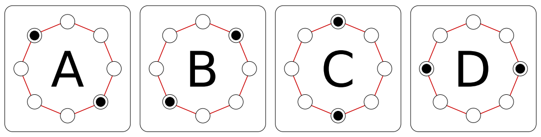


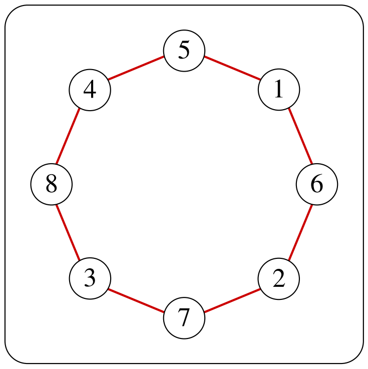
Figure 3. Stable polarizations of ternary QCA cell.

Ternary QCAs cells that we used in our experiments, have either eight quantum dots, as shown in figure 4a, or nine quantum dots, as shown in figure 4b. Each cell type is further parametrized by quantum dot size and distance between quantum dot centres. By varying those parameters, several architectures can be defined. The most commonly used ones are 91, 92, 60, 72 and 110. Quantum dot size is set to be 10 nm for all listed architectures, while the number of dots and distance between dot centres varies according to the table 1.

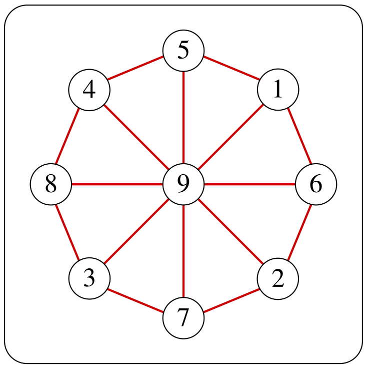
architecture	60	72	91	92	110
number of dots	8	8	9	9	8
r [nm]	14.1421	17.4208	17.4208	26.1313	26.1313

Table 1. Ternary QCA cell architectures.

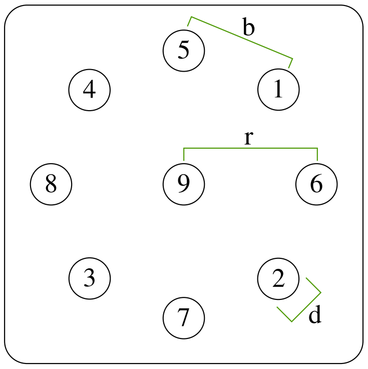
As with binary QCAs, we construct tQCA by placing cells next to each other. Distances between cell centres are not defined by architecture and greatly affect the final outcome of the signal transmission and/or calculation. In our analyses we most often used distances between 70 nm and 170 nm and deviated from this general range in special cases.



(a) Dots and tunnels on architectures 60, 72 and 110.



(b) Dots and tunnels on architectures 91 and 92.



(c) General cell layout. (r - radius, b - distance between dots, d - dot diameter)

Figure 4. Main structural properties of quantum-dot cells.

Signal transmission

We analysed how the architecture and the distance between the centres of tQCA cells affect the transmission of a signal through them. The analysis was done by simulating the transmission of a signal through several tQCA primitives. We considered the architectures 60, 72, 110, 91 and 92.

For each architecture, we ran the simulation multiple times while varying the distance between cell centres. This way we were able to determine the minimum and maximum distance between the cells needed for successful transmission in a given architecture. A transmission was considered successful if the resulting truth table was equal to expected truth table. For a simple (odd-length) line the output should be equal to the input as shown in table 2. State D was considered as an internal state and was excluded in these simulations as a possible input[4].

input	output
A	A
B	B
C	C

Table 2. Expected truth table of a simple line.

Horizontal line

We constructed a basic horizontal line, as shown on figure 5. The line consists of nine tQCA cells (excluding the electrode and the output cell). The length is odd to ensure the correct transmission of state C. We used only one clock phase for the driver, internal and output cells. The output was read after 0.375 clock periods, which is in the middle of the second clock phase.

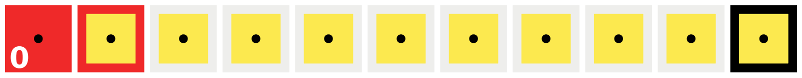


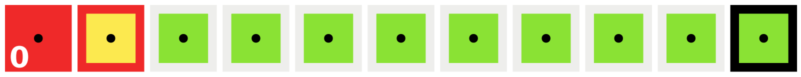
Figure 5. A horizontal line.

The results are displayed in table 3. As we are analysing a horizontal line only the horizontal distance between the centres of cells (denoted by x in the table) should impact the ability to successfully transmit a signal. We determined that the transmissions are successful when the horizontal distance is between 70 nm and 174 nm for architecture 60, between 102 nm and 198 nm for architecture 72, and between 135 nm and 261 nm for architecture 110. We observed that when the distance exceeds these upper bounds, the input state C will result in output state D, even though the length of the line is odd. When the distance is below these lower bounds, input states A or B result in output state C.

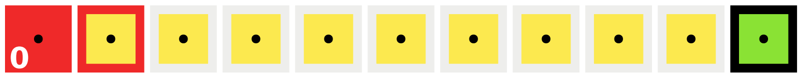
Unfortunately we were not able to find any distances for the 9-dot architectures (91, 92) with the line from figure 5 such that the transmission would be successful. We suspected the problem might lie in only using one clock phase. Therefore we tried some configurations with two clock phases, as shown on figures 6a, 6b and 6c. The output was read after 0.675 clock periods, which is in the middle of the third clock phase. With the line from figure 6c we were finally able to achieve successful

Distance	Architecture				
x	60	72	110	91 (fig. 6c)	92 (fig. 6c)
60	✗ $*$	-	-	✗	✗ $*$
68	✗ $*$	-	-	-	-
69	✗ $*$	-	-	-	-
70	✓ $*$	-	-	✗	-
72	✓	✗	-	-	-
77	-	-	-	✗ $\substack{**\\*}$	-
78	-	-	-	✓	-
80	-	-	-	✓	-
100	-	✗ $*$	-	-	✗ $\substack{**\\*}$
101	-	✗ $*$	-	-	-
102	-	✓ $*$	-	-	-
106	-	-	-	-	✗ $\substack{**\\*}$
107	-	-	-	-	✓
110	✓	✓	✗ $*$	✓	✓
120	-	✓	-	-	✓
133	-	-	✗ $*$	-	-
134	-	-	✗ $*$	-	-
135	-	-	✓ $*$	-	-
140	✓	✓	✓	-	✓
158	-	-	-	-	✓
159	-	-	-	-	✗ $\substack{**\\*}$
160	✓	✓	✓	✓	✗ $\substack{**\\*}$
170	✓	-	-	-	-
173	✓	-	-	-	-
174	✓	-	-	-	-
175	✗ $**$	-	-	-	-
180	✗ $**$	✓	✓	✓	-
181	-	-	-	✗ $\substack{**\\*}$	-
190	-	-	-	✗ $\substack{**\\*}$	-
197	-	✓	-	-	-
198	-	✓	-	-	-
199	-	✗ $**$	-	-	-
200	✗ $**$	✗ $**$	✓	✗ $\substack{**\\*}$	-
210	-	✗ $**$	-	-	-
250	-	-	✓	-	-
260	-	-	✓	-	-
261	-	-	✓	-	-
262	-	-	✗ $**$	-	-
263	-	-	✗ $**$	-	-
280	-	-	✗ $**$	-	-

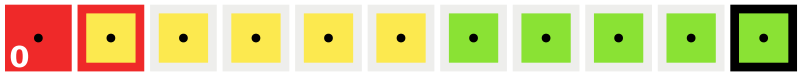
Table 3. Results for transmissions over a horizontal line. ✓and ✗ denote a successful and unsuccessful transmission, respectively. No data was collected at -. Asterisks denote the number of epsDisregards: $*$ 1-10, $**$ 11-50, $\substack{**\\*}$ 51-100, and $\substack{**\\**}$ 100+.



(a) A horizontal line with two clock phases. The transition is between the input cell and the first internal cell.



(b) A horizontal line with two clock phases. The transition is between the last internal cell and the output cell.



(c) A horizontal line with two clock phases. The transition is in the middle of the line.

Figure 6. Horizontal Lines with two clock phases.

transmission using 9-dot cells. These results are also shown in table 3. Signal transmission works for distances between 78 nm and 180 nm for architecture 91, and between 107 nm and 158 nm for architecture 92. However, having to use two clock phases on lines this short raises the question of their feasibility. The resulting delay might not be desirable in all situations.

Vertical line

We constructed a vertical line, which looked exactly the same as the horizontal line from the previous section, except rotated by 90 degrees. As the line is now vertical, only the vertical distance between the centres of cells impacts the ability to successfully transmit a signal.

Unsurprisingly, the results are very similar to those of a horizontal line. The lower distance bounds for successful transmission stayed the same, 70 nm for architecture 60, 102 nm for architecture 72, and 135 nm for architecture 110. The upper distance bound for architecture 60 also stayed the same at 174 nm. However, the upper distance bounds for architectures 72 and 110 have increased to 211 nm and 270 nm respectively. This came as a surprise as the tQCA cells are symmetrical and the direction of the line (horizontal or vertical) should not affect its ability to transmit a signal.

For 9-dot architectures, the results were exactly the same as with the horizontal line. Using lines shown on figures 6a, 6b and 6c yield no successful transmissions. Using the line from figure 6c, successful transmissions were observed on distance intervals from 78 nm to 180 nm for architecture 91, and from 107 nm to 158 nm for architecture 92.

Layered line

A crossing of multiple tQCA lines can be implemented by using multiple layers[5]. Therefore we analysed how signal transmission behaves with a line spanning multiple layers. We constructed a line similar to the horizontal and vertical line except that it now runs along the ‘normal’ z-axis, as shown on figure 7. Consequently, only the z-distance affects the signal transmission. With a layered line states A and B also alternate, not just states C and D. The line must therefore have an odd length.

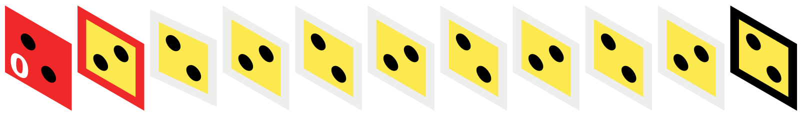


Figure 7. A layered line.

The results of our tests can be seen in table 4, however they are not as straightforward to interpret as before. This time we were also able to find some distances where the 9-dot architectures can successfully transmit a signal while using only one clock phase. The lower z-distance bounds for successful transmission are 18 nm for architecture 60, 22 nm for architecture 72, 31 nm for architecture 110, 17 nm for architecture 91, and 18 nm for architecture 92. Surprisingly, the successful transmission intervals are not continuous as they were before. We have found some gaps with architectures 72, 91, and 92, where the signal was not transmitted successfully. The upper distance bounds seem to be quite high at over 1000 nm for all architectures. However, the probabilities reported by the simulator drop below 1 somewhere between 350 nm and 500 nm (denoted by ‘P’ in table 4).

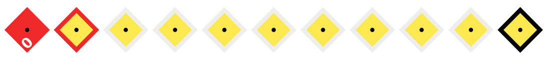
Diagonal line

We also constructed a diagonal line, as shown in figure 8a. Here, both the vertical and the horizontal distance between two neighbouring cell centres determine the result of signal transmission. Therefore we cannot directly compare results of diagonal lines against those of horizontal and vertical lines. The actual diagonal distance must be calculated first. As it turns out, the

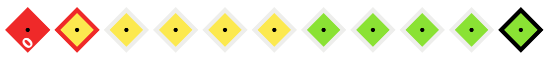
Distance	Architecture				
z	60	72	110	91	92
10	✗ $\overset{**}{*}$	-	-	✗ $*$ P:0.36	-
15	✗ $\overset{**}{*}$	-	-	✗ $*$ P:0.36	-
16	✗ $*$	-	-	✗ $*$ P:0.36	-
17	✗ $\overset{**}{*}$	-	-	✓ $\overset{**}{*}$	-
18	✓	-	-	-	-
20	✓	✗ $*$	-	✓ $\overset{**}{*}$	✗ $\overset{**}{*}$ P:0.36
21	-	✗ $\overset{**}{*}$	-	-	-
22	-	✓	-	-	-
25	-	-	-	-	✗ $\overset{**}{*}$
26	-	-	-	-	✓ $\overset{**}{*}$
30	✓	-	✗ $\overset{**}{*}$	✓	✓
31	-	-	✓	-	-
32	-	-	✓	-	-
33	-	-	✓ $\overset{**}{*}$	-	-
35	-	-	✓	-	-
36	-	✓	-	✗	✓
40	-	-	✓	-	-
55	✓	✓	✓	✓	✓
77	-	✓	-	-	-
78	-	✗ $\overset{**}{*}$	-	-	-
80	✓	✗ $\overset{**}{*}$	✓	✗ $\overset{**}{*}$	✓
82	-	✗ $\overset{**}{*}$	-	-	-
83	-	✓	-	-	-
85	-	✓ $\overset{**}{*}$	-	-	-
90	✓	-	-	✓	-
100	-	✓	-	✓	-
120	✓	-	-	-	-
150	-	-	✓ $\overset{**}{*}$	-	✓ $\overset{**}{*}$
200	✓	✓	✓ $\overset{**}{*}$	✓	✗ $\overset{**}{*}$
250	-	-	-	-	✗ $\overset{**}{*}$
300	-	-	-	✓	✓
350	-	-	-	✓ P:0.99	-
400	✓ P:0.99	✓ P:0.99	-	-	-
500	-	-	✓ P:0.99	✓ P:0.99	✓ P:0.99
1000	✓ P:0.93	✓ P:0.98	✓ P:0.99	✓ P:0.98	✓ P:0.99
1100	✓ P:0.84	-	-	-	-
1180	✓ P:0.69	-	-	-	-
1190	✗ P:0.67	-	-	✓ P:0.89	-
1340	-	✓ P:0.68	-	✓ P:0.68	-
1350	-	✗ P:0.67	-	✗ P:0.67	-
1500	-	-	✓ P:0.90	-	-
1700	-	-	✓ P:0.70	-	✓ P:0.70
1720	-	-	✗ P:0.67	-	✗ P:0.67
1730	-	-	✗ P:0.69	-	✗ P:0.68
1750	-	-	✗ P:0.73	-	✗ P:0.73

Table 4. Results for transmissions over a layered line. ✓and ✗ denote a successful and unsuccessful transmission, respectively. No data was collected at -. Asterisks denote the number of epsDisregards: $*$ 1-10, $**$ 11-50, $\overset{**}{*}$ 51-100, and $\overset{**}{**}$ 100+.

minimum and maximum actual distances between two neighbouring cell centres of a diagonal line, resulting in successful signal transmission, closely match those of horizontal or vertical lines. Results are shown in table 5. We did not analyse distances where two neighbouring cells start to overlap, even though the transmission of a signal might still be successful. Therefore the lower distance boundaries for architectures 60, 110 and 91 cannot be determined from table 5.



(a) Diagonal line with single clock.



(b) Diagonal line with two clocks.

Figure 8. Diagonal lines.

		Distance	Architecture				
x	y	$\sqrt{x^2+y^2}$	60	72	110	91 (fig. 8b)	92 (fig. 8b)
60	60	84,85	✓ $\ast$	-	-	✓	✗
72	72	101,82	-	✗ $\ast$	-	✓	✗ $\ast$
73	73	103,24	-	✓ $\ast$	-	-	-
74	74	104,65	-	✓	-	-	-
80	80	113,14	-	✓	-	✓	✓
100	100	141,42	✓	✓	-	✓	✓
110	110	155,57	✓	-	✓	✓	✓
120	120	169,71	✓	-	-	✓	✓
126	126	178,19	✓	-	-	-	-
127	127	179,61	✓	-	-	-	-
128	128	181,02	✗ $\ast\ast$	-	-	-	-
130	130	183,85	✗ $\ast\ast$	-	-	✗ $\ast\ast$	✓
140	140	197,99	✗ $\ast\ast$	✓	-	✗ $\substack{\ast\ast\\ \ast}$	✓
144	144	203,65	-	✓	-	-	-
145	145	205,06	-	✗ $\ast\ast$	-	-	-
150	150	212,13	-	✗ $\ast\ast$	-	✗ $\substack{\ast\ast\\ \ast}$	✓
170	170	240,42	-	-	✓	✗ $\substack{\ast\ast\\ \ast\ast}$	✗ $\substack{\ast\ast\\ \ast}$
178	178	251,73	-	-	✓	-	-
179	179	253,14	-	-	✓	-	-
180	180	254,56	-	-	✗ $\ast\ast$	✗ $\substack{\ast\ast\\ \ast\ast}$	✗ $\substack{\ast\ast\\ \ast}$
190	190	268,70	-	-	✗ $\ast\ast$	✗ $\substack{\ast\ast\\ \ast\ast}$	✗ $\substack{\ast\ast\\ \ast}$

Table 5. Results for transmissions over a diagonal line. ✓and ✗ denote a successful and unsuccessful transmission, respectively. No data was collected at -. Asterisks denote the number of epsDisregards: $\ast$ 1-10, $\ast\ast$ 11-50, $\substack{\ast\ast\\ \ast}$ 51-100, and $\substack{\ast\ast\\ \ast\ast}$ 100+.

Just like with horizontal and vertical lines we were unable to get any successful transmissions for architectures 91 and 92 using only a single clock phase line from figure 8a. By switching half of the line to another clock phase (as shown on figure 8b) and reading the output a quarter of a clock cycle later we managed to get these architectures working too.

Corner

Another structure that is often needed when routing signals is a corner. This relatively simple structure proved to be quite unstable when we tried to implement it by only using single clock phase. This instability can be attributed to interferences between non-neighbourhood cells that are impeding signal transmission. We resolved this problem by using separate clock phases for different sections of the corner as shown on figure 9. We read the output value after 0.625 clock periods.

Results of the simulation are shown in table 6. The data clearly suggests that cell architecture and distance between cell centres greatly affect the transmission success. Architectures with larger distances between dots fail to work properly when distances between cell centres are too small. Contrary to previous experiments, where architectures with nine dots usually took longer to transmit a signal, compared to architectures with eight dots, this line requires the same number of clock phases to transmit a signal on all architectures.

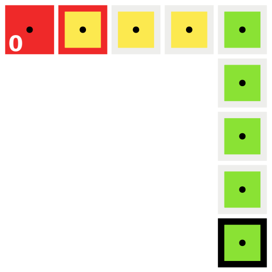


Figure 9. Structure that offers stable signal transmission.

Distance		Architecture				
x	y	60	72	110	91	92
70	70	✗ $\ast$	✗ $\substack{\ast\ast\\ \ast}$	✗	✗ $\ast$	✗
80	80	✗	✗ $\substack{\ast\ast\\ \ast}$	✗ $\ast$	✗ $\ast$	✗ $\ast$
90	90	✓	✗ $\ast$	✗ $\ast$	✗	✗
100	100	✗ $\ast$	✗ $\ast$	✗	✓ $\ast$	✗ $\ast$
110	110	✓	✓	✗ $\ast$	✓	✗
120	120	✓	✓	✗ $\ast$	✓	✗
130	130	✓	✓	✗ $\substack{\ast\ast\\ \ast}$	✓	✗ $\substack{\ast\ast\\ \ast}$
140	140	✓	✓ $\ast$	✗	✓ $\ast$	✓ $\substack{\ast\ast\\ \ast}$
150	150	✓	✓	✗ $\ast$	✓	✗ $\ast$
160	160	✓	✓	✓	✗ $\substack{\ast\ast\\ \ast}$	✓
170	170	✓	✓	✓	✗ $\substack{\ast\ast\\ \ast}$	✓

Table 6. Results for corner implementation. ✓and ✗ denote a successful and unsuccessful transmission, respectively. Asterisks denote the number of epsDisregards: $\ast$ 1-10, $\ast\ast$ 11-50, $\substack{\ast\ast\\ \ast}$ 51-100, and $\substack{\ast\ast\\ \ast\ast}$ 100+.

Fanout

We implemented and tested a fanout of a line using the construction shown in figure 10. The output has been read in the middle of the third clock phase, after 0.625 clock periods. As can be seen from table 7, this implementation is fairly robust and works on almost all architectures when distance between cell centres is between 100 nm and 170 nm.

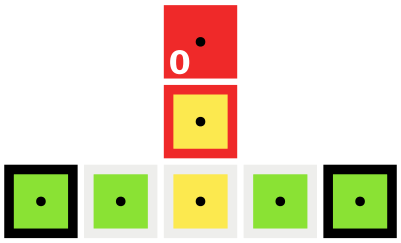


Figure 10. Fanout implementation using two clocks.

In order for the fanout to work as expected when transmitting state C, we must ensure that both paths through the fanout contain an odd number of cells. Failing this, states C on input can cause states D to appear on the output.

Distance		Architecture				
x	y	60	72	110	91	92
70	70	✓	✓	✗	✓	✗
80	80	✓	✓	✗	✓	✗
90	90	✓	✓	✗	✓	✗
100	100	✓	✓	✓	✓	✓
110	110	✓	✓	✓	✓	✓
120	120	✓	✓	✓	✓	✓
130	130	✓	✓	✓	✓	✓
140	140	✓	✓	✓	✓	✓
150	150	✓	✓	✓	✓	✓
160	160	✓	✓	✓	✓	✓
170	170	✓	✓	✓	✓	✓

Table 7. Results for fanout implementation. ✓and ✗ denote a successful and unsuccessful transmission, respectively.

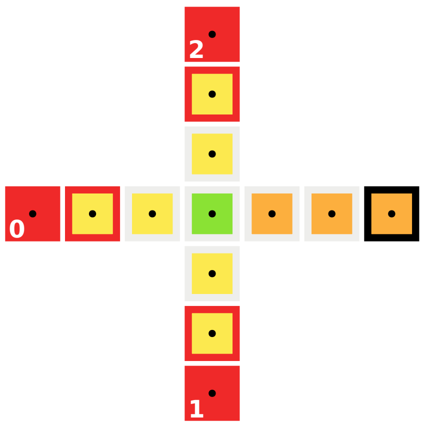
Information processing

QCAs can of course also process the signals, not just transmit them. In this section we analysed how various architectures and distances between cell centres influence information processing. This is achieved by combining influences of source cells at the target (output) cell. We tested two logical gates: the majority gate and the inverter.

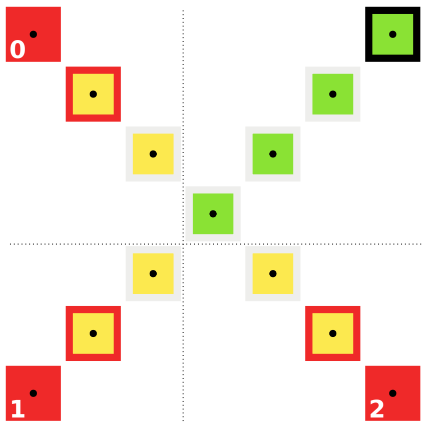
Majority gate

Majority gate is one of the most fundamental assets when it comes to designing QCAs. Its function is to calculate the majority of its three inputs. Having such functionality available, one can build AND and OR gates by simply fixing one of the inputs to an appropriate value.

There are two types of ternary majority gates, the regular majority gate and the diagonal majority gate. Both are shown on figure 11 and are directly adapted from their binary counterparts. In order to function properly, they require multiple clock phases. The regular majority gate requires three consecutive clock phases, while the diagonal majority gate requires only two. An alternative clock phase layout for the diagonal majority gate also exists, however we did not analyse it[6].



(a) Regular majority gate that uses three clock phases.



(b) Diagonal majority gate that uses two clock phases.

Figure 11. Two implementations of majority gates.

We analysed these gates at different architectures and distances between cell centres. We compared their output with the expected truth table (table 8), corresponding to ternary logic. Notice that we did not use state D as an input, since we consider this to only be an internal state.

Result for both types of majority gates are shown in table 9. The regular majority gate seems quite stable for architectures 60, 72 and 110. It works as expected with most of the distances between cell centres we tested. With some distances it also works for architectures 91 and 92. The diagonal majority gate however produced very different results. For architectures 91 and 92 we were unable to find a distance between cell centres where the gate would work as expected. Architecture 91 at the

x	y	z	r	x	y	z	r	x	y	z	r
A	A	A	A	B	A	A	A	C	A	A	A
A	A	B	A	B	A	B	B	C	A	B	C
A	A	C	A	B	A	C	C	C	A	C	C
A	B	A	A	B	B	A	B	C	B	A	C
A	B	B	B	B	B	B	B	C	B	B	B
A	B	C	C	B	B	C	B	C	B	C	C
A	C	A	A	B	C	A	C	C	C	A	C
A	C	B	C	B	C	B	B	C	C	B	C
A	C	C	C	B	C	C	C	C	C	C	C

Table 8. Expected truth table for ternary majority gate.

distance of 100 nm came the closest with only one combination of inputs resulting in an unexpected output (CAC $\rightarrow$ A). The architectures 60, 70, and 110 do work, albeit only at a very limited range of distances between cell centres.

Distance		Architecture (figure 11a)					Architecture (figure 11b)				
x	y	60	72	110	91	92	60	72	110	91	92
70	70	✓ $\ast$	✓ $\ast$	✗ $\overset{\ast\ast}{\ast}$	✓ $\ast$	✗ $\overset{\ast\ast}{\ast}$	✗ $\ast$	✗	✗ $\ast$	✗	✗ $\ast$
80	80	✓ $\ast$	✓ $\ast$	✗ $\ast$	✓ $\ast$	✗ $\ast$	✓ $\overset{\ast\ast}{\ast\ast}$	✗ $\ast$	✗ $\ast\ast$	✗ $\overset{\ast\ast}{\ast\ast}$	✗ $\ast\ast$
90	90	✓ $\ast$	✓ $\ast$	✗ $\ast$	✓ $\ast$	✓ $\ast$	✓ $\overset{\ast\ast}{\ast\ast}$	✓ $\overset{\ast\ast}{\ast\ast}$	✗ $\ast$	✗ $\overset{\ast\ast}{\ast\ast}$	✗ $\ast$
100	100	✓ $\ast$	✓ $\ast$	✓ $\ast\ast$	✗ $\ast$	✓ $\ast$	✗ $\overset{\ast\ast}{\ast}$	✓ $\overset{\ast\ast}{\ast\ast}$	✗	✗ $\overset{\ast\ast}{\ast\ast}$	✗ $\overset{\ast\ast}{\ast}$
110	110	✓ $\ast$	✓ $\ast$	✓ $\ast$	✗ $\ast$	✓ $\ast$	✗ $\ast$	✗ $\overset{\ast\ast}{\ast}$	✗ $\overset{\ast\ast}{\ast\ast}$	✗ $\overset{\ast\ast}{\ast}$	✗ $\overset{\ast\ast}{\ast\ast}$
120	120	✓ $\ast$	✓ $\ast$	✓ $\ast$	✗ $\ast$	✓ $\ast$	✗	✗	✓ $\overset{\ast\ast}{\ast\ast}$	✗	✗ $\overset{\ast\ast}{\ast\ast}$
130	130	✓ $\ast$	✓ $\ast$	✓ $\ast$	✗ $\ast$	✓ $\ast$	✗ $\overset{\ast\ast}{\ast}$	✗	✓ $\overset{\ast\ast}{\ast\ast}$	✗	✗ $\overset{\ast\ast}{\ast\ast}$
140	140	✓ $\ast\ast$	✓ $\ast$	✓ $\ast$	✗ $\ast$	✗ $\ast$	✗ $\overset{\ast\ast}{\ast}$	✗ $\ast\ast$	✗ $\overset{\ast\ast}{\ast\ast}$	✗ $\ast\ast$	✗ $\overset{\ast\ast}{\ast\ast}$
150	150	✓ $\ast$	✓ $\ast$	✓ $\ast$	✗ $\ast$	✗ $\ast$	✗ $\overset{\ast\ast}{\ast\ast}$	✗ $\overset{\ast\ast}{\ast}$	✗	✗ $\overset{\ast\ast}{\ast\ast}$	✗
160	160	✓ $\ast$	✓ $\ast\ast$	✓ $\ast$	✗ $\ast$	✗ $\ast$	✗ $\overset{\ast\ast}{\ast\ast}$	✗ $\overset{\ast\ast}{\ast}$	✗	✗ $\overset{\ast\ast}{\ast\ast}$	✗
170	170	✓ $\ast$	✓ $\ast$	✓ $\ast$	✗ $\ast$	✗ $\ast$	✗ $\overset{\ast\ast}{\ast\ast}$	✗ $\overset{\ast\ast}{\ast\ast}$	✗	✗ $\overset{\ast\ast}{\ast\ast}$	✗ $\ast\ast$

Table 9. Results for regular and diagonal majority gates. ✓and ✗ denote an expected and unexpected result, respectively. Asterisks denote the number of epsDisregards: $\ast$ 1-10, $\ast\ast$ 11-50, $\overset{\ast\ast}{\ast}$ 51-100, and $\overset{\ast\ast}{\ast\ast}$ 100+.

The diagonal majority gate is a good candidate for the 5×5 clock pattern. The central cell of the gate can easily be moved into the corner of a 5×5 area as shown on figure 11b. However this would only work with a scheme where the clock phases can be chosen freely as three of the four 5×5 areas the gate covers need to be in the same phase. This is not possible with a scheme such as USE[2].

It does not seem possible to put the regular majority gate on a 5×5 clock pattern. The gate is very compact and the boundary between the clock phases is deep inside the centre of the gate. Thus adjusting it for a 5×5 clock pattern would require severe changes in its structure.

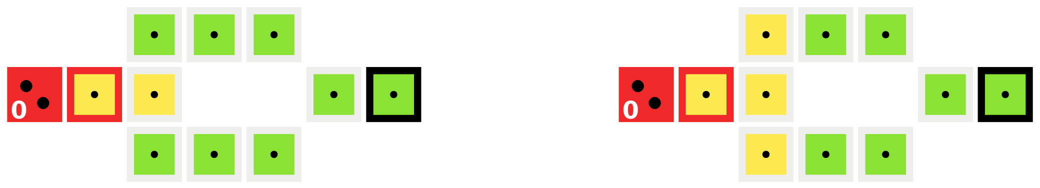
Inverter

In binary logic, an inverter flips the logic value of the input. The input *true* results in the output *false* and vice versa. The adaptation to ternary logic is simple. The inverter flips states A and B, and preserves state C as shown in table 10.

input	output
A	B
B	A
C	C

Table 10. Expected truth table of a ternary inverter.

Several different implementations of the inverter exist[6]. As already mentioned, a diagonal line of odd length functions as an inverter. Another possibility is the symmetric inverter, adapted from the binary QCAs. With these, multiple clock schemes can be used. Two possibilities are shown on figure 12. We designed the "modified" clocking scheme on figure 12b with the 5×5 clock patterns in mind. It can easily be placed on any 5×5 grid, USE scheme included, as it only requires two adjacent clock phases and the boundary between them is simple.



(a) Basic inverter implementation.

(b) Modified inverter implementation.

Figure 12. Two implementations of a symmetric inverter.

We already tested the diagonal line in a previous section so here we only tested the inverters from figure 12. Results can be seen in table 11. The basic implementation (figure 12a) seems to be quite stable on a variety of different architectures and distances between cell centres. However, it turns out that the modified implementation (figure 12b) provides even more stability than the basic implementation. This is a major advantage regardless whether a 5×5 clock pattern is used or not.

Distance		Architecture (figure 12a)					Architecture (figure 12b)				
x	y	60	72	110	91	92	60	72	110	91	92
70	70	✗	✗	✗ $\ast$	✗ $\ast$	✗ $\ast$	✓	✗	✗	✗	✗ $\ast$
80	80	✓	✗	✗	✗	✗	✓	✓	✗	✓	✗
90	90	✓	✓	✗	✓	✗ $\ast$	✓	✓	✗	✓	✗
100	100	✓	✓	✗	✓	✗ $\ast$	✓	✓	✗	✓	✗
110	110	✓	✓	✗ $\ast$	✓	✗ $\ast$	✓	✓	✗	✓	✗
120	120	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓
130	130	✓	✓	✗ $\overset{\ast\ast}{\ast\ast}$	✓	✓	✓	✓	✓	✓	✓
140	140	✓	✓ $\ast$	✓	✓	✓	✓	✓	✓	✓	✓
150	150	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
160	160	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
170	170	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 11. Results for both inverter implementations. ✓and ✗ denote a successful and unsuccessful result, respectively. Asterisks denote the number of epsDisregards: $\ast$ 1-10, $\ast\ast$ 11-50, $\overset{\ast\ast}{\ast}$ 51-100, and $\overset{\ast\ast}{\ast\ast}$ 100+.

Tool development

The editor and simulator we used to create and test automatons were first developed almost a decade ago for Microsoft Windows only. While the simulator received regular updates, the editor lagged behind and slowly became incompatible with the simulator, forcing users to manually edit the automata description files.

We ported the simulator to POSIX compliant C++ to cater for users on platforms other than Microsoft Windows and created a new graphical editor for creating ternary QCAs.

Porting simulator to POSIX C++

Authors of the original simulator were kind enough to grant us access to the simulator's source code, which we used as a base. While porting to GNU/Linux, we discovered a bug in the TNT mathematics library that caused the simulator to abort when calculating matrix eigenvalues. Fortunately, the TNT library authors had already released a fix for this problem, so we simply updated the internal copy of the library to the latest available version.

Apart from the problem described above, porting consisted mainly of replacing Windows specific functionality (such as various string types and threading API) with POSIX compliant alternatives. In order to get the simulator working on GNU/Linux, we were forced to make it single-threaded.

Future work on the simulator could include the removal of old format loaders and a reimplementation of multi-threaded simulation by utilizing OpenMP.

New QCA editor

While performing various tests and experiments, we quickly discovered the need for a new graphical editor for qdStruct files. These are files that are used to convey the structure of QCAs to the simulator. An existing editor is available, however it produces files that are not compatible with the latest version of the simulator and thus require manual modification. Additionally, multi-layered QCAs cannot be edited with this editor, and the user experience is severely lacking.

We contemplated fixing the old editor to support the new file format and layers, but modifying existing C++ code and using deprecated GTK+ libraries made little sense here. Instead, we created a new editor using python and the latest stable GTK+ libraries. Python was chosen in order to make the editor easy to maintain and extend.

Main features of the new editor are: support for multi-layered QCAs, latest qdStruct file format support, infinite canvas, and PDF export functionality. For more information about the new editor visit the editor's documentation.

Conclusions

In this work we analysed and tested various simple tQCA structures that can serve as building blocks for more complex automata. We determined that signal transmission over straight lines is fairly robust and that placing straight lines on a 5×5 grid is feasible. Transmitting signals over corners and fanouts is a bit more sensitive to interference from non-neighbouring cells, which necessitates the use of two consecutive clock phases in order to operate properly. This consequentially means that such structures need to be placed on the border of the 5×5 grid.

Cell architectures with eight quantum dots are generally more robust when it comes to cell placement, since they usually work on a wider array of spacings between cells. 9-dot architectures also seem to be more susceptible to interference from non-neighbouring cells and thus require more clock phases to work properly.

Regular majority gate needs three consecutive clock phases for its operation compared to only two phases that are needed by the diagonal one. This requirement makes it impossible to place regular majority gates onto a 5×5 grid. On the other hand, regular majority gates are much more robust than diagonal ones. The implemented inverter is one of the most robust structures we tested. A slightly modified version can be placed on a 5×5 grid with almost no limitations.

Insights gathered during these experiments can be readily applied when trying to place structures onto other clock placement patterns such as various signal distribution grids.

Author contributions statement

J.B. performed the analysis of the horizontal, vertical, layered line, and the majority gates. M.B. did research, wrote the introduction and reviewed the paper. T.Brelih performed the analysis of the diagonal line. T.Borovšak ported the simulator to POSIX C++, wrote the new editor, analysed the corner line, the fanout, and the inverter.

References

1. Ahmad, F. *et al.* Towards single layer quantum-dot cellular automata adders based on explicit interaction of cells. *Journal of Computational Science* **16**, 8 – 15 (2016).
2. Campos, C. A. T., Marciano, A. L., Neto, O. V. & Torres, F. S. Use: A universal, scalable, and efficient clocking scheme for qca. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **35**, 513–517 (2016).
3. Pečar, P. & Lebar Bajec, I. *The Key Elements of Logic Design in Ternary Quantum-Dot Cellular Automata*, 177–188 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2011).
4. Lebar Bajec, I., Zimic, N. & Mraz, M. The ternary quantum-dot cell and ternary logic. *Nanotechnology* **17**, 1937 (2006).
5. Lebar Bajec, I. & Pečar, P. Two-layer synchronized ternary quantum-dot cellular automata wire crossings. *Nanoscale research letters* **7**, 1 (2012).
6. Pečar, P., Ramšak, A., Zimic, N., Mraz, M. & Bajec, I. L. Adiabatic pipelining: a key to ternary computing with quantum dots. *Nanotechnology* **19**, 495401 (2008).