Using Symbolic Substitution Logic as an automated design procedure for QCA arithmetic circuits

Dietmar Fey, Bruno Kleinert
Department Computer Science 3
Friedrich-Alexander-University Erlangen-Nuremberg
D-91058 Erlangen, Germany
dietmar.fey@informatik.uni-erlangen.de, bruno.kleinert@informatik.uni-erlangen.de

***Abstract* — The paper presents a work-in-progress for an automatic synthesis procedure of an optical computing design principle onto future nanocomputing circuits based on Quantum-dot Cellular Automata (QCA). The goal of this work is to provide a contribution for the elimination of the lack of automatic design procedures for regular built-up QCA arithmetic circuits, which are built-up mostly manually so far. Furthermore by means of such a synthesis procedure a lot of designs made for optical computing arithmetic circuits can be directly transferred to QCA circuits. In this sense, this work is a contribution to ease the automatic synthesis of future arithmetic nanocomputing circuits.**

Keywords - optical computing; symbolic substitution logic; quantum-dot cellular automata, nanocomputing

I. INTRODUCTION

Symbolic Substitution Logic (SSL) was invented by Brenner et al. [1] in 1986 as a new method for the design of optical computing circuits. It was exactly tailored to the constraints and possibilities of a high-dense pixel parallel processing offered by optical hardware. The idea behind SSL is to search for a certain binary pattern within a binary pixel image and to replace the found patterns by another pattern. This substitution process can be exploited to realize a digital arithmetic in a highly parallel manner. The key features of SSL are characterized by their strong regularity concerning the pixel processing and the focusing on operating on elementary binary information cells, namely pixels, arranged in a grid structure.

In particular this situation is also given in Quantum-dot Cellular Automata (QCA) [2]. QCA is one of the promising nanotechnologies besides carbon nanotube field effect transistors and further nano device technologies based on tunneling effects that are considered as candidates for a new device technique to realize logic circuitry in the post CMOS area. Analogue to an optical computing scheme like SSL QCA are characterized by a highly dense implementation of binary information cells and a regular information flow. Whereas the elementary binary information cell in SSL was a pixel, which is either bright or dark, the binary information cell in QCA corresponds to two electrons, which are arranged in two distinguishable directions in a four dot quantum cell.

In literature, a really large number of proposals for QCA arithmetic circuits can be found, which have been developed largely manually (e.g. [3],[4]). However, there is still a lack of design methodologies that can be used for an automatic design process of arithmetic circuits based on QCA. There is an exception presented in [5], which proposes a methodology how to convert Boolean sum-of-products in an algorithmic way to QCA logic, in particular to QCA majority gates, which is the basic gate structure in QCA (see Section III). However, most of the QCA arithmetic circuits are still developed in a time consuming try-and-error process by hand.

On the other side there was a lot of research in the 1980s and 1990s in the Optical Computing community on SSL (e.g. [6],[7]), which brought numerous proposals for digital optical computing circuits based on the basic SSL logic building block, the so-called SSL rule (see Chapter II). Due to this fact and the similarities given in the kind how elementary information is handled in QCA and SSL, we present in this paper on-going research on developing strategies how SSL rules can be used for an automatic mapping process onto QCA circuits, which can be used in future design tools.

The rest of the paper is organized as follows. In Section II, we present the basic principles of digital optical computing based on SSL. In Section III, we explain nanotechnology information processing based on QCA. In Section IV, we present the mapping process between SSL rules and QCA cells for the example of one stage of a bit-serial QCA adder deduced from a SSL adder. Section V concludes and points out the remaining steps of this work.

II. OPTICAL COMPUTING WITH SSL

SSL [6],[7] has drawn a lot of attention during the 1980s and 1990s as a method for exploiting the space invariance of regular optical imaging systems for the set-up of digital optical hardware. The base of information processing in a SSL is the implementation of a so-called SSL rule. An SSL rule depicts a pattern substitution process and consists of two parts, a left-hand side (LHS) and right-hand side (RHS) pattern, (see Fig. 1). By a corresponding optical hardware each occurrence of the LHS pattern is searched within a binary image and is replaced by the RHS pattern. Fig. 2 shows schematically a possible optical set-up for the search process as it was frequently realized in SSL hardware demonstrators. The principle processing works as follows.

For each switched-off pixel, i.e., a black pixel, in the LHS of an SSL rule a copy of the image is produced, e.g., by a beam splitter. Furthermore a reference point is defined within the LHS pattern, e.g., the lower left corner pixel. Each of the copies is reflected, e.g., by tilted mirrors, in such a way that the copies are superimposed and pixels, which have

 ISBN: 978-1-61208-217-2

the same relative position to each other as defined in the LHS pattern, meet at the same location.

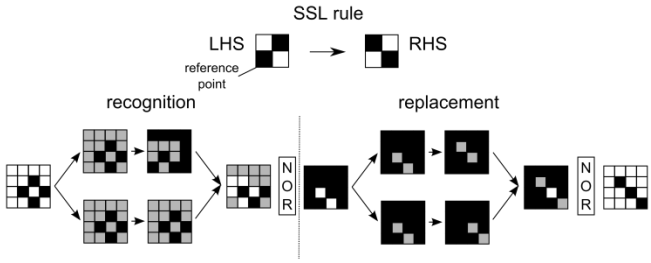

Figure 1. Principle of SSL.

For the example of Fig. 1, this means that one copy of the image is not tilted since it corresponds to the set pixel in the LHS pattern, which is already located in the reference point. Whereas the other copy is shifted by the tilted mirror, such that each pixel in the copy of the input image is shifted one pixel position down and left. At each position, where two dark pixels meet, an occurrence is given of the LHS pattern in the original input image. The superimposed image is mapped onto an array of optical threshold detectors. Each detector operates on one pixel of the superimposed image as a NOR device. The detector output is used for switching on a LED or laser diode. As a result, one gets a high light intensity at each pixel position, which corresponds to the occurrence of the LHS search pattern in the input image. We denote this new image as detector output image.

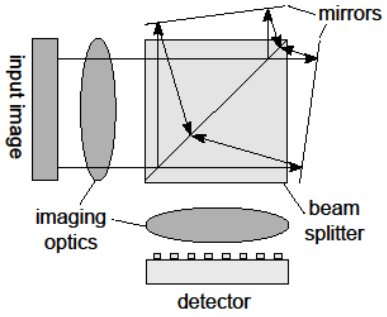

Figure 2. Implementation of SSL with optical hardware.

The recognition step is followed by a replacement step, which works analogue to the recognition step but in opposite direction. For each switched-on pixel in the RHS pattern a copy of the detector output image is again produced by optical beam splitter hardware in such a way that the copies are shifted towards the switched-on pixel in the RHS pattern. This means for the example of Fig. 1 that two copies from the detected output image are generated and each of the copies are shifted one pixel up resp. right before superimposing the copies. Once again, the superimposed image is mapped onto a pixel-by-pixel operating NOR detector and LED/laser diode array. The reproduced output is a new image, in which each occurrence of the LHS pattern in the original input image is substituted by the corresponding RHS pattern.

Implementing appropriate SSL rules by splitting the input image into multiple optical recognition and replacement paths, which are applied simultaneously and joined at the end, have been used for the proposing and realizing of digital optical computer arithmetic circuits. Fig. 3 shows this schematically for an optical ripple carry adder based on SSL. A large number of further arithmetic circuits using SSL or similar techniques like optical shadow logic [8] have been published in the past for optical adders, multipliers or image processing tasks. All these proposals can be used to transfer them to QCA due to the similarities between SSL and QCA we outlined above.

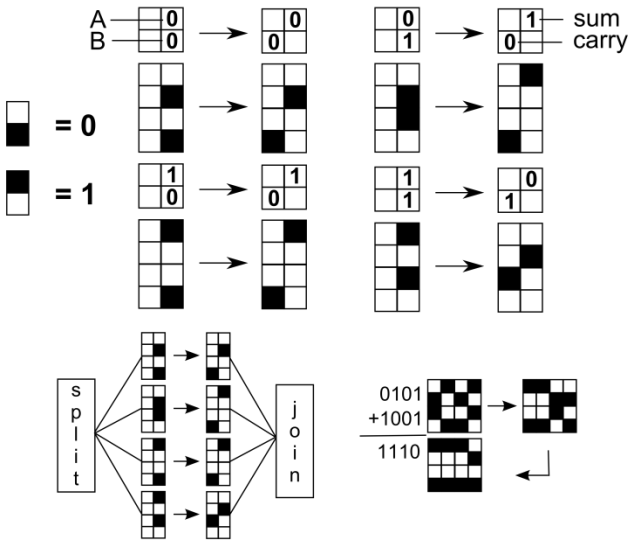

Figure 3. Realization of a ripple carry adder with SSL. For reasons of improved robustness a dual rail coding is used for 0 and 1.

III. NANOCOMPUTING WITH QCA

The elementary information cell in a QCA is a kind of container that groups a few quantum dots, at which charged particles, like e.g., electrons, are fixed (see Fig. 4). Mostly a QCA cell consists of four dots, in which two electrons are grouped in opposite order. Consequently, the cell knows exactly two polarizations orders, which are assigned to the binary values 0 or 1. Due to quantum mechanical rules it is possible that a cell can switch between the two states by tunneling of the charged particles between the dots. Concerning the two different particle arrangements one distinguishes between type 1 and type 2 cells.

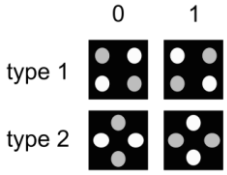

Figure 4. Binary coding in QCA cells. White circles correspond to empty qauntum dots, gray ones represent dots occupied with electrons.

A QCA cell serves not only as an information storage cell but also as a transport cell since neighbored QCA cells interchange by Coulumb forces. This means that a cell, which is fixed to a certain polarization, transfers its state to a neighbored cell because this arrangement shows the mini-

 ISBN: 978-1-61208-217-2

mum electrical field energy between neighbored charged particles. Consequently, a QCA wire can be built up, in which information is transported not by an electric current flow but by subsequent reordering of the quantum states in neighbored QCA cells. Due to the fact that no current is flowing and due to the extreme small dimensions of a QCA cell this technology offers very low power dissipation. Besides information transport one needs also logical gates to realize computing circuits. QCA logic utilizes an inverter and a so-called majority gate for this purpose. Fig. 5 shows an inverter built with cells of type 1. In both circuits, the output cell adopts the opposite state of the input cell state, again due to Coulombic forces. In contrast to CMOS circuits, QCA gate logic is not based on the switching of parallel and serial connected transistors but on the states of the cells surrounding a certain QCA cell, serving as output cell of the gate. The majority of the states in these surrounding cells determines the state of the output cell. In Fig. 5, a 3-input majority QCA gate is shown. The output cell adopts the same state, which at least is stored in two of the three neighbor cells. By fixing one of the inputs to a certain polarization 2-input AND, OR, NAND and NOR gates can be built-up.

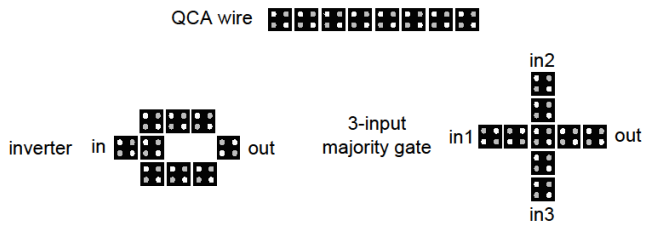

Figure 5. QCA logic building blocks.

Based on these three building blocks, QCA wire, QCA inverter and QCA majority gate, various proposals exist in literature for different typical digital circuits like adders, multipliers, shifters, multiplexers, flip-flop memories and registers, which have been found in a more or less try-and-error procedure. A very impressive collection of computer arithmetic QCA adder and multiplier circuits can be found in the work made by Hänninen [9]. The solutions proposed in this work are distinguished by their regular set-up that helps to realize QCA cells in the future. This is an important feature since QCA technology has a long-term perspective concerning its realization with real hardware.

Also design tools, which support the automatic synthesis of regular built-up QCA circuits, will encourage and give hints to device technologists how QCA technology should develop in the best way. In this sense, we propose to use optical computing SSL design procedure as a design entry point for the systematic design of nanocomputing QCA logic. How this mapping can be done is presented in the next section.

IV. MAPPING SSL RULES TO QCA LOGIC

The procedure to map SSL logic to a regular built-up QCA layout is subdivided in three steps. These steps correspond (i) to the core of the logic circuitry, namely the synthesis of a SSL rule into an equivalent QCA circuit, (ii) the realization of the splitting process, because we want to realize systems, which apply multiple SSL rules simultaneously, and (iii) the realization of the join at the end of the recognition-substitution stages. We will demonstrate the generic approach for these mapping steps in the following subsections without loss of generality on the example of the ripple carry adder from Fig. 3. Furthermore, we will use this example also to show generic applicable optimization measures for mapping SSL rules, which saves otherwise necessary QCA logic resources.

A. Mapping the Split stage to QCA cells

As shown in Fig. 3, the applying of multiple SSL rules starts with a Split function. The mapping of the Split stage onto QCA logic can be done in a straightforward manner. Producing copies of input cells can be simply done with branches of QCA wires running orthogonally to the input QCA wires. If one has to copy more than one input, as for example for the LHS rules in a ripple carry adder, one has to observe that crossing branches can interchange without conflicts. This can be done by crossing lines between QCA cells of type 1 and type 2. Converting between these two types can be realized with QCA cells, which are shifted about one half cell height (see Fig. 6, part Split).

B. Mapping SSL rules to QCA cells

The mapping of SSL rules onto equivalent QCA layouts is divided in two substeps, (i) the mapping of the recognition step, and (ii) the mapping of the replacement step. The recognition of a LHS of an SSL rule is mapped to an equivalent QCA majority gate realizing an appropriate AND gate. The number of inputs of this AND gate depends on the given number of values in the LHS. For example, the number of relevant inputs for the rules of the ripple carry adder is two. This means that a three-input QCA majority gate can be used, if one of the three inputs is fixed to 0 (see Fig. 6). For rules with a higher number of input values an appropriate majority AND gate has to be used. A lot of solutions for QCA gates with more than 3 inputs can be found in literature, e.g., in [10] an optimized solution for a 5-input majority gate is presented. If the value in the LHS is 0, then an inverter has to be included in the path of QCA cells that leads the input value corresponding to the LHS entry to the input of majority gate. The output of the majority cell is exactly 1, if the LHS pattern is detected. In this sense the majority gate works analogous to the photo detector NOR device used in SSL (see Fig. 1).

The following explanations correspond to the replacement stage in SSL. If the output of the majority gate is 0, then 0's are produced for all 1 values in the RHS of the corresponding SSL rule. If the output of the majority gate is 1, i.e., the LHS pattern was detected, a 1 is produced for each value 1 given in the RHS by an additional majority gate operating as an OR gate (see Fig. 6, majority gate in replacement part with one input fixed to 1). If the RHS is 0 no majority gate is necessary since we will work with wired-OR buses in the Join stage that carry already the 0 value, which is possibly inserted by the replacement stage located at the lowest position () in the wired-OR bus (see rule 2 in Fig. 6).

 ISBN: 978-1-61208-217-2

C. Mapping the Join stage to QCA cells

As just mentioned the principle of the Join stage in SSL is the realization of an optical wired-OR. The same idea is pursued for the equivalent QCA logic. If a 1 has to be inserted in the wire due to a relevant 1 from a RHS, which is output from a replacement stage, this can be done with 3-input majority gates with one input fixed to 1 (see Fig. 6,wired-OR bus in block rule 1). In this case, a 1 is only injected in the wire if the output of the attached recognition stage to the wire is 1 or the third input coming from the wire is already 1. This functioning corresponds exactly to a wired-OR bus. A logical 1 is injected if a LHS was found and a 1 in the corresponding output of the RHS is given. If the detected rule requires a 0 in the RHS this is automatically given by the fixed injection of a 0 in the QCA wire by the lowest replacement stage attached to the wired-OR bus. If the rules are not in conflict, i.e., only the LHS of exactly one rule was found, then only the output of the RHS belonging to the LHS is injected. This can be either a 0 or a 1. If it is a 0 an explicit injection is not necessary. This causes that rules, which have only 0's in the RHS, must not be implemented if it is secure that exactly one of the rules is always valid. This is given for the case of the ripple carry adder. Therefore, the rule corresponding to (A,B)=(0,0) has not to be implemented with corresponding QCA cells. If it is the only rule that holds, then the corresponding 0's in the output are already on the QCA wires. Utilizing this a priori knowledge the requirements to QCA hardware can be optimized during the synthesis process from SSL logic to QCA logic.

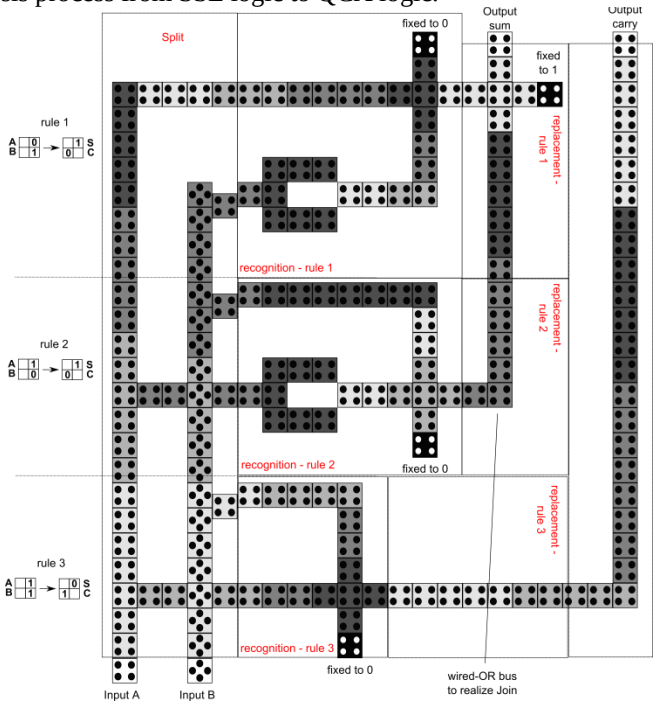

Figure 6. Result of the mapping process of SSL logic onto QCA logic for the ripple carry adder. To synchronize the changing of QCA cell states' four different clock zones have to be defined. In the figure these four clock zones are marked with a different gray level in QCA cells.

V. CONCLUSION AND FURTHER WORK

We have presented a generic design procedure for mapping digital optical computing circuits based on SSL onto nanocomputing QCA circuits. This will form both the base for future design tools for compact, regular built-up QCA circuits and supports the direct mapping of optical computing circuits to QCA technology. For example, we intend to map an integer arithmetic unit based on SSL, designed by us [11], onto a complete QCA integer unit, which does not yet exist so far. In addition, we have to verify the schematically shown QCA circuit of Fig. 6 by simulation with the QCA designer tool [12], the standard for simulating QCA layouts. So far, we have only verified parts of the circuit. Furthermore the insertion of an exact clocking scheme for the QCA cells has to be considered in the synthesis procedure. Nevertheless, the basic step for an automatic synthesis of SSL arithmetic circuits to QCA layouts is established.

REFERENCES

[1] K.-H.Brenner, A. Huang, and N. Streibl, "Digital Optical Computing with Symbolic Substitution", Appl. Opt. **25**, No. 18, pp. 3054 - 3060, (1986).

[2] C.S. Lent, P. Tougaw, W. Porod, and G. Bernstein, "Quantum cellular automata" Nanotechnology, vol. 4, 1993, pp. 49-57.

[3] V.A. Mardiris and Ioannis G. Karafyllidis, "Design and simulation of modular 2^n to 1 quantum-dot cellular automata (QCA) multiplexers," Int. J. Circ. Theor. Appl., 2010, vol. 38, pp. 771–785, doi: 10.1002/cta.595.

[4] F. Bruschi, F. Perini, V. Rana, and D. Sciuto, "An efficient Quantum-Dot Cellular Automata adder," Design, Automation & Test in Europe Conference & Exhibition (DATE), March 2011 , pp.1-4.

[5] R. Zhang, K. Walus, W. Wang, and G.A. Jullien, "A method of majority logic reduction for quantum cellular automata," IEEE Trans. on Nanotechnology, vol.3, no.4, Dec. 2004, pp. 443- 450, doi: 10.1109/ TNANO.2004.834177.

[6] Ahmed Louri, "Parallel implementation of optical symbolic substitution logic using shadow-casting and polarization," Applied Optics, Vol. 30, Issue 5, pp. 540-548 (1991) http://dx.doi.org/10.1364/AO.30.000540.

[7] K.-H. Brenner, W. Eckert, and C. Passon, "Demonstration of an optical pipeline adder and design concepts for its microintegration," Optics & Laser Technology, Vol. 26, No. 4, 1994, pp. 229 – 237.

[8] Y. Ichioka and J. Tanida, "Optical parallel logic gates using a shadow-casting system for optical digital computing," Proceedings of the IEEE , Vol. 72, No. 7, July 1984 , pp. 787 - 801 , doi:10.1109/PROC.1984.12939.

[9] I. Hänninen, "Computer Arithmetic on Quantum-Dot Cellular Automata Technology," Ph. D. thesis, Tampere University of Technology, 2009.

[10] R. Akkela and M.D. Wagh, "A Five-input Majority Gate in Quantum-dot Cellular Automata", NSTI-Nanotech 2011, Vol. 2, pp. 13-16, 2011.

[11] D. Fey and K.-H. Brenner, "Digital optical arithmetic based on systolic arrays andsymbolic substitution logic," Opt. Comput. 1, 153–167 (1990).

[12] K. Walus, T.J. Dysart, G.A. Jullien, and R.A. Budiman, "QCA Designer: a rapid design and Simulation tool for quantum-dot cellular automata," IEEE Trans. on Nanotechnology, Vol. 3, No. 1. (2004), pp. 26-31, doi:10.1109/ TNANO.2003.820815

 ISBN: 978-1-61208-217-2