Logic Synthesis for Regular Fabric realized in Quantum Dot Cellular Automata

Marek Perkowski
Portland State University

Alan Mishchenko
University of California - Berkeley

Logic Synthesis for Regular Fabric realized in Quantum Dot Cellular Automata

Marek Perkowski and Alan Mishchenko +

Department of Electrical and Computer Engineering, Portland State University,
Portland, OR 97207, USA, mperkows@ece.pdx.edu
+ Department of Electrical Engineering and Computer Science, University of California,
Berkeley, Berkeley, CA 94720

Abstract

Quantum Dot Cellular Automata are one of the most prospective nano-technologies to build digital circuits. Because of the requirements of only 2 layer wiring and noise avoidance, realizing the circuit in a regular fabrics is even more important for this technology than for classical technologies. In this paper, we propose a regular layout geometry called 3×3 lattice*. The main difference of this geometry compared to the known 2×2 lattices is that it allows the cofactors on a level to propagate to three rather than two nodes on the lower level. This gives additional freedom to synthesize compact functional representations. We propose a SAT-based algorithm, which exploits this freedom to synthesize 3×3 lattice representations of completely specified Boolean functions. The experimental results show that the algorithm generates compact layouts in reasonable time.

1. Introduction

Designers face many problems related to shrinking feature sizes of IC processes. Cross-talk, electromigration, self-heat, and other problems become more and more important. Cross-talk can lead to unpredictable design behaviors, high variation of delays and signal integrity problems in a wire depending on a state of a neighbor wire [10] and thus should be avoided at any cost. Book [10] proposes to solve the problem of cross-talk-immune IC design by using predetermined layout patterns on the IC, called the "layout fabric". It can be observed that the same problems exist in new technologies such as quantum dot cellular automata.

Bridging together logic synthesis and layout synthesis proceeds in several directions. One approach to this problem, exemplified by [28], makes logic synthesis aware of the layout early in the design flow. Another approach aims at creating specific layout structures with regular properties and synthesizing circuits using these structures. Research in regular structures to implement Boolean functions has started with the work of Akers [1]. The advantage of the regular layout fabrics is that they guarantee short wire length, predictable delay, and the absence of crosstalk. The disadvantage is that it may be difficult to derive compact representation for relatively complex logic functions.

Recently, regular layout fabrics are becoming popular with the new hardware implementation technologies such as single-electron transistor (SET) devices [9] and quantum dots [11, 12, 39]. A slide taken Papers [9, 39] illustrate the use of the 2*2 lattices, especially in SET technology. There is no routing and all connections are short. Another circuits of interest for regular structure approach are

* The term *lattice* is used to described the layout geometry because it is similar to a grid formed by logic gates and interconnections. The use of this term in the paper is not related to the set-theoretic concept of a lattice.

called Chemically Assembled Electronic Nanotechnology (CAEN) [6, 7]. CAEN is expected to offer significantly denser devices than CMOS technology. For example, a single RAM cell will require roughly 100nm^2 [6, 7]. In CMOS technology, a similar cell occupies 100,000nm^2 [9].

The regular layout structure called 2*2 lattice with Shannon gates at the nodes has been first proposed in [18, 19, 20, 21]. Publication [21] proposed also more general regular geometries and expansions other than Shannon, including Davio. These ideas were next expanded by several research groups [2, 5, 6, 13, 16, 23, 25] leading to the development of a number of efficient logic synthesis methods [5, 13, 23, 24, 36, 38].

There has been recently many papers published on Quantum Dots Cellular Automata (QCA) [4, 11, 12, 17, 30, 31, 32, 34, 35, 37, 39]. These papers use the majority and inverter gates for logic. Although the authors appreciate regularity in their designs of special functions, they did not consider a general-purpose design methods to design arbitrary combinational logic blocks. The only exception is [17] which proposes a general purpose FPGA, but without synthesis algorithms for it.

In this paper, we propose a new regular layout structure called 3*3 lattice. The 3*3 geometry preserves the close localization of logic elements characteristic of the 2*2 lattice but allows for more flexibility when implementing logic functions. We also propose the lattice synthesis algorithm, based on Boolean satisfiability, which makes use of the flexibility to reduce the number of levels and nodes in the lattice.

The remaining of the paper is organized as follows. Section 2 presents background on expansions and expansion-based regular fabric. We discuss quantum dot cellular automata in section 3 and we present our designs of regular fabric cells for them. Section 4 presents the lattice synthesis algorithm based on Satisfiability. Section 5 gives experimental results. Section 6 concludes the paper and outlines future work.

2. Background on expansion-based regular fabric

2.1. Expansions

Given a Boolean function F: $B^n \rightarrow B$, where $B = \{0,1\}$, the *negative (positive) cofactor* of F with respect to (w.r.t.) variable x is the Boolean function F_0 (F_1) derived by substituting into F instead of x the value 0 (1).

Let us denote F_2 the exclusive sum (EXOR) of the negative and positive cofactors: $F_2 = F_0 \oplus F_1$.

Three canonical expansions of F are defined as follows:

$F = \bar{x} F_0 \oplus x F_1$ Shannon expansion (S)

$F = F_0 \oplus x F_2$ Positive Davio expansion (pD)

$F = F_1 \oplus \bar{x} F_2$ Negative Davio expansion (nD)

Cofactors w.r.t. two and more variables are defined as repeated cofactoring w.r.t to each variable in the set. The final result does not depend on the variable order.

For example, function $F(a,b,c) = ab \vee a\bar{b}c \vee \bar{a}bc$ has the following cofactors w.r.t. variable a:

$F_{a=0} = F(\ 0, b, c\ ) = bc$.

$F_{a=1} = F(\ 1, b, c\ ) = b \vee c$.

Lattice Types

Regular layout fabrics discussed in this paper are called *lattices* [23]. Essentially, a lattice is a regular arrangement of gates locally interconnected to form a grid. Each gate has a control signal propagating from left to right and two data signals propagating from bottom to top. Lattice synthesis is typically performed from top to bottom when the levels of gates are synthesized one at a time until the level with constant cofactors is reached. Lattice with n levels can implement every totally symmetric binary function. It was proved [1] that all non-symmetric functions can be implemented in a 2*2 lattice be repeating variables in the levels and it was shown experimentally [36, 38] that the number of repetitions is rather low for many real-life benchmarks. In addition, many designs can be well-partitioned to logic blocks realized by lattices and next these blocks are placed and connected by routing.

Differences between the 2*2 lattices and the 3*3 lattices are illustrated in Figure 1, where circles represent gates and edges represent possible interconnections. The number pairs ("2*2" and "3*3") specify the lattice geometry: the first number tells how many gates of the lower level can feed into the given gate; the second number tells how many gates of the upper level can receive the output of the given gate. The general concept of "*k**k*" diagrams that included 2*2 and 3*3 lattices has been presented in [18, 20]. It should be noted that there exist various other 3*3 regular diagrams [2, 3, 16, 25] that are not 3*3 lattices in the sense of this paper but have been also called "lattices" by us, as a general class of expansion based regular layouts. However, in this paper the terms 2*2 and 3*3 lattices will refer only to structures shown in Figure 1.

Depending on the lattice type, three types of gates can be used in the nodes: Shannnon gates (MUXes), Positive Davio gates, and Negative Davio gates, created according to the three canonical expansions. Shannon lattices are built using only Shannnon gates, Kronecker lattices can have any of the three gates but only one gate type on each level. Pseudo-Kronecker lattices can have any of the three gates assigned to any node.

The lattices considered in this paper have the following additional flexibilities:

(1) the data inputs of a gate can be complemented;

(2) both data inputs of a gate can be connected to the same gate below.

In the synthesis methods developed for 3*3 lattices, in this paper the gates are limited to only Shannon gates.

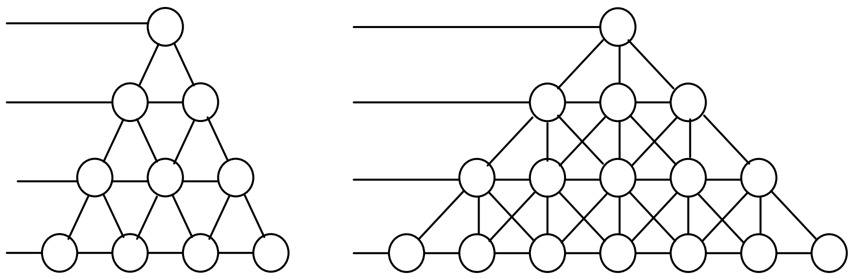


*Figure 1. 2*2 and 3*3 lattices.*

Comparison of 2*2 and 3*3 Geometries

Note that although, in the 3*3 lattice, a gate can receive inputs from any of the three gates on the lower level (left, center, right), no more than two gates actually provide the inputs (because each gate has only two data inputs). The synthesis algorithm can use this additional freedom for choosing two gates out of three candidates on the lower level to achieve a compact layout, with less logic levels and fewer gates.

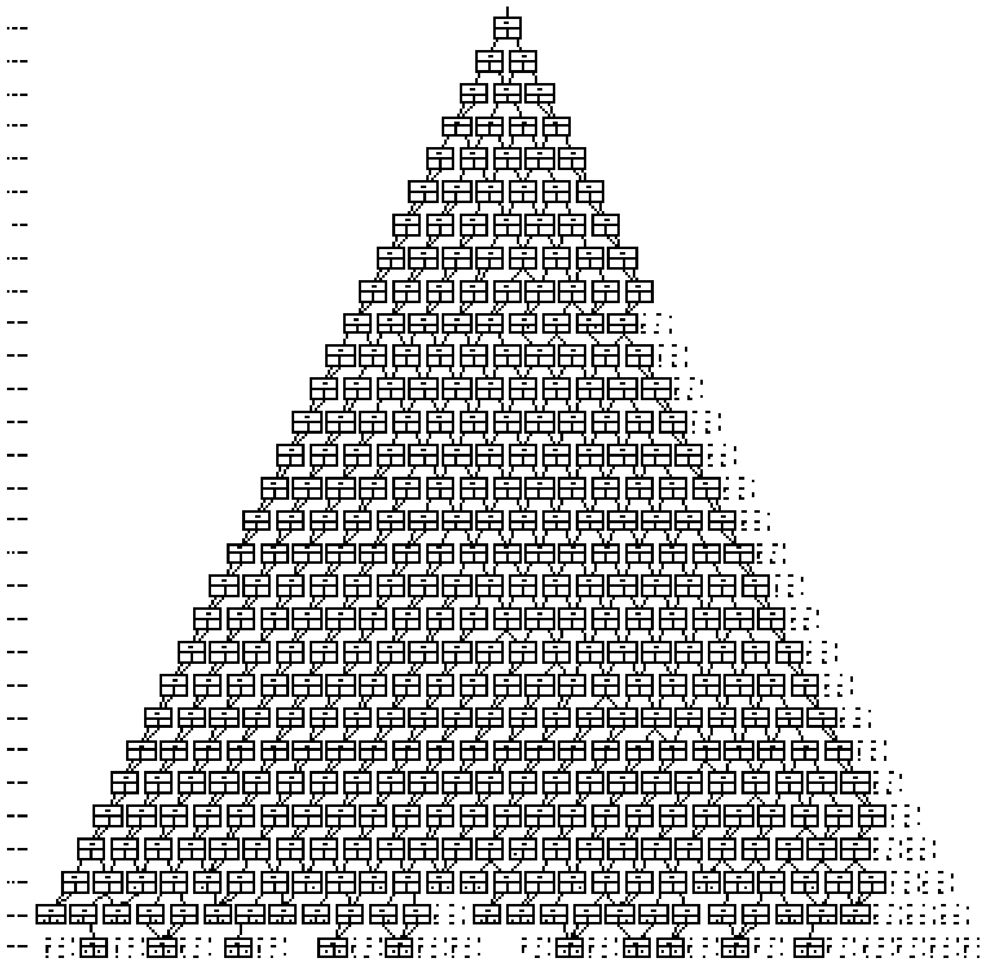
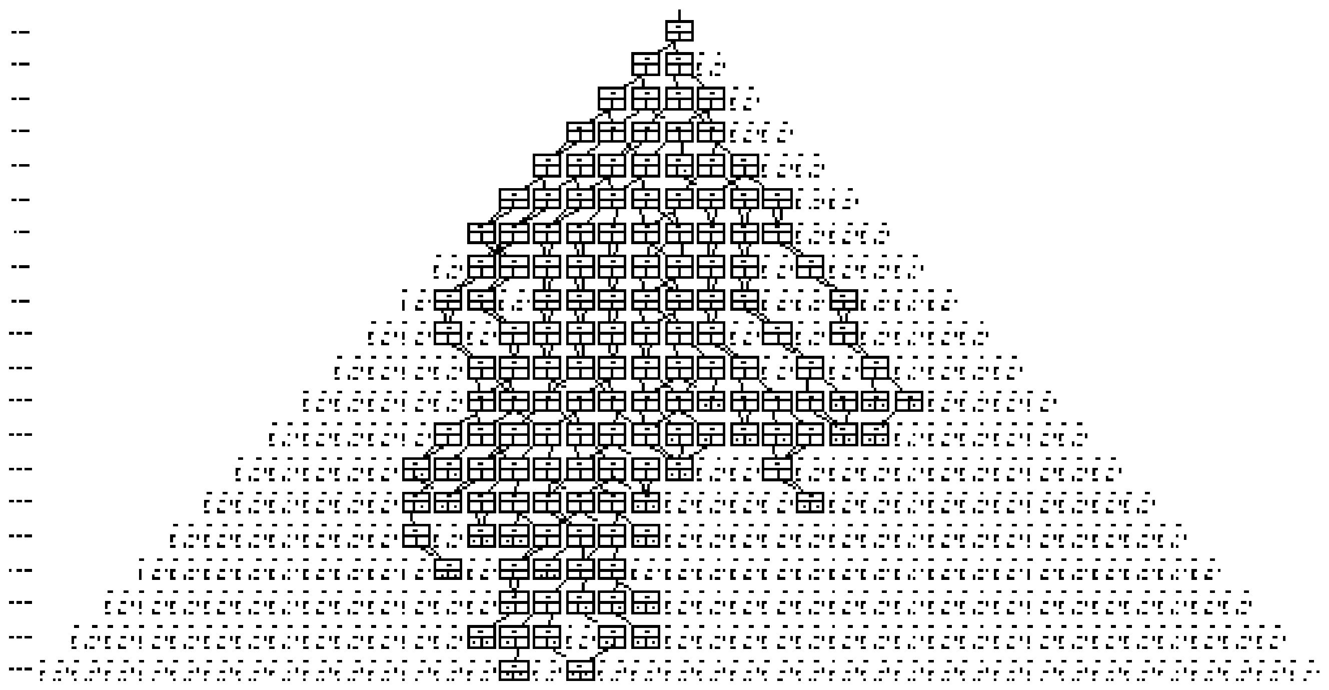


*Figure 2. Layou tof a 2*2 lattice for a random function of 7 variables.*



3*3 Lattice layout for the same function as in Figure 2.

Figure 2 shows the 2*2 lattice for a randomly generated 7 variable function. The 2*2 lattice was synthesized using the tools from [33]. Observe variable repetition that occurred, because the function was not totally symmetric. For comparison, Figure 3 shows the 3*3 lattice synthesized for the same function. The 3*3 lattice was created by a new tool, which implements the algorithm presented in this paper. Both lattices are synthesized using only Shannon gates. The 2*2 lattice has 29 levels and 392 nodes. The 3*3 lattice has 20 levels and 150 nodes.

Random functions are among those that are the most difficult to implement in regular structures. Figures 2 and 3 clearly demonstrate the advantage of the 3*3 geometry. For other functions, 3*3 lattices are never larger than 2*2 lattices but the difference is less pronounced. Of course, in the final layout the user can select a 2*2 or a 3*3 lattice for every combinational logic block.

3. Quantum Dot Cellular Automata Cells.

The exponential scaling in feature sizes follows still the Moore's Law using standard lithography based VLSI technology. However, as the fundamental limits of CMOS technology that are imposed by the laws of physics are approached, one thinks about what technology will replace the today's standard. The difficult challenges include noise, lithography costs, power, and many others. It is predicted that new nano-technologies will achieve density of 10^{12} devices/cm^{2} , will operate at THz frequencies, and with ultra low power consumption.

Quantum Dot Cellular Automata (QCA) have been first proposed by Lent et al in 1993 [11] and fabricated in 1997. In QCA, binary information is encoded in the charge configuration within quantum dot cells. A QCA cell can be in simplification treated as a structure of four dots (containers of charge, like electrons) located at the corners of a square. The electrons tend to occupy antipodal sites in a cell as a result of their mutual electrostatic repulsion. Thus there exist two equivalent energetically minimal arrangements of the two electrons in the QCA cell (as seen in Figures 4(a)). These two arrangements are denoted as cell polarization P = +1 and P = -1 respectively. We encode information with +1 representing logic 1 and -1 representing logic 0. Operation of QCA is based on quantum-mechanical interaction of electrons within quantum dots. Observe that this is a phenomenon that one wants to avoid in standard CMOS. The cell contain two extra mobile electrons which can quantum mechanically tunnel between cells. The dots can be realized in several different ways – electrostatically formed quantum dots in a semiconductor, small metallic islands connected by tunnel junctions, or redox centers in a molecule. [17]. While in standard CMOS information is transmitted by an electric current, operation of QCA is based on Coulombic interactions between cells. This way logic functions are realized because the state of a cell influences the neighbor cells. The fundament of QCA is self-latching where information is stored at each device by the position of single electrons. This means, that even at its lowest level this is a pipelining technology, in which pipelining is imbedded in every wire and inside every logic gate.

It is anticipated that the QCAs will be one of main technologies and that a QCA cell will have few nanometer size and will be fabricated through molecular implementation by a self-assembly process (which is one more argument of the same basic layout pattern to be repeated). QCA technology allows to realize both combinational and sequential circuits and several complete although simple computing blocks have been already designed, such as adders, barrel shifters, memories, microprocessors and FPGAs. Some QCA devices have been fabricated with metal cells operating at 50mk. It is predicted that QCAs operating at room-temperature will follow.

Current QCA systems use majority and inverter gates for which no good CAD tools exist. In addition, the layout is planar with only one layer and special cells and rules are required to realize the intersecting wires. There are no CAD tools for placement and routing of QCA systems based on these restrictions. As observed in [40], two wires in a QCA placed close to one another interfere similarly to the crosstalk effect in standard CMOS. Thus, one has to try to use for QCA CAD methods that have been already developed in standard CMOS to fight cross-talk and again, regular fabrics such as lattices are one of proposed methods with respect to this aspect. The current work on QCA takes already into account combining the logic synthesis and physical design aspects, since for this technology these two aspects cannot be separated from the very beginning [17, 31, 37].

See [37] for a brief introduction to realization of logic functions in QCAs and examples of designs and design methodology. Figure 4 has a cell for 2*2 Shannon Lattice. Figure 5 includes a cell for a 3*3 Shannon Lattice in which there are two ways of ordering cofactors. Figure 6 presents a cell of a generalized Lattice that can be personalized to Positive Davio, Negative Davio and Shannon expansions, and also to some other functions. These cells have been designed for abutting in regular fabrics.

4. SAT-Based Lattice Synthesis

This section shows how the problem of synthesizing one level of the 3*3 lattice with Shannon gates can be reduced to a SAT instance. We assume the reader's familiarity with the basics of Boolean satisfiability as presented, for example, in [14, 15].

Outline of the Algorithm

Synthesis of the 3*3 lattice is performed level by level. On each level, we solve the SAT problem, create the layout of that level, and proceed to the next level. Synthesis terminates when a level is reached on which all cofactors are constants. To formulate the SAT problem for one level of the lattice, we consider all the nodes on the level following immediately after the given one. The given level is called the *upper level*; the level following immediately after the given one is called the *lower level*. We formulate the set of requirements representing all possible routings of cofactors from the upper level to the lower level. Only non-constant cofactors are considered for routing. The constraints generated for all the nodes of the lower level are added to the set of all constraints representing the SAT instance.

Variables

Let us consider node N on the lower level. In the 3*3 lattice, there are six cofactors ($c_1,\ldots,c_6$) that can potentially be routed to this node. These cofactors come from the nodes on the left (L), in the center (C), and on the right (R) nodes, with respect to the node N (Figure 7).

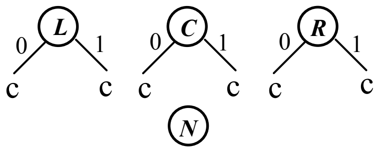


Figure 7. The routing choices for node N.

A group of SAT variables (x^N_1, …, x^N_k) is associated with each node N of the lower level. These variables represent mutually exclusive possibilities of joining in node N the cofactors coming from the upper level.

Suppose the cofactors (c_1,…,c_6) are different non-constant Boolean functions. The group of six variables (x^N_1, …, x^N_6) is used to represent the possibilities that node N receives only one cofactor. Another group of six variables (x^N_7, …, x^N_{12}) is used to represent the possibility that N receives a pair of cofactors combined using join-vertex operation [16, 18, 23].

The join-vertex operation is defined for cofactors coming from different nodes and having opposite polarity. This is why there are only six pairs, namely (c_1,c_4), (c_2,c_3), (c_3,c_6), (c_4,c_5), (c_1,c_6), and (c_2,c_5).

Clauses

The clauses of the SAT problem are divided into two categories: *covering constraints* and *closure constraints*. The covering constraints specify the requirement that each non-constant cofactor on the upper level is routed to the lower level. The closure constraints represent two types of requirements:

(1) each cofactor on the upper level is routed no more than once, and

(2) each node on the lower level is used no more than once.

There are as many covering constraints as there are non-constant cofactors on the upper level. Each covering constraint is the disjunction of variables, which represent routing choices involving the given cofactor. Suppose, for some cofactor, there are m such variables. The mutual exclusiveness of these variables translates into $m(m\text{-}1)/2$ closure constraints of type (1), specifying that no two variables are equal to 1 at the same time.

The closure constraints of type (2) are similar. For each node N_i on the lower level, they represent the mutual exclusiveness of variables in (x^{Ni}_1, …, x^{Ni}_{ki}). There are $k_i(k_i\text{-}1)/2$ such constraints for node N_i with k_i associated variables. Because $k \leq 12$, the number of these constraints does not exceed 66 for any node N_i.

The Number of Variables and Clauses

In this subsection, we approximate the number of clauses and constraints in a SAT problem, which arises in the lattice synthesis on level n.

In the 3*3 lattice, level n (the upper level) consists of $2n - 1$ nodes. The level n+1 (the lower level) consists of $2n + 1$ nodes. Each node on the lower level can have up to 12 variables, so the upper bound V on the number of variables is

$$V(n) = 12(2n+1).$$

The number of covering constraints is equal to the number of cofactors on level n, which is 2(2n-1). In the worst case, the number of the closure constraints of type (1) and type (2) is 66(2n-1) and 66(2n+1), respectively. This yields the upper bound C on the number of clauses in the SAT problem

$$C(n) = 2n - 1 + 66(2n\text{-}1) + 66(2n\text{+}1) \approx 266n.$$

The worst-case number of variables and clauses grows *linearly* with the number of lattice levels. For example, for n=10, there are at the most V = 252 variables and C = 2660 clauses. When some of the cofactors on the upper level are constants or equal up to complementation, it is possible to introduce less than 12 variables for nodes of the lower level. These special cases can be used to significantly reduce the routing choices. As a result, the actual number of clauses is typically two times smaller than the worst-case estimation.

Weighted SAT Problem

Each variable (x^{Ni}_1, …, x^{Ni}_{ki}) in the SAT problem is assigned a cost. The negative assignment of a variable has the cost 0. The positive assignment of a variable representing the join-vertex operation has a positive cost which is higher compared to the cost of the variable representing routing of the cofactors or joining of the cofactors equal up to complementation.

In solving the SAT problem, we look for satisfying variable assignments having the smallest cost. Such solutions would guarantee that there are relatively few nodes with the join-vertex operation, which helps reducing the number of levels in the lattice.

The complexity of the weighted SAT problem is NP-complete.

Trading Quality for Runtime

To simplify the complexity of the SAT problem, each level can be split into parts, for which SAT is solved independently. There is no routing of cofactors across the boundaries of the parts.

This approach to reduce complexity of SAT allows for trading the layout quality for runtime. The smaller are the individual SAT problems, the faster they are solved. On the other hand, splitting a level into many parts leads to the sub-optimal layout near the boundaries.

In practice, we found that having the parts composed of 5-7 nodes allows for a reasonable tradeoff between the optimality of the solution and the runtime of weighted SAT problems, which in this case takes approximately 0.1 seconds per instance.

5. Experimental Results

We have implemented the 3x3 lattice synthesis in a C program using the BDD package CUDD [29] for the manipulation of Boolean functions and the MINCOV package in SIS [27] to iteratively solve the weighted Boolean satisfiability problem.

We tested our program on selected MCNC benchmarks. The resulting lattices were written into BLIF files and verified against the initial specification of the benchmark functions. The runtime for any particular example was dominated by the runtime of SAT solver and did not exceed three seconds on a 933MHz Pentium III PC under MS Windows 2000.

Table 1 shows the comparison of our results with those published in [5] for 2*2 lattices. The 2*2 lattices are called Pseudo-Symmetric Binary Decision Diagrams (PSBDDs). They use only Shannon gates and allow for the same additional flexibilities:

(1) the data inputs of a gate can be complemented;

(2) both data inputs of a gate can be connected to the same gate below.

Column “Name” gives the name of the benchmark circuit. Column “Outs” gives the total number of outputs in the circuit and, in parentheses, the particular output used for testing. Similarly, column “Ins” gives the total number of input in the circuit and, in parentheses, the number of variables in the support of the output used for testing. Column “2*2 Lattice” gives the number of levels and nodes in the lattice reported in [5]. Column “3*3 Lattice” gives the number of levels and nodes in our implementation.

The experimental result in Table 1 show that synthesizing benchmark functions into the 3*3 lattices helps reducing the number of levels by 24% and the number of nodes by 56%, which speaks for the efficiency of the SAT-based synthesis algorithm.

Table 1. Experimental results.

Name	**Outs**	**Ins**	**2 * 2 Lattice**		**3 * 3 Lattice**	
			levels	**nodes**	**levels**	**nodes**
apex7	37(30)	49(17)	25	148	19	33
clip	5(1)	9(9)	18	103	9	30
	5(2)	9(9)	27	220	9	30
cm162a	5(3)	14(10)	11	24	10	18
cps	109(1)	24(22)	26	134	24	61
	109(2)	24(18)	26	164	21	66
	109(3)	24(22)	39	342	30	128
duke2	29(3)	22(15)	18	52	15	69
	29(6)	22(17)	18	47	17	36
	29(18)	22(15)	22	92	15	44
example 2	66(23)	85(16)	21	52	16	45
	66(59)	85(14)	17	31	14	21
	66(63)	85(13)	15	37	13	26
frg2	139(99	143(20)	22	189	20	28
	139(10 0)	143(19)	28	164	20	61
sao2	4(2)	10(10)	18	71	14	56
	4(3)	10(10)	16	73	14	65
	4(4)	10(10)	16	68	11	46
Total			383	2011	291	863
Ratio, %			100	100	**76**	**43**

6. Conclusions

Quantum Dots Cellular Automata (QCA) are a new idea which promises to build highly dense, low power, high-speed nano-scale computing. This technology allows to build logic into wires so that processing is done everywhere in the chip, but so far this idea has not been used practically and much percent of the chip are just wires that are not performing logical functions. Here comes a perfect match with the old idea of regular logic [18 – 26], fabrics and cellular automata organization of computing systems. In this paper we introduced the concept of regular fabric for quantum dot cellular automata and we presented a regular layout structure called 3*3 lattice. We demonstrated that this structure gives additional freedom to implement Boolean functions and is quite favourable for realizations of arbitrary logic blocks in QCA. We proposed a synthesis algorithm, which uses this freedom to generate the lattice layout with a smaller number of levels and nodes, compared to the known 2*2 lattice synthesis algorithms. The proposed synthesis algorithm reduces the problem of lattice synthesis on a level to an instance of weighted Boolean satisfiability. We showed that the worst-case number of variables and clauses in the SAT instances is linear in the level number to be synthesized. An additional advantage of the SAT formulation is that it allows for an efficient trade-off between the layout quality and runtime.

Our future work in this area includes clocking schemes (see [17, 31] for clock design issues) and simulation using QCADesigner [34]. Other work will include generalizing the synthesis algorithm to synthesize Kronecker and Pseudo-Kronecker 3*3 lattices and integrating our tools with one of the powerful SAT solvers developed recently, for example [15]. It is important to note that these ideas can be expanded to regular structures based on multiple-valued and fuzzy logic expansions and their reversible counterparts [2, 3, 18, 19, 20, 21, 22, 25, 26] and are thus applicable to many emerging nano-technologies for computing. As observed in [9, 39], the Single Electron Transistor technology can be used as an example. Additional research is needed to extend our algorithm for multi-output functions. The method will be used in conjunction with Ashenhurst/Curtis and Bi-Decomposition, because only after such decompositions the results are well-realizable for several benchmarks. Otherwise, too many variable-repetitions are needed and the shape of the lattice differs too much from a rectangle. The presented 3*3 lattices cannot be used alone, they should be used only as a part of the comprehensive logic synthesis system. We are not claiming that 3*3 lattices are always better than 2*2 lattices or other realizations of QCAs, we believe only that various regular fabric tools should be available in a system so that the user will make the best choice. For instance, 2*2 lattices are good for completely symmetric functions.

7. References

[1] S. B. Akers. A Rectangular Logic Array, *IEEE Trans. on Computers*. Vol. C-21, pp. 848-857, 1972.

[2] Al-Rabadi and M. Perkowski, Shannon and Davio Sets of New Lattice Structures for Logic Synthesis in Three-Dimensional Space, *Proc. RM'0.*

[3] Al-Rabadi and M. Perkowski, New Classes of Multi-valued Reversible Decompositions for Three-Dimensional Layout, *Proc. RM'01.*

[4] G.H. Bernstein, Quantum-dot Cellular Automata: Computing by Field Polarization, *Proc. DAC 2003*, pp. 268-273.

[5] M. Chrzanowska-Jeske, Y. Xu, M. Perkowski, Logic Synthesis for a Regular Layout. *VLSI Design: An International Journal of Custom-Chip Design, Simulation, and Testing*, 1999.

[6] B.T. Drucker, C.M. Files, M. A. Perkowski, and M.Chrzanowska-Jeske. Polarized Pseudo-Kronecker Symmetry with an Application to the Synthesis of Lattice Decision Diagrams. *Proc. ICCIMA'98 Conference*, pp. 745-755.

[7] P. Farm and E. Dubrova, Technology Mapping for Chemically Assembled Electronic Nanotechnology, *Proc. IWLS '02.*

[8] S. Goldstein and M. Budiu, Nanofabrics: Spatial computing using molecular electronics, *Proc. 28th Annual International Symposium on Computer Architecture*, (Gothenborg, Sweden), June 2001.

[9] H. Hasegawa, A. Ito, Ch. Jiang, and T. Muranaka, Atomic Assisted Selective MBE Growth of InGaAs Linear and Hexagonal Nanowire Networks For Novel Quantum Circuits, *Proc. 4th Intern Workshop on Novel Index Surfaces (NIS'01),* Sept 16-20, Apset France.

[10] S.P. Khatri, R.K. Brayton, A.L. Sangiovanni-Vincentelli, Cross-Talk Noise Immune VLSI Design Using Regular Layout Fabrics, Kluwer Academic Publishers, Boston, Hardbound, ISBN 0-7923, 7407-X, June 2001, 144 pp

[11] C. Lent, P. Tougaw, W. Porod, and G. Bernstein, Quantum cellular automata, Nanotechnology, 4: pp. 49-57, 1993.

[12] C.S. Lent, and P.D. Tougaw, A device architecture for computing with quantum dots, Proceedings of the IEEE, 85, 4, pp. 541 - 547, 1997.

[13] P. Lindgren, R. Drechsler, B. Becker, Synthesis of Pseudo-Kronecker Lattice Diagrams. *Proc. of Intl. Worshop of Applications of Reed-Muller Expantions to Circuit Synthesis, 1999,* Victoria, B. C., Canada, pp. 197 - 204.

[14] J. P. Marques-Silva, K. A. Sakallah, GRASP: A Search Algorithm for Propositional Satisfiability. *IEEE Trans. Comp*. Vol. 48, No. 5, May 1999, pp. 506-521.

[15] M. W. Moskewicz et al., Chaff: Engineering an Efficient SAT Solver. *Proc. DAC'01,* pp. 530-535.

[16] A. Mukherjee, R. Sudhakar, M. Marek-Sadowska, S. I. Long, Wave Steering in YADDs: A Novel Non-Iterative Synthesis and Layout Technique. *Proc. DAC'99*, pp. 466-471.

[17] M. T. Niemier, A. F. Rodriques, P.M. Kogge, A Potentially Implementable FPGA for Quantum Dot Cellular Automata, *Workshop on Non-Silicon Computation (NSC), in conjunction with Int. Symp. On High Performance Computer Architecture (HPCA)*, February 3, 2003

[18] M. Perkowski, and E. Pierzchala, New Canonical Forms for Four-Valued Logic, Report, Electrical Engineering Department, PSU . 1993.

[19] E. Pierzchala, and M. Perkowski, *Patent #5,959,871,* September 28, 1999. *Programmable Analog Array Circuit.*

[20] M. Perkowski, E. Pierzchala, and R. Drechsler, Ternary and Quaternary Lattice Diagrams for Linearly-Independent Logic, Multiple-Valued Logic and Analog Synthesis, Proc. ISIC-97, Singapur, 10-12 Sept.1997.

[21] M. Perkowski, L. Jozwiak, R. Drechsler, and B. J. Falkowski, Ordered and Shared, Linearly Independent, Variable-Pair Decision Diagrams, Proc. First International Conference on Information, Communications and Signal Processing, ICICS'97, Singapur, 9-12 Sept. 1997. Session 1C1: Spectral Techniques and Decision Diagrams.

[22] M. Perkowski, E. Pierzchala, and R. Drechsler, Layout-Driven Synthesis for Submicron Technology: Mapping Expansions to Regular Lattices, Proc. First International Conference on Information, Communications and Signal Processing, ICICS'97, Singapur, 9-12 Sept. 1997. Session 1C1: Spectral Techniques and Decision Diagrams.

[23] M. Perkowski, M. Chrzanowska-Jeske, and Y. Xu, Lattice Diagrams Using Reed-Muller Logic. *Proc. of Intl. Worshop of Applications of Reed-Muller Expansions to Circuit Synthesis, 1997,* Oxford Univ., U.K., pp. 85 - 102.

[24] M. A. Perkowski, M. Chrzanowska-Jeske, and Yang Xu, Multi-Level Programmable Arrays for Sub-Micron Technology based on Symmetries, Proc. ICCIMA'98 Conference, pp. 707-720, February 1998, Australia, published by World Scientific.

[25] M. Perkowski, A. Al-Rabadi, P. Kerntopf, A. Mishchenko and M. Chrzanowska-Jeske, Three-Dimensional Realization of Multiple-Valued Functions using Reversible Logic, Invited Talk, Proc. Workshop on Post-Binary Ultra-Large Scale Integration Systems (ULSI), , pp. 47 - 53, May 21, 2001, Warsaw, Poland.

[26] E. Pierzchala, M. A. Perkowski, S. Grygiel, A Field Programmable Analog Arrray for Continuous, Fuzzy and Multi-Valued Logic Applications, Proc. ISMVL'94, pp. 148 - 155, Boston, MA, May 25-27, 1994.

[27] E. Sentovich, et al., SIS: A System for Sequential Circuit Synthesis, *Tech. Rep. UCB/ERI, M92/41*, ERL, Dept. of EECS, Univ. of California, Berkeley, 1992.

[28] A. Singh, G. Parthasarathy, M. Marek-Sadowska, Interconnect Resource-Aware Placement for Hierarchical FPGAs. *Proc. ICCAD '01*, pp. 132-137.

[29] F. Somenzi. *CUDD Package, Rel. 2.3.1*, http://vlsi.Colorado.EDU/~fabio/CUDD/cuddIntro.html

[30] P.D. Tougaw, C.S. Lent, Logical devices implemented using quantum cellular automata, *J. Appl. Phys.*, American Institute of Physics, 75, (3), 1818, 1994.

[31] A. Vetteth, K. Walus, V.S. Dimitrov, G.A. Jullien, Quantum-Dot Cellular Automata Carry-Look-Ahead Adder and Barrel Shifter, *Proc. IEEE Emerging Telecommunications Technologies Conference*, Dallas, Texas, September 2002.

[32] A. Vetteth, K. Walus, G. A. Jullien, V. S. Dimitrov, "RAM design using quantum-dot cellular automata", Vol. 2, Pages 160-163, 2003 Nanotechnology Conference, February 23-27, 2003, San Francisco, CA.

[33] VLSI Design Automation Laboratory. *Pseudo-Symmetric Kronecker Functional Decision Diagrams.* http://web.pdx.edu/~suresh/pskfdd/main.php

[34] K. Walus, QCADesigner Homepage, http://www.atips.ca/~walus, ATIPS Laboratory, University of Calgary, Calgary, AB, CA, 2002.

[35] K. Walus, G. A. Jullien, V. S. Dimitrov, Computer Arithmetic Structures for Quantum Cellular Automata, *Proc. Asilomar Conference on Signals, Systems, and Computers*, November 9-12, 2003, Pacific Grove, CA. IEEE

[36] W. Wang, M. Chrzanowska-Jeske, Optimizing Pseudo-Symmetric Binary Decision Diagrams Using Multiple Symmetries. *Proc. IWLS'98*, pp. 134-140.

[37] W. Wang, K. Walus, and G.A. Jullien, Quantum-Dot Cellular Automata Adders, Adders", *IEEE Nano 2003 Conference*, San Francisco, CA. 2003.

[38] W. Wang, M. Chrzanowska-Jeske, Generating Linear Arrays Using Symmetry Chain. *Proc. IWLS'99*, pp.115 -119.

[39] T. Yamada, Y. Kinoshita, S. Kasai, H. Hasegawa, Y. Amemiya, Quantum Dot Logic Circuits Based on Shared Binary-Decision Diagram, *Jpn. J. Appl. Phys.* Vol. 40, 2002, pp. 4485-4488, Part 1, No. 7, July 2001.

[40] M. B. Tahoori, M. Momenzadeh, J. Huang, F. Lombardi, Defects and Faults in Quantum Dot Cellular Automata Based Designs, in *Proc. of Design Automation and Test in Europe (DATE)*, 2004.