Partitioning and Placement for Buildable QCA Circuits

Ramprasad Ravichandran†, Mike Niemier†, and Sung Kyu Lim‡
†College Of Computing, Georgia Institute of Technology
‡School of Electrical and Computer Engineering, Georgia Institute of Technology
{raam@cc, mniemier@cc, limsk@ece}.gatech.edu

***Abstract*— Quantum-dot Cellular Automata (QCA) is a novel computing mechanism that can represent binary information based on spatial distribution of electron charge configuration in chemical molecules. In this paper, we present partitioning and placement algorithms for a large-scale automatic QCA layout. The purpose of zone partitioning is to initially partition a given circuit such that a single clock potential modulates the inter-dot barriers in all of the QCA cells within each zone. We then place these zones during our placement step. We identify several objectives and constraints that will enhance the buildability of QCA circuits and use them in our optimization process. The results are intended to define what is computationally interesting and could actually be built within a set of predefined constraints.**

I. Introduction

Nanotechnology and devices will have revolutionary impact on the Computer-Aided Design (CAD) field. Similarly, CAD research at circuit, logic and architectural levels for nano devices can provide valuable feedbacks to nano research and illuminate ways for developing new nano devices. It is time for CAD researchers to play an active role in nano research.

Our goal in this paper is to explain how CAD can help research to move from small circuits to small systems of quantum-dot cellular automata (QCA) [1], [2] devices shown in Figure 1. We leverage our ties to physical scientists who are working to build real QCA devices. Based upon this interaction, a set of near-term *buildability constraints* has evolved - essentially a list of logical constructs that are viewed as implementable by physical scientists in the nearer-term. Until recently, most of the design optimizations have been done by hand. Then these initial attempts to automate the process of removing a single, undesirable, and unimplementable feature from a design were quite successful. We now intend to use CAD, especially physical layout automation, to address all undesirable features of design that could hinder movement toward a "buildability point" in QCA. In particular, we propose QCA partitioning and global placement algorithms so that the total wire crossing is minimized and 4-phase clocking constraints are satisfiled. The net result should be an expanded subset of computationally interesting tasks that can be accomplished within the constraints of a given buildability point. CAD will also be used to project what is possible as the state-of-the-art in physical science expands.

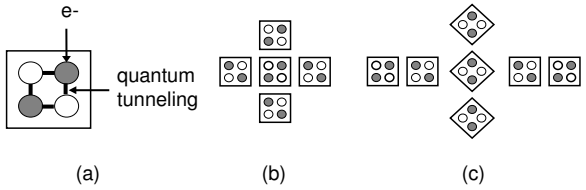

Fig. 1. (a) Schematic representation of a QCA cell, (b) QCA majority gate that can be configured to implement AND2 and OR2 functions, (c) horizontal wire is 90-degree wire and vertical wire is 45-degree wire. Crossing is allowed between 90-degree and 45-degree wires.

II. Problem Formulation

A. Overview of the Approach

QCA placement is divided into three steps: zone partitioning, zone placement, and cell placement. The purpose of zone partitioning is to decompose an input circuit such that a single potential modulates the inner-dot barriers in all of the QCA cells that are grouped within a clocking zone. Unless QCA cells are grouped into zones to provide zone-level clock signals, each individual QCA cell will need to be clocked. The wiring required to clock each cell individually would easily overwhelm the simplicity won by the inherent local interconnectivity of the QCA architecture. However, because the delay of the biggest partition also determines the overall clock period, the size of each partition must also be determined carefully. In addition, four-phase clocking imposes a strict constraint on how to perform partitioning. The zone placement step takes as input a set of zones–with each zone assigned a clocking label obtained from zone partitioning. The output of zone placement is the best possible layout for arranging the zones on a two dimensional chip area. Finally, cell placement visits each zone to determine the location of each individual logic QCA cell – (cells are used to build majority gates). Our recent work on cell placement is available in [3].

B. Zone Partitioning Problem

A gate-level circuit is represented with a directed acyclic graph (DAG) $G(V, E)$. Let P denote a partitioning of V into K non-overlapping and non-empty blocks. Let $G'(V', E')$ be a graph derived from P, where V' is a set of logic blocks and E' is a set of cut edges based on P. A directed edge $e(x, y)$ is *cut* if x and y belong to different blocks in P. Two paths p and

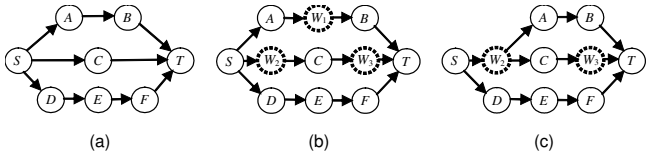

Fig. 2. Illustration of reconverent path constraint. (a) all three reconvergent paths from S to T are unbalanced. If S is in the switch phase, A, B, and T will be in relax, release, and hold phase. This puts C and T into relax and release, thereby causing a conflict at T. The bottom path forces T to be in switch phase, causing more conflict. (b) wire blocks W_1, W_2, and W_3 are inserted to resolve this QCA clocking inconsistency. (c) some wire blocks are shared to minimize the area overhead.

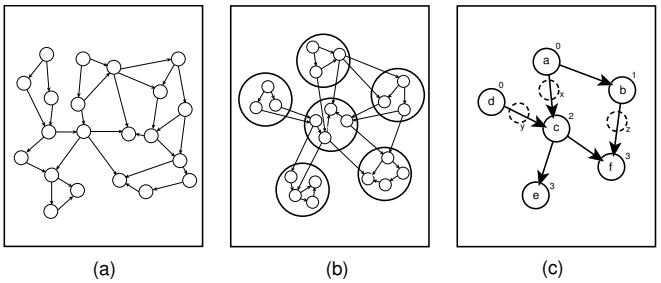

Fig. 3. Illustration of zone partitioning and wire block insertion. (a) directed graph model of input circuit, (b) zone partitioning under acyclicity and reconvergent path constraint, (c) wire block insertion, where the numbers denote the longest path length. The dotted nodes indicate wire blocks.

q in G' are *reconvergent* if they diverge from and reconverge to the same blocks as illustrated in Figure 2(a). If $l(p)$denotes the length of a reconvergent path p in G', then $l(p)$ is defined to be the number of cut edges along p. A formal definition of zone partitioning problem is as follows:

Definition 1: Zone partitioning: we seek a partitioning of logic gates in the given netlist into a set of zones so that cutsize (= total number of cut nets), wire block (= required during the subsequent zone placement) are minimized. The area of each partition needs to be bounded (area constraint), and there should not exist cyclic dependency among partitions (acyclic constraint). In addition, the length of all reconvergent paths should be balanced (clocking constraint).

The reconvergent path constraint is illustrated in Figure 2. Cycles may exist among partitions as long as their lengths are multiples of four (i.e. because of an assumed 4-phase, QCA clock). However, it becomes difficult to enforce this constraint while handling other objectives and constraints. Therefore, we prevent any cycles from forming at the partition level. In addition, it is difficult to maintain the reconvergent path constraint during the partitioning process. Therefore, we allow the reconvergent path constraint to be violated and perform a post-process to add *wire blocks* to fix this problem. Since the addition of wire blocks causes an overall increase in area to increase, we minimize the amount of wire blocks that are needed to completely remove the reconvergent path problems during zone partitioning.

C. Zone Placement Problem

Assuming that all partitions (= zone) have the same area, placement of zones becomes a geometric embedding of the partitioned network onto a $m \times n$ grid, where each logic/wire block is assigned to a unique location in the grid. In this case, a bipartite graph exists for every pair of neighboring clocking levels. We define the *k-layered bipartite graph* as follows:

Definition 2: K-layered bipartite graph: a directed graph $G(V, E)$ is k-layered bipartite graph iff (i) V is divided into k disjoint partitions, (ii) each partition p is assigned a level, denoted $lev(p)$, and (iii) for every edge $e = (x, y)$, $lev(y) = lev(x) + 1$.

Therefore, the zone placement problem is to embed a *zone-level k-layered bipartite graph* onto an $m \times n$ grid so that all blocks in the same layer are placed in the same row. All the I/O terminals are assumed to be located on the top and bottom boundary of each block, and we may insert routing channels between clocking levels for the subsequent routing. A formal definition of zone placement problem is as follows:

Definition 3: Zone placement: we seek to place the zones we obtain from zone partitioning onto a 2D space so that area, wire crossings and wire length are minimized. Each zone (= logic/wire block) is labeled with a clocking level (= longest path length from input zones), and all zones with the same clocking level should be placed in the same row (clocking constraint). In addition, all inter-zone wires need to connect two neighboring rows (neighboring constraint).

III. Zone Partitioning Algorithm

A. Zone Partitioning

Let $lev(p)$ denote the longest path length from the input partitions (partitions with no incoming edges) to partition p, where the path length is the number of partitions along the path. Then $wire(e)$ denotes the total number of wire blocks to be inserted on an inter-partition edge e to resolve the unbalanced reconvergent path problem (clocking constraint of the QCA zone partitioning problem). Simply, $wire(e) = lev(y) + lev(x) - 1$ for $e = (x, y)$, and the total number of wiring blocks required without resource sharing is $\sum wire(e)$. Thus, our heuristic approach is to minimize the $\sum wire(e)$ among all inter-zone edges while maintaining acyclicity. Then, during post-processing, any remaining clocking problems are fixed by inserting and sharing wire blocks. An illustration of zone partitioning and wire block insertion is shown in Figure 3.

First, the cells are topologically sorted and evenly divided into a number of partitions $(p_1, p_2, \cdots p_k)$. The partitions are then level numbered using a breadth-first search. Next, the acyclic FM partitioning algorithm [4] is performed on adjacent partitions p_i and p_{i+1}. Constraints that must be met during any cell move include area and acyclicity. The cell gain has two components: cutsize gain and wire block gain. The former indicates the reduction in the number of inter-partition wires, whereas the latter indicates the reduction in the total number of wire blocks required. We then find the best partition based on a combined cost function for both cutsize and wire block

gain. Multiple passes are performed on two partitions p_i and p_{i+1} until there is no more improvement on the cost. Then, this acyclic bipartitioning is performed on partitions p_{i+1} and p_{i+2}, etc.

Movement of a single cell could change $lev(p)$, the level number of a partition p. Therefore every time a cell move is made, we check to see if this cell move affects the level number. Levels can change as a result of a newly introduced inter-zone edge or from completely removing an inter-zone edge. To update levels, we maintain a maxparent for each p so that the level number of the parent of p is $lev(p)-1$. $lev(F)$ is defined as the level number of the "from block" of a cell c and $lev(T)$ is defined as the level number of the "to block" of c. In the first case where a new inter-partition edge is created, $lev(T)$ is updated if $lev(F) \geq lev(T)$ after the cell move. In this case, $lev(T) = lev(F) + 1$. Then, we recursively update the maxparent and levels of all downstream partitions. In the second case where an existing inter-partition edge is removed, the maxparent again needs to be update.

B. Wire Block Insertion

During post-processing, any remaining clocking problems are fixed by inserting and sharing wire blocks, while satisfying wire capacity constraints. The input to this algorithm is the set of partitions and inter-partition edges. First, a super-source node is inserted in the graph whose fan-out neighbors are the original sources in the graph. This is done to ensure that all sources are in the same clocking zone. Then the single-source longest path is computed for the graph with the super-source node as the source–and every partition is assigned a clocking level based on its position in the longest path from the source. For a graph with E' inter-partition edges, this algorithm runs in exactly $O(E')$ iterations. In the algorithm's next stage, any edge connecting partitions that are separated by more than one clock phase is marked, and the edge is added to an array of bins at every index where a clocking level is missing in the edge.

The number of wire blocks in each bin is calculated based on a predetermined capacity for the wire blocks. This capacity is calculated based on the width of each cell in the grid. Then the inter-partition edges are distributed amongst the wire block, filling one wire block to full capacity before filling the next. It might seem that a better solution would be to evenly distribute the edges to all the wire blocks in the current level. This is not true because the wire blocks with the most number of feed-throughs are placed closer to the logical blocks in the next stage. This minimizes wire length, and hence the number of wire crossings.

IV. Zone Placement Algorithm

A. Placement of k-Layered Bipartite Graph

The logical blocks (obtained from the partitioning stage) and the wire blocks (obtained from post-processing) are placed on an $m \times n$ grid with a given aspect ratio and skew. The individual zone dimensions and the column widths are kept constant to ensure scalability and manufacturability of this design as clocking lines would have to be laid underneath QCA circuits with great precision. The partitions are laid out on the grid, with the cells belonging to the first clocking zone occupying the leftmost cells of the first row of the grid, and the next level occupying the leftmost cell of the next row, etc., until row r. The next level of cells is placed again on row r to the right of the rightmost placed cell amongst the r placed rows. Then, the next level of cells is placed in row $r-1$ and the rest of the cells are placed in a similar fashion until the first row is reached. This process is repeated until all cells are placed (thereby forming a "snake-shape").

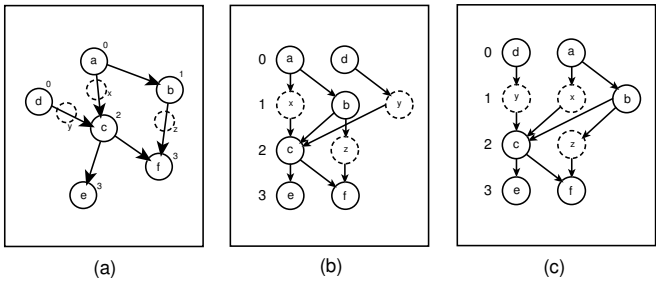

Fig. 4. Illustration of zone placement and wire crossing minimization. (a) zone partitioning with wire block insertion, (b) zone placement, where a zone-level k-layered bipartite graph is embedded onto a 2D space, (c) wire crossing minimization via block re-ordering.

The white nodes are white space that is introduced because of variations in the number of wire and logic blocks among the various clocking levels. The maximum wire length between any two partitions in the grid determines the clock frequency for the entire grid as all partitions are clocked separately. For the first and last rows (where inter-partition edges are between partitions in two different columns), maximum wire length was given more priority as maximum wire length at these end zones can be twice as bad as the maximum wire length between partitions on the same column. An illustration of zone placement and wire crossing minimization is shown in Figure 4.

B. Wire Crossing Minimization

During the next phase, blocks are reordered within each clocking level to minimize inter-partition wire length and wire crossings. Two classes of solutions were applied to minimize the above objectives: an analytical solution that uses a weighted barycenter method, and Simulated Annealing. The analytical method only considers wire crossings since as there is a strong correlation between wire length and the number of wire crossings.

Analytical Solution: A widely used method for minimizing wire crossings (introduced by Sugiyama et al. [5]) is to map the graph into k-layer bipartite graph. The vertices within a layer are then permuted to minimize wire crossings. This method maps well to this problem as we need to only consider the latter part of the problem (the clocking constraint provides the k-layer bipartite graph). Still, even in a two-layer graph, minimizing wire-crossings is NP-hard [5]. Among the many

heuristics proposed, the *barycenter heuristic* [5] has been found to be the best heuristic in the general case for this class of problems. A modified version of the barycenter heuristic was used to accommodate edge weights. Edge weights represent the number of inter-partition edges that exist between the same pair of partitions. The heuristic can be summarized as follows:

$$barycenter(v) = \frac{\sum_N [weight(n) \times position(n)]}{\sum_N weight(n)}$$

where v is the vertex in the variable layer, n is the neighbor in the fixed layer, and N is the set of all neighbors in the fixed layer.

Simulated Annealing: A move is done by randomly choosing a level in the graph and then swapping two randomly chosen partitions $[p_1, p_2]$ in that level in order to minimize the total wire length and wire crossings. In our approach, we initially compute the wire length and wire crossing and incrementally update these values after each move so that the update can be done in $O(m)$ time where m is the number of neighbors for p_i.

V. Experimental Results

Our algorithms were implemented in C++/STL, compiled with gcc v2.96 run on Pentium III 746 MHz machine. The benchmark set consists of seven biggest circuits from ISCAS89 and five biggest circuits from ITC99 suites due to the availability of signal flow information.

Table I shows the zone partition results for our QCA placement. The number of partitions is determined such there are 100 ± 10 majority gates per partition. We set the capacity of each wire block to 200 QCA cells. We compare acyclic FM [4] and QCA zone partitioning in terms of cutsize, white space, and wire blocks needed *after* zone placement. With QCA partition, we see a 20% improvement in cutsize at the cost of a 6% increase in runtime. A new algorithm was implemented to reduce the number of white space. Our new algorithm for reducing the number of white nodes involves moving wire blocks to balance the variation in the number of partitions per clocking level. Although our algorithm results in a 67% decrease in wire nodes and 66% decrease in white nodes, there is a tradeoff in a resulting increase in the number of wire crossings. Since wire crossings have been seen as a much more significant problem, we choose to sacrifice an increase in area for a decrease in the number of wire crossings.

Table II details our zone placement results, where we report placement area, wire length, and wire crossings for the benchmarked circuits. We compare the analytical solution to simulated annealing. Comparing simulated annealing to the analytical solution, we see an 87% decrease in wire length and slight increase in wire crossings.

VI. Conclusions and Ongoing Work

In this paper, we proposed a QCA partitioning and placement problem and present an algorithm that will help to automate the process of design within the constraints imposed by physical scientists. Work to address QCA routing and node duplication for wire crossing minimization are underway.

TABLE I

QCA zone partitioning results.

	Acyclic FM			Zone Partitioner		
name	cut	white	wire	cut	white	wire
b14	2948	151	138	2566	168	127
b15	4839	220	260	4119	144	256
b17	16092	1565	1789	13869	1616	1710
b20	6590	641	519	6033	642	518
b21	6672	599	560	6141	622	557
b22	9473	1146	1097	8518	1158	1098
s13207	2708	143	138	1541	144	137
s15850	3023	257	183	2029	254	181
s35932	7371	875	1014	5361	734	1035
s38417	9375	757	784	5868	775	773
s38584	9940	1319	1155	7139	1307	1095
s5378	1206	34	30	866	34	30
s9234	1903	99	81	1419	104	76
Ave	6318	600	596	5036	592	584
Ratio	1.00	1.00	1.00	0.8	0.99	0.98
time	14646			14509		

TABLE II

QCA zone placement results.

		Analytical		SA-based	
name	area	length	xing	length	xing
b14	20x17	81	67	23	67
b15	20x24	59	90	34	90
b17	69x52	3014	346	305	345
b20	36x36	414	165	99	166
b21	36x37	140	172	100	172
b22	48x50	1091	230	188	230
s13207	18x21	28	9	28	9
s15850	24x23	81	16	11	14
s35932	45x44	1313	64	78	68
s38417	42x43	493	54	48	54
s38584	55x48	1500	102	110	80
s5378	10x10	3	10	2	9
s9234	15x16	15	11	5	11
Ave		633	103	79	101
Ratio		1.00	1.00	0.13	0.98
time		23		661	

Acknowledgment

This research is partially supposed by the National Science Foundation under project number E-21-6TD.

References

[1] C. Lent, B. Isaksen, and M. Lieberman, "Molecular quantum-dot cellular automata," *J. Am. Chem. Soc.*, pp. 1056–1063, 2003.

[2] M. Lieberman, S. Chellamma, B. Varughese, Y. Wang, C. Lent, G. Bernstein, G. Snider, and F. Peiris, "Quantum-dot cellular automata at a molecular scale," *Annals of the New York Academy of Science*, pp. 225–239, 2002.

[3] R. Ravichandran, N. Ladiwala, J. Nguyen, M. Niemier, and S. K. Lim, "Automatic cell placement for quantum-dot cellular automata," in *Proc. Great Lakes Symposum on VLSI*, 2004, pp. 634–639.

[4] J. Cong and S. K. Lim, "Performance driven multiway partitioning," in *Proc. Asia and South Pacific Design Automation Conf.*, 2000, pp. 441–446.

[5] K. Sugiyama, S. Tagawa, and M. Toda, "Methods for visual understanding of hierarchical system structures," *IEEE Trans. Syst. Man,. Cybern*, pp. 109–125, 1981.