Proceedings of 2005 5th IEEE Conference on Nanotechnology
Nagoya, Japan, July 2005

QCA Physical Design With Crossing Minimization

Wook Jin Chung, Brian Smith, and Sung Kyu Lim
School of Electrical and Computer Engineering
Georgia Institute of Technology
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. QCA circuit layout is currently restricted to a single layer with very limited number of wire crossing permitted. Thus, wire crossing minimization is crucial in improving the manufacturability of QCA circuits. The the focus of this paper is to develop the first QCA physical design algorithms for wire crossing minimization.**

I. Introduction

One approach towards nano-scale computing is the quantum dot cellular automata (QCA) concept that represents information in a binary fashion, but replaces a current switch with a cell having a bi-stable charge configuration. A high-level diagram of a "candidate" four-dot metal QCA cell as well as logic gate and wires appear in Figure 1. QCA devices can be realized in metal [1], or with chemical molecules [2]. A wealth of experiments have been conducted with metal-dot QCA with individual devices, logic gates, wires, latches, and clocked devices all having been realized. CAD can help research to move from small circuits to small systems of quantum-dot cellular automata (QCA) devices. At this point, QCA routing is restricted to a single plane with very limited number of wire crossing permitted due to the large size of intersection molecules. Thus, wire crossing minimization is crucial in improving the buildability of QCA layouts. Thus, the focus of this paper is to develop the first QCA physical design algorithms for wire crossing minimization.

II. Problem Formulation

The purpose of zone partitioning is to decompose a circuit such that a single potential modulates the inner-dot barriers in all of the QCA cells that are grouped within a clocking zone. This is a requires step for a QCA layout to function correctly under the 4-phase clocking scheme [3]. Assuming that all partitions (= zone) have similar area, placement of zones becomes a geometric embedding of the partitioned network onto a $m\times n$ grid. In this case, a bipartite graph exists for every pair of neighboring clocking levels. A directed graph $G(V,E)$ is K-layered Bipartite Graph (KLBG) 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. During QCA placement and duplication, we seek

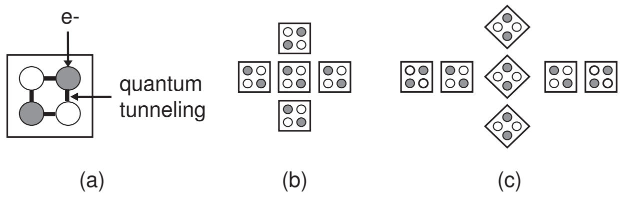

Fig. 1. (a) QCA device, (b) majority gate that can implement AND2 or OR2, (c) horizontal and vertical wires as well as wire crossing.

a KLBG placement of individual logic gates in each zone so that area, wire crossing, and wirelength are minimized. Some logic gates in the original circuit are duplicated in the KLBG placement for further wire crossing minimization. The goal of QCA channel routing is to finish connection among the terminals in every two adjacent layers of the zone-level and cell-level KLBG so that the total wire crossing and channel width are minimized.

During zone partitioning, the length of all reconvergent paths should be of the same length. Two paths p and q in are *reconvergent* if they diverge from and reconverge to the same node. In addition, cycles may exist among partitions as long as their lengths are in multiples of four due to QCA clocking. However, it is hard to enforce this constraint while handling other objectives and constraints. Therefore, we decide to prevent any cycles from forming at the partition level. In addition, it is difficult to maintain the reconvergent path constraint during 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.

III. QCA Physical Design Algorithm

A. Zone Partitioning

An illustration of zone partitioning and wire block insertion is shown in Figure 2. 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 bipartitioning algorithm is performed on adjacent partitions p_i and p_{i+1}. In this case, all cut directed edges should be uni-directional. 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

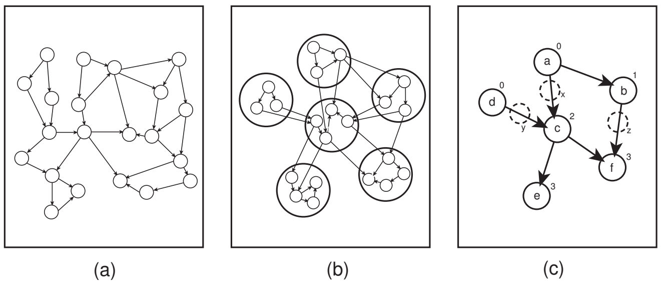

Fig. 2. Illustration of zone partitioning and wire block insertion. (a) directed graph model of input circuit, (b) zone partitioning under acyclicity and reconvergent path constraint, (c) wire block insertion, where the numbers denote the longest path length. The dotted nodes indicate wire blocks.

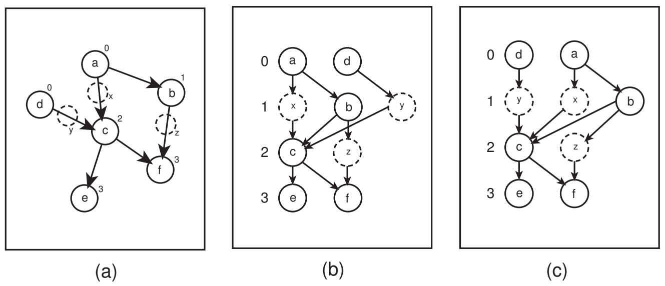

Fig. 3. Illustration of zone placement and wire crossing minimization. (a) zone partitioning with wire block insertion, (b) zone placement, where a zone-level k-layered bipartite graph is embedded onto a 2D space, (c) wire crossing minimization via block re-ordering.

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.

During the post-processing, we fix any remaining clocking problems by inserting and sharing wire blocks, while satisfying wire capacity constraints. 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. In the 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.

B. Zone Placement

An illustration of zone placement and wire crossing minimization is shown in Figure 3. The logic and wire blocks obtained from zone partitioning are placed into a KLBG first. Then these blocks are reordered within each clocking level to minimize inter-partition wirelength and wire crossings. Two classes of solutions were applied to minimize the above objectives: an analytical solution that uses a weighted barycenter method [4], and Simulated Annealing. Wire crossing minimization for a bipartite graph is proven to be NP-complete [4]. The barycenter heuristic fixes the placement of the top-layer while rearranging the bottom layer. The ordering of the bottom layer nodes based on their center of mass gives highly optimized solution. In 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 wirelength and wire crossing. In our approach, we initially compute the wirelength 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. This speedup allows us to explore a greater number of candidate solutions, and as a result, obtain better quality solutions.

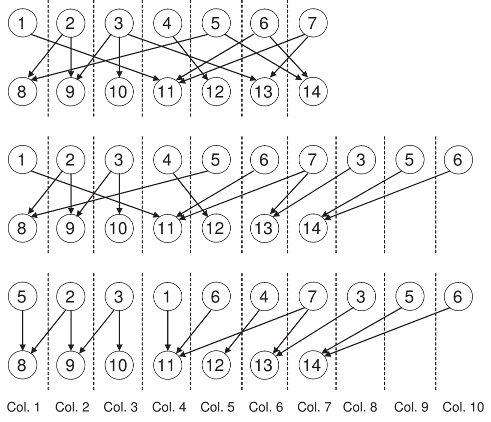

Fig. 4. An example of the FTD algorithm with the threshold value of 2. All nodes whose fan-outs spread out more than the threshold are duplicated. Node 3 originally has fan-outs to column 2, 3, and 6. Since the difference between 6 and 2 is greater than the threshold, it is duplicated. After duplication, the layer is reorganized with Barycenter to minimize crossing.

C. Cell Placement and Duplication

We note that there is a limit to the reduction of wire crossing just by rearranging the logic nodes. Thus, we introduce duplication of these nodes within the process to aid the reduction, which on the other hand also results in a larger circuit. We observe that the longer a wire that connects two nodes, the higher chance it will cause wire crossing. Thus our algorithm focuses on identifying long wires and reducing its length. If a particular node in question had only one output, the crossing its output wire causes can be eliminated via moving the node to the same column that its destination node is placed in. However if a node had two or more fan-outs and the destination nodes were split up far away from each other, then no matter where this node is positioned the total length of all its fan-out wires would remain constant. Thus we measure the difference in the wirelength of any two fan-outs from a single source node, and if this length exceeds the set threshold value, the node would then be duplicated to reduce the overall length. After the duplication process the subject layer would be reorganized via Barycenter. Figure 4 illustrates this process in detail.

With high duplication rates, the unbalance in row length

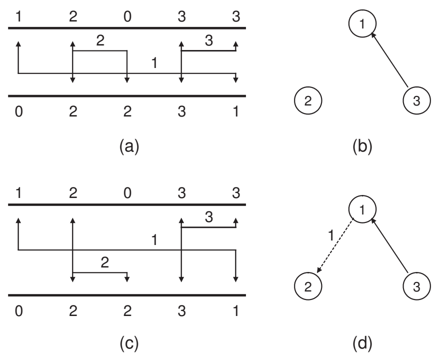

Fig. 5. (a-b) A channel and VCG with minimum channel width but reducible wire crossing. (c) optimum solution for minimizing wire crossing. (d) corresponding modified VCG with an increased longest path.

could become so severe that it becomes impractical to physically lay it out. Thus we post process the circuit with our folding algorithm to produce a square-like shape. The basis upon reducing the width of a layer comes from the fact that bypass nodes are physically thinner than actual nodes as it is just a QCA wire. In our experiment, we set the width ratio between a bypass and a node to 5. Thus by moving a node to another layer and inserting a bypass wire for signal propagation, we can reduce the width of the upper layer by 4. But since all the nodes of the lower layer require their inputs simultaneously, we have to fold the problematic layer onto the newly created layers in between the two original layers.

D. Channel Routing

To reduce wire crossing during channel routing, we add *crossing edges* between nets in the vertical constraint graph (VCG) to enforce the vertical relationship that results in the minimum number of crossing between them. An original VCG is shown in Figure 5(b), and a modified version is shown in Figure 5(d). The resulting channel, shown in Figure 5(c), results in the fewest wire crossings. Each crossing edge is assigned a weight equal to the number of wire crossings saved by orienting the nets according to that vertical orientation. If the VCG is not acyclic after the addition of crossing edges, then the cycles must be removed before track assignment can take place. We remove crossing edges until the VCG is once again acyclic. The weight of crossing edges represents wire crossing reduction, so our goal is to minimize the total weight of the crossing edges removed to make VCG acyclic. This problem is formally defined as follows: Weighted Minimum Feedback Edge Set Problem: Given an edge-weighted directed graph $G(V, E)$ with cycles, the goal is to find a set of edges $A \subset E$ with the minimum total weight such that $G'(V, E-A)$ is acyclic. Since the non-weighted version of the Minimum Feedback Edge Set Problem is NP-complete [5], this weighted version also becomes NP-complete.

We note that a simple heuristic exists to solve the weighted minimum feedback edge set problem: perform DFS once and remove all backward crossing edges found. This will guarantee that the remaining VCG is acyclic. A major shortcoming of this approach, named backEdge, is that it ignores the edge weights and may produce a feedback edge set with a high total weight. Our new heuristic named cycleBreaker first traverses the VCG in depth-first order and collects the cycles detected. When this traversal is complete, it will have added some cycles from each strongly connected subgraph in the VCG to the collection of cycles that must be broken. Because the VCG was acyclic until the crossing edges were added, each cycle must depend on the presence of a crossing edge. Once these cycles are found, the crossing edges of each are keyed in. We use the following function to determine the edge key value:

$$key(e) = \frac{|c(e)| \cdot \sum_{e' \in c(e)} weight(e')}{weight(e)} \tag{1}$$

where $c(e)$ denotes the cycles that contain e. The edge with the highest key is removed from the VCG, and the cycles that contain this edge are removed from the collection of detected cycles. The edges are rekeyed and further cycles are removed until no more detected cycles remain unbroken. In this manner, small-weighted edges that occur in many cycles with other higher-weighted edges are more likely to be targeted for removal to break cycles. As discussed previously, cycleBreaker runs multiple iterations of “DFS plus cycle removal” until all cycles are removed from the VCG.

IV. Experimental Results

Our algorithms were implemented in C++/STL, compiled with gcc v2.96 run on Pentium III 746 MHz machine. 14 combinational circuits were selected from ISCAS benchmark. Table I shows the benchmark characteristics. We report the total number of nodes and edges in the original circuit. We also report the number of levels, columns, and bypass nodes in each corresponding KLBG. Table I shows the crossing reduction by our duplication strategies. dup represents the ratio between the number of nodes between the initial and final circuits, where 1.00 indicates nothing was duplicated. $\%red$ shows the percentage of crossings reduced in respect to the that of the Barycenter, since it is the lowest number of crossings without duplication. Because all processed circuits were put through the post process of folding described in the previous section, the area of a circuit can be represented with only the width; all circuits have a square-like shape so the approximate area would be the width squared. We developed the PATH algorithm, where each path from an output node to all its relevant input nodes is separated and attributed an individual column in the KLBG. Therefore, wire crossing is zero at the cost of huge increase in area. We observe that our duplication algorithm FTD in most cases perform between the upper and lower bounds of the duplication strategies. In other words, the crossings compared to Barycenter is reduced via node duplication and the final area is less than that of PATH.

Table II first shows the theoretical wire crossing lower and upper bound (x-low and x-upp) for each KLBG. We also show the lower bound of channel width (= the longest paths from source to sink in VCG) before crossing edge insertion (t-low). These circuits were routed by four methods, and the results

TABLE I

QCA NODE PLACEMENT AND DUPLICATION RESULTS. WE APPLY FOLDING FOR ALL ALGORITHMS AS A POST PROCESS. THE CROSSING REDUCTION PERCENTAGE AND DUPLICATION RATIO ARE NORMALIZED TO BARYCENTER RESULTS.

	init digraph		KLBG			Barycenter			PATH				FTD			
ckt	#node	#edge	#lyr	#col	#byps	xing	wth	dup	xing	wth	dup	%red	xing	wth	dup	%red
cmb	34	57	7	27	50	22	41	1.00	0	82	2.35	100	0	65	1.73	100
decod	39	84	4	50	48	529	54	1.00	0	100	3.28	100	0	96	3.12	100
pm1	60	98	6	40	56	613	60	1.00	0	89	1.81	100	1	87	1.78	100
i1	74	88	7	38	90	132	62	1.00	0	114	2.31	100	18	89	1.71	86
sct	74	177	5	76	90	3108	84	1.00	0	154	2.56	100	96	124	2.14	97
my_adder	99	130	19	62	600	409	70	1.00	0	345	5.12	100	0	345	5.12	100
i2	238	269	6	201	52	132	62	1.00	0	311	1.43	100	1061	227	1.02	-704
x4	301	716	5	303	518	37967	311	1.00	0	1059	4.57	100	4022	593	2.75	89
k2	317	2893	4	215	35	810343	235	1.00	0	10343	R.O.	100	299112	1252	4.73	63
i5	398	622	8	254	824	4781	262	1.00	0	903	3.52	100	200	695	2.83	96
apex6	472	959	10	423	1692	36406	427	1.00	0	1403	4.84	100	829	800	2.81	98
rot	485	844	12	336	1769	31300	340	1.00	0	4540	11.78	100	3689	1283	3.70	88
frg2	808	2035	10	813	3422	205073	817	1.00	0	4446	8.02	100	11416	2236	4.38	94
pair	1140	2157	20	666	5720	80447	674	1.00	0	7160	R.O.	100	7537	2903	4.73	91
TIME	-		18			21			-				126			

TABLE II

QCA CHANNEL ROUTING RESULTS. WE REPORT THE PERCENTAGE OF WIRE CROSSING THAT INVOLVE DOGLEG (%DOG).

	theoretical bounds			LEF			YK			BE			CB		
ckt	x-low	x-upp	t-low	xing	%dog	track	xing	%dog	track	xing	%dog	track	xing	%dog	track
cmb	0	0	42	0	NA	42	0	NA	42	0	NA	42	0	NA	42
decod	0	55	64	26	0.00	65	26	0.00	65	26	0.00	65	0	NA	77
pm1	1	113	25	56	0.00	28	52	0.00	28	50	0.00	29	1	0.00	43
i1	15	250	105	126	1.59	113	112	1.79	110	97	2.06	119	17	11.76	136
sct	71	4806	48	2447	0.08	82	2180	0.09	79	2025	0.10	99	73	2.74	140
my_adder	0	0	72	0	NA	72	0	NA	72	0	NA	72	0	NA	72
i2	1577	5410	118	3430	0.64	131	3388	0.62	142	3319	0.51	158	2270	0.97	281
x4	2126	113984	215	58672	0.02	372	54528	0.04	402	50656	0.02	523	2242	1.16	1253
k2	80832	967007	279	533694	0.02	858	493600	0.02	874	375464	0.02	1274	92059	0.33	1692
i5	182	55000	508	28443	0.01	596	27260	0.01	624	25602	0.01	669	190	2.11	1317
apex6	582	126406	563	64361	0.01	739	62877	0.01	807	58498	0.01	902	605	1.82	2755
rot	2985	296307	815	150372	0.01	1011	143856	0.01	1108	144636	0.01	1163	3289	1.19	4133
frg2	6783	1310127	799	666234	0.00	1358	630384	0.00	1614	620448	0.00	1878	7219	0.32	7128
pair	6050	1091586	1807	554124	0.01	2348	530954	0.01	2643	527186	0.01	2850	6599	0.61	12516
TIME	-			685			727			808			736		

are shown in Table II. The results of our algorithm are listed as "CB" for our cycleBreaker algorithm in Table II. It was compared to the LEF algorithm (LEF) [6], as well as the algorithm presented by Yoshimura and Kuh (YK) [7]. It was also compared to the backEdge algorithm (BE) (a topological feedback edge set implementation). We note that LEF, YK, and BE generated channel routing solutions well above the circuits theoretical minimum crossing (x-low), but very close to the theoretical minimum track count (t-low). Our channel routing method CB comes much closer to the theoretical minimum crossing count, but produces wider channels.

V. CONCLUSION

QCA promises numerous benefits over today's CMOS circuits. However, due to its technological constraints, the circuit has to be laid on a single layer. This causes wire crossing problems which are too costly to be solved via current intersection molecules. In this article, we presented new algorithms for wire crossing reduction during partitioning, placement, and routing. The outputs of this research are used to generate computationally interesting and optimized designs for experiments by QCA physical scientists.

ACKNOWLEDGMENT

This research has been supported by the National Science Foundation under contract CCF-0404011.

REFERENCES

[1] I. Amlani, A. Orlov, G. Snider, and C. Lent, "Demonstation of a func. quantum-dot cellular automata cell," *J. Vac. Sci. Technology*, pp. 3795–3799, 1998.

[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. Kummamuru, J. Timler, G. Toth, C. Lent, R. Ramasubramaniam, A. Orlov, and G. Bernstein, "Power gain in a quantum-dot cellular automata latch," *Applied Physics Letters*, pp. 1332–1334, 2002.

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

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide To the Theory of NP-Completeness.* Freeman, San Francisco, 1979, pp. 209–210.

[6] A. Hashimoto and S. Stevens, "Wire routing by optimizing channel assignment within large apertures," in *Proc. ACM Design Automation Conf.*, 1971, pp. 155–169.

[7] T. Yoshimura and E. Kuh, "Efficient algorithms for channel routing," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pp. 25–35, 1982.