A Hierarchical Layout Generation Method for Quantum Circuits

Mina Chookhachizadeh Moghadam, Naser Mohammadzadeh, Mehdi Sedighi, Morteza Saheb Zamani
Department of Computer Engineering and Information Technology
Amirkabir University of Technology
Tehran, Iran
mina_moghadam, mohammadzadeh, msedighi, szamani@aut.ac.ir

***Abstract*— Quantum circuit design flow consists of two main tasks: synthesis and physical design. Synthesis converts the design description into a technology-dependent netlist and then, physical design takes the fixed netlist, produces the layout, and schedules the netlist on the layout. Quantum physical design problem is intractable. This process can be divided into two main processes: scheduling and layout generation. Some heuristic techniques have been proposed for the layout generation. These techniques do not produce good layouts for large netlists in terms of latency. Focusing on this issue, in this paper, a hierarchical layout generation algorithm is proposed that generates better layouts in terms of latency. Ion trap is used as the underlying technology in this paper. Experimental results show that the proposed algorithm decreases the average latency of quantum circuits by about 22% for the attempted benchmarks.**

Keywords— Quantum Circuits, Physical Design, Hierarchical Layout Generation, Ion Trap Technology

I. INTRODUCTION

Quantum computing is an active research topic. Various aspects of this field have been focused on for the last decade. Among them, one can name the theoretical analysis of quantum circuits, design of efficient algorithms to perform computation on quantum machines that would have been extremely difficult on conventional machines, synthesis of quantum circuits, and physical design of these circuits based on a given technology. The latter usually involves the process of generating an optimal layout and producing an appropriate scheduling of tasks that meets some given constraints such as the overall area taken by the circuit, or its overall latency. The focus of this paper is on the layout generation process for a given netlist.

There have been previous works that deal with quantum circuit scheduling and layout generation. However, it appears that the researchers were originally more concerned with reaching an appropriate architecture for quantum circuits than solving the specific problems of scheduling those circuits and creating a layout for them. This is because those works use a trivial hand-generated grid-based layout to obtain a suitable architecture that meet a given latency constraint [1][2]. In one subsequent contribution, the focus shifted from architecture to scheduling but layout generation still remained the secondary issue in the physical design process. Hence, once again, a simple manually-created layout was used to base the optimal schedule on it [3]. In a later contribution three automated layout generation schemes were proposed [4]. The first two schemes consider the two processes of scheduling and layout generation independently. However, the third one involves iterative refinements of the original schedule and layout until a local optimum is reached. Furthermore, unlike the first two approaches, the latter one does seem to be scalable as the size of the input circuit grows.

In this paper, a scalable physical design method is proposed that solves the scheduling and layout generation problems in an intertwined fashion. This method uses a hierarchical approach to overcome the complexity of the two problems. The proposed method takes a data flow graph (DFG) and partitions it into smaller parts with a desired granularity. For each part of the partitioned DFG, a sub-layout is generated considering not only the quality of the individual sub-layout, but also the quality of the overall layout that is generated after merging the sub-layouts. All proposed steps have been organized with the aim of achieving the minimum latency.

The rest of this paper is organized as follows: an overview of previous work is presented in Section II followed by an overview of ion trap technology along with the definition of terms used in layout generation algorithm in Section III. The proposed hierarchical layout generation and scheduling method is presented in Section IV. Section V contains the experimental results. Section 6 concludes the paper.

II. RELATED WORK

Quantum physical design has been a topic of research for the last decade. The pioneer works were naturally concerned with a design flow to map a high-level model of a circuit into a physical model that considers both the spatial and temporal aspects of the design. In those early works, to handle the complexities of the temporal domain, a rudimentary spatial domain foundation was often used as the basis. For instance, in [2], in the context of reaching a CAD flow, a simple grid-based layout using H-Tree cells is employed as the spatial basis. In another related work, the same simple grid-based layout is generated with the aim of organizing a desired architecture considering some latency constraints [7]. Even though a different cell is used in this research, but still, a trivial grid-based quantum layout is utilized as a platform to implement the architecture [1]. Further improvements of the proposed architecture are still based on the same layout with no difference in its physical characteristics [5]. Later, the same group of researchers shifted their focus from coming up with a fault-tolerant architecture into reaching an optimum scheduling [3]. In this work an algorithm has been developed to schedule instructions on a given physical layout to obtain the minimum

978-1-4799-0565-2/13/$31.00 ©2013 IEEE

IEEE COMPUTER SOCIETY
IEEE

latency. However, like the previous contributions, to handle the complexities of the temporal domain, i.e., the scheduling problem, the paper does not deal with the issue of optimum physical layout generation. As this brief review shows, the previous approaches paid less attention to the layout generation problem compared with the other aspects of the design. The grid-based layouts that they based their work on are usually constructed by simply tiling some basic predefined cells chosen according to the actual design target, i.e., a specific architecture or a desired scheduling.

As the algorithms dealing with the architectural concerns or scheduling issues were making progress, the need for improvements in the spatial domain solutions became more evident. In other words, the problem of optimum layout generation, especially for larger circuits, moved to the forefront of the research efforts. As such, three heuristic methods were proposed to automate all of the tasks involved in layout generation and its associated control logic [4]. The first one, a grid-based layout generation method, like the previous approaches, generates a layout using predefined building blocks. However, unlike the earlier grid-based approaches, this one exhaustively explores all possible combinations of the patterns and sizes of primitive cells to obtain the best result. Due to its exhaustive nature, it has been shown that this method is not scalable [4]. To address this shortcoming, the authors have presented two other heuristic methods with polynomial time complexity that can generate layout for larger quantum circuits. The first one, called greedy place and route, starts with one gate location per each input qubit with no interconnection among them. Subsequently, the scheduler introduced in [3] is used to iteratively connect the gate locations and refine layout until the circuit is successfully completed. This method can be useful for simple circuits but it loses its effectiveness as the size of the circuit grows [4]. For larger circuits, the authors propose a method, called dataflow-based, which is still heuristic but it features a more global view of the input circuit in order to avoid local minimums. This method provides better latency results for larger circuits [4].

One of the missing points in reaching an optimum layout is the fact that the temporal and spatial domains have been traditionally considered separately. To address this point, in some previous works, the authors introduced the concept of physical synthesis in which the mutual interactions of scheduling and layout are considered in order to reduce the overall latency of the circuit [6], [8], [9]. While this flow showed superior results as compared with the earlier approaches, it could still be considered a post-process technique since it relied on the dataflow-based layout generation approach of [4] to create the initial layout before refining. Our research showed that the structure of the initial layout has a considerable impact on the improvements that the physical synthesis approach can reach. Therefore, it is plausible to assume that an original layout generation method may produce even better results.

There seems to be an emerging trend towards generation and implementation of larger quantum circuits [2], [4]. This trend would naturally require algorithms that can handle larger quantum circuits as well as appropriate benchmarks to evaluate them. Recently, there have been efforts to address the latter need, as evidenced by a new algorithm proposed in [10]. This algorithm generates scalable and robust random quantum benchmarks, with no limitation in the number of gates or qubits, using some mathematical rules. However, as for actual physical design algorithms, it seems that there is a significant need for scalable solutions that can design large quantum circuits efficiently. The main challenge in reaching such an algorithm is that the complexity of layout generation task and the search space that needs to be explored to obtain an optimal result increase dramatically as the size of quantum circuits increases. Given the fact that the existing physical design algorithms, especially layout generation methods, look at the entire design as a flat architecture, it seems plausible to assume that one way to tackle this challenge is to come up with a hierarchical approach. In other words, a hierarchical view might provide opportunities to handle the complexity of layout generation and scheduling processes using a divide and conquer approach to obtain the better physical layout in terms of latency and area. This notion is one of the main contributions of this paper. Another important feature of the proposed approach is that it embodies a layout generation algorithm along with an instruction scheduling scheme that are performed concurrently. At each stage of the layout generation process, no decision is made without considering its impact on the scheduling. As will be shown subsequently, the proposed method can provide superior results in term of latency.

III. TECHNOLOGY ABSTRACTION

While certain aspects of physical design algorithms can be technology-independent, an accurate layout design requires a model of the underlying technology for quantum circuit implementation. In this paper, the ion trap technology [11], was chosen as the underlying technology. This technology is potentially compatible with scalable applications [12]. In this technology a qubit is represented by an ion. A gate location is a location in which a trapped ion can be operated upon via a modulated laser beam. A channel is a path that qubits can move in it. In the proposed method, a library of basic macroblocks, shown in Fig. 1, was used to construct physical layouts. Interested readers may consult reference [4] to find a detailed description of this library.

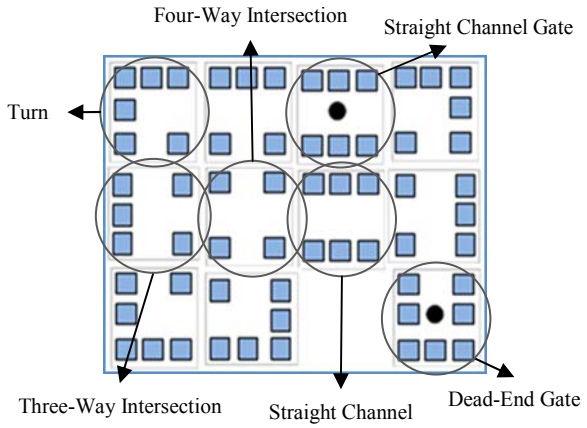

Fig. 1. A simple layout in ion-trap technology. The six main macroblocks of the used library [4] are shown at various parts of the layout.

IV. HIERARCHICAL LAYOUT GENERATION APPROACH

Finding the best layout in terms of area and latency is an intractable problem in ion trap technology [2]. Therefore, some heuristic approaches should be followed to manage the complexity of the problem. One of those approaches is using the hierarchal design methodology that is widely used in classical circuits design.

The first step in developing a hierarchical method is to break down the problem into smaller parts. A global solution can then be formed using the local solutions. It seems that for the specific problem of hierarchical layout generation a top-down approach can be devised to best handle the complexities of the problem. The main idea is that at higher levels only some coarse decisions are made regarding the general location of gates and global routing. As we gradually progress down the hierarchical levels, more detailed decisions are made.

The top-down flow of the proposed hierarchical layout generation method has been shown in Fig. 2. The flow takes a DFG and divides it into sub-DFGs. The global decisions about gate locations and channel routings are made during the partitioning step. A layout is constructed for each sub-DFG during the sub-layout generation step. These sub-layouts are then fed into the merging step in which a global view is used to efficiently connect the constructed sub-layouts and generate an initial layout for the overall circuit. After generating the initial layout, the place and routing information stored in the previous steps are used to reach an instruction schedule. If a scheduling is found, the final layout is generated. However, if the scheduling process fails, the layout is modified. In the following sections, the details of each step will be described.

A. Prtitioning

The partitioning step takes a DFG as its input and partitions it into sub-DFGs. It was empirically discovered that having a sub-DFG with 4 to 7 nodes usually leads to better results. Each edge in the DFG is mapped to a routing path on the corresponding physical layout. If two connected nodes are located in two different sub-DFGs, the routing path connecting them may be longer than when they are in the same DFG. This potentially increases the overall latency. Therefore, the number of edges connecting the resultant sub-DFGs should be minimized to decrease the latency overhead. However, by considering only the minimum cut factor, the subsequent merging procedure may not result in a favorable final layout in terms of area and latency. This is because that the sub-layouts for each sub-DFG will be constructed independently without considering the global placement information. Based on these facts, a min-cut placement-aware partitioning approach [13], is considered as the basis of the partitioning algorithm. Furthermore, by using the terminal propagation approach [14] in the proposed algorithm, it becomes possible to partition a DFG considering the requirements of the next step, i.e., placement. Following this approach causes the adjacent nodes[1] located in different sub-DFGs to be placed as close as possible to each other on the physical layout. In other words, considering the interoperability between the partitioning and merging steps, some global placement and routing decisions

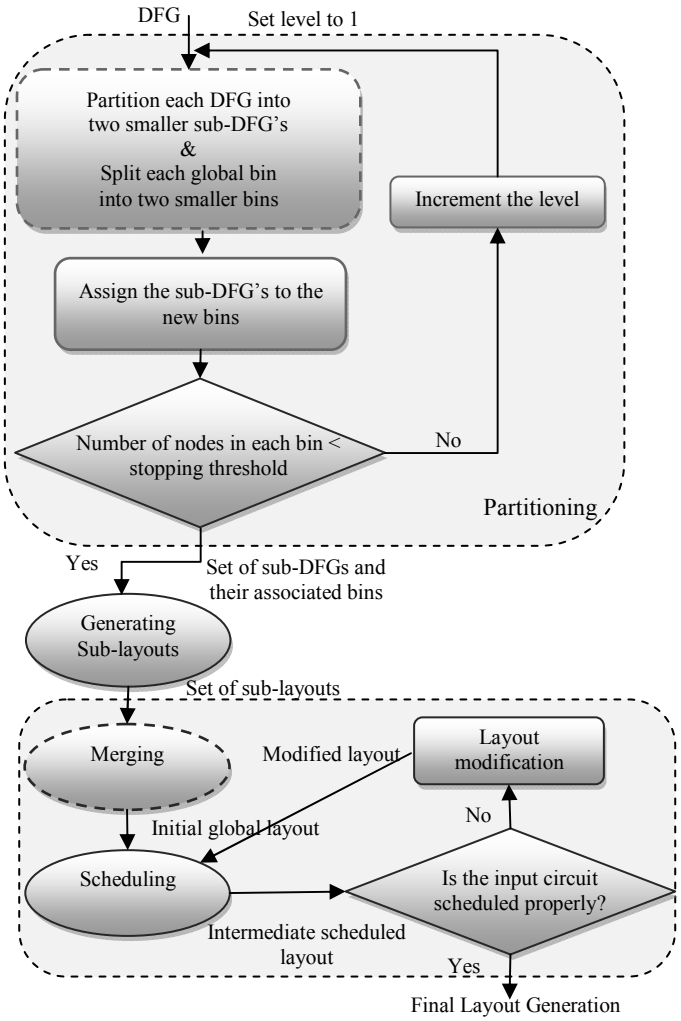

Fig. 2. The flow of the proposed hierarchical layout generation approach.

are made during the partitioning process using the proposed top-down approach.

Fig. 3 shows the terminal propagation approach [14] used in the proposed placement-aware partitioning algorithm. The partitioning process will be continued until the number of nodes in each physical area reaches to the stopping threshold. The outcome of this partitioning process is a set of sub-DFGs as well as their corresponding physical areas, called *bin* (Fig. 3). Each bin includes up to 7 nodes and is mapped to a sub-layout that will be generated for the corresponding sub-DFG in the following step. During the partitioning process, the propagated terminals are considered as dummy nodes. These nodes are not operational and only are used to propagate terminals and store interconnections between two bins and the corresponding sub-layouts.

B. Generating Sub-layouts

After partitioning the initial DFG into sub-DFGs and assigning gates to the bins, a sub-layout is generated for each sub-DFG. For each sub-DFG, a 4*4 bin is used to construct the corresponding sub-layout. Each bin includes 16 fine units called *micro-bin*. One of the basic macroblocks shown in

[1] Each two nodes connected by an edge in the input DFG

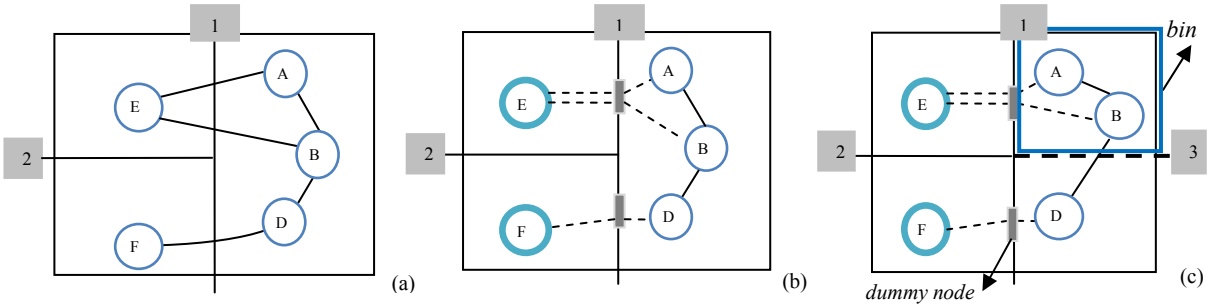

Fig. 3. Terminal propagation approach used in the placement-aware partitioning algorithm [14]. a) The cut number 1 partitions nodes into two parts located on the left and right sides. Then the cut number 2 partitions nodes located on the left side into two new parts. b) Nodes E and F propagate terminals (dummy nodes) for the nodes connected to them and located on the right side. c) Nodes located on the right side are partitioned considering both internal and dummy connections (cut number 3).

Fig.1 is assigned to each micro-bin. To find an appropriate macroblock for each micro-bin and also the best gate location for each gate, mostly the local information i.e. the internal communication links between nodes are used. However, to take into account the global information in sub-layout generation process, the external connections are involved in decisions making. These external connections are taken into account via dummy nodes formed in the partitioning step. Considering these connections is useful to generate better local sub-layouts that potentially can be merged together to construct a better global layout. To consider both internal and external connections in sub-layout generation step, an order-based algorithm is proposed.

1) Priority Assignment

In this step, a priority number is assigned to each node located in each bin. To consider the global connections between parts (bins), a higher priority should be assigned to the nodes that are connected to the nodes located in the other bins. Similarly, the lower priority is assigned to the nodes that only are connected to the nodes located in the same bin. In other words, the highest priority number is assigned to the nodes with all dummy predecessors or successors. These nodes can find their best places according to their corresponding dummy nodes, because the places of dummy nodes are determined during the partitioning step. For two-qubit gates that have one dummy successor or predecessor, the priority can be assigned if and only if the place of the other predecessor or successor has been determined before. Otherwise, in some cases the gate location selected for such gates may increase the latency (Fig. 4). The priority assignment algorithm is shown in Fig. 5.

2) Order-based Placement within Bins

When the priority assignment process was completed, the local placement step should be done for the nodes located in each bin based on their priorities. To do this, all unoccupied micro-bins are explored to find the feasible cases for the node with the highest priority, as will be defined later. Then, among all of the feasible micro-bins, a micro-bin with the minimum cost is selected to be assigned to the current node.

To verify the feasibility of a micro-bin in this process, each micro-bin is known via the type and the orientation of the macroblock used to initialize it. A micro-bin could be assigned to a node, if the two following conditions are satisfied:

- **Gate location feasibility:** the micro-bin should be initialized to a *dead-end gate* or a *straight channel gate* macroblock or could be changed to one of these two types with the proper orientation to route qubits to its fixed predecessors or successors. This orientation should also satisfy the qubit movements between nodes placed in the previous steps.
- **Path feasibility:** there should be the possibility of constructing path(s) between this micro-bin and the micro-bins of fixed predecessor or successor nodes. All of the micro-bins located on the routing paths may be initialized with a proper type and orientation of the

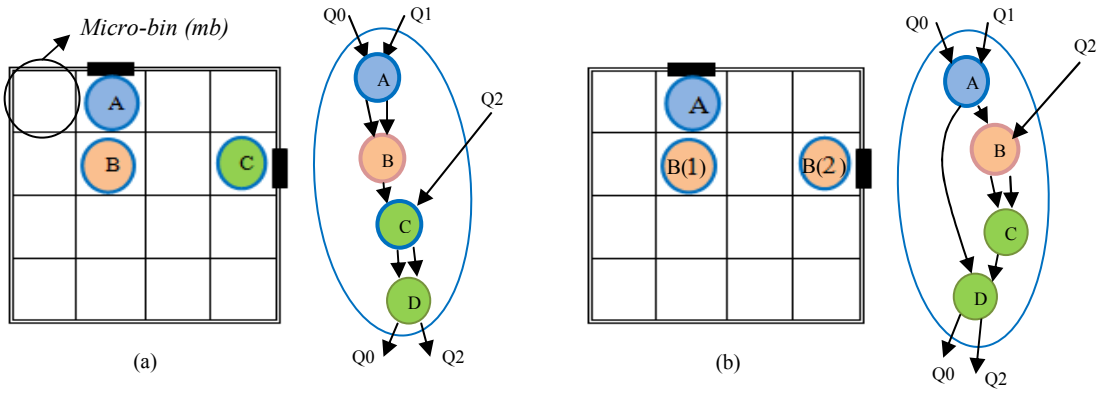

Fig. 4. Two sub-DFGs with the corresponding sub-layouts. a) The order (a, b, c) and (a, c, b) are led to the same results in the local placement process. b) The order (a, b) assigns the place "1" for node "B" and the order (b, a) assigns the place "2" to it that causes more latency overhead in the path from node A to B.

```
Set the state of all dummy nodes to fixed state;
Set the state of the other nodes to unfixed state;
1. Set P to the maximum value;
2. while (there is a node that has no priority number, n)
      {
       For (each node)
       {
         If (((the state of node's predecessor1) is fixed)
      and
       (the state of node's predecessor2) is fixed))
                Set flag1=true;
         If (((the state of node's successor1) is fixed)
      and
       (the state of node's successor2) is fixed))
                Set flag2=true;
       If (flag1 or flag2 is true)
         {
            node. priority = P;
            node. state = fixed;
         }
       }// end of for
       P--;
      }//end of while
```

Fig. 5. Priority assignment algorithm for the nodes located in each bin.

basic macroblocks to route qubits properly, or can be changed to a proper one dynamically. In this paper, routing paths are considered in three categories: the straight path that all of its micro-bins are located either in the same column or in the same row with an appropriate orientation; the one-turn path that contains only one turn macroblock; and the two-turn path that includes two turn macroblocks. Since the other cases suffer from more turns and the higher latency overhead, they are not considered in the search space.

After finding all feasible micro-bins that can be used to place a node with the higher priority, the one with the minimum cost is selected. This cost is estimated according to the latency of the paths between the current node and its fixed predecessors or successors. The physical parameters used in this paper are shown in TABLE I. . The cost of a path does not merely depend on the Manhattan distance between its two ends. Because moving through a right-angle turn takes longer than straight movements over the same distance [15]. Therefore, the proposed algorithm attempts to choose a path with less number of turns between three possible types. This greedy approach may lead to a dead lock. In such situation no proper micro-bin is remained to be chosen for further nodes. To resolve a deadlock, a trace back should be done to find new locations for some of the previous.

The proposed scenario is applied to all of the nodes located in each bin. During the sub-layouts generation process, each node finds its local place and routing paths to its parents and children. This placement and routing information will be used to find the global places of nodes and the global routing paths between them on the global layout. This information also is used to form the corresponding scheduling algorithm.

TABLE I. THE LATENCY PARAMETER VALUES FOR VARIOUS PHYSICAL OPERATIONS IN ION-TRAP TECHNOLOGY [16]

Physical Operation	***Latency Symbol***	***Latency (μs)***
One-Qubit Gate/ Straight Move	t_{1q} / t_{move}	1
Two-Qubit Gate/ Turn	t_{2q} / t_{turn}	10

C. Final Layout Generation

In this step, the sub-layouts generated in the previous steps are merged together to build the final physical layout. This merging process tries to determine the global places of nodes and the global routing paths between connected nodes located in different bins. Although the terminal propagation approach tries to place the adjacent nodes as close as possible to each other, in a few cases the connected nodes may be placed in non-adjacent bins. In such cases, each node has the specified path to its corresponding dummy node but the path between the two dummy nodes should have been determined. To do this, the layout is updated by adding new rows and columns of channel macroblocks, and the routing paths between non-adjacent bins are completed.

1) Scheduling Instructions and Layout Modification

Since the global view has been considered during the partitioning and sub-layout generation steps, the proposed merging process potentially result in an efficient initial layout. However, it may sometimes generate a layout that cannot be scheduled properly. Therefore, the proposed scheduling and layout modification procedures are applied to the initial layout to make some modifications and generate a proper scheduled layout.

In the proposed scheduling algorithm, a customized version of ASAP algorithm has been used. In this algorithm, the data-flow and timing-dependency information are used to make the best decision in each step of the scheduling process. The data flow is extracted from the initial DFG and the timing information is estimated by using the placement and routing information that can be obtained from the intermediate physical layout. To address this aim in the proposed algorithm, a timed-DFG that includes both time and data dependency information is used.

2) Resolving Conflicts between Qubits

By using timed-DFG in each time-slot of the scheduling process, the nodes that their predecessors have been scheduled before estimate their start-times. Considering this initial estimation of gates' start-times, two main types of conflict may be discovered between each two qubits during the instruction scheduling process. These conflicts are called moving conflict and stationary conflict. The former may occur between two moving qubits, and the latter may happen between a moving qubit and a stationary one waiting in the next gate location to be used by the corresponding gate. Fig. 6 shows an example of moving and stationary conflicts. In the first case, two congested qubits can be the inputs of one gate or two different gates. In both cases, these two qubits use some shared micro-bins on their paths toward the gate locations of their next destinations. To resolve the conflict through the scheduling process, one of the qubits should wait at the beginning of the shared part of the paths. The waiting time is measured based on

the delay of the slowest micro-bin in that shared path. Moreover, to remove the latter type of conflicts, the initial layout should be updated via building new routing paths. After modifying the intermediate layout, the resulted layout is given to the scheduling part to be checked for new probable moving conflicts. These steps, i.e. layout modification and instruction scheduling are iterated to resolve all of the probable stationary and moving conflicts.

V. EXPERIMENTAL RESULTS

In this paper, some QECC benchmark circuits [17] have been used to evaluate the proposed hierarchical layout generation method. TABLE II. shows the latency of the benchmark circuits resulting from the best method in the literature [4] and ours. Physical latencies shown in TABLE I. are used for the gates and for the two types of move operations in ion trap technology [16]. The first three columns contain the names of the benchmarks, number of their gates and qubits, respectively. The first column under the column "latency" shows the results achieved by the best previous layout generation method [4] and the later one shows the results achieved by the proposed hierarchal layout generation method. The column "imp" shows the improvement achieved by using our method compared with the best previous one. As shown, the achieved improvement is about 22.25% on average. The columns "Prior data-flow-based approach" and ''Our Approach" under ''Runtime'' show the runtimes of the prior best method and our approach, respectively. The run-time overhead shown in column "Overhead" is about 20.14% on average. Furthermore, the column "area" shows the area parameter of generated physical layouts. For each benchmark,

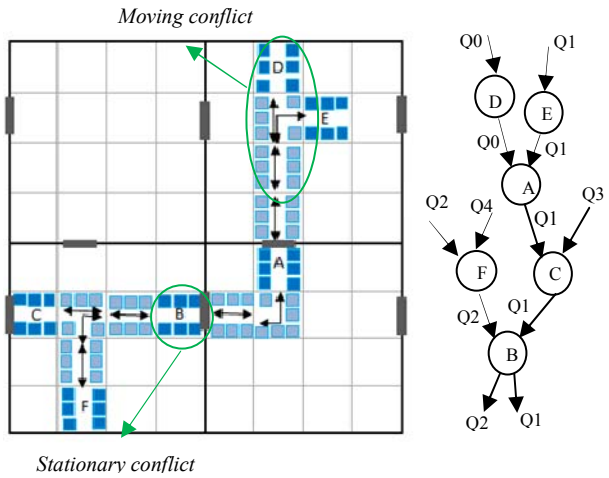

Fig. 6. Moving and Stationary conflicts between qubits

area is estimated based on the number of micro-bins that are used in the layout bounding box to construct it. The first column under the column "area" shows the results achieved by the best previous algorithm [4] and the later one shows the results achieved by the proposed method. The column "imp" shows the improvement achieved by using our method compared with the best previous one. This improvement is about "46.52%" on average. The achieved improvements in terms of latency and area show that the complexity of layout generation process was handled effectively using the hierarchical approach.

TABLE II. THE LATENCY, RUNTIME AND AREA OF BENCHMARKS ACHIEVED BY THE PROPOSED METHOD COMPARED WITH THE BEST IN THE LITERATURE.

Benchmark				Latency (µs)			Run time (s)			Area (# of micro-bins)		
#	L1 encode circuit name	# of gates	# of qubits	Prior data-flow approach [4]	Our approach	Imp (%)	Prior data-flow approach	Our approach	Over head (%)	Prior data-flow approach	Our approach	Imp (%)
1	[7,1,3]	18	7	331	285	13.90	0.45	0.61	35.5	190	80	57.89
2	[10,3,3]	44	10	960	772	19.58	0.53	0.72	35.8	364	196	45.60
3	[13,1,5]	64	13	1281	1020	20.37	0.88	0.90	2.3	544	432	20.96
4	[16,3,5]	89	16	1757	1442	17.93	0.92	1.27	38	754	420	44.56
5	[18,1,7]	102	18	1612	1335	17.18	1.1	1.40	27.2	931	412	55.76
6	[21,1,7]	140	21	3068	2447	20.24	1.6	2.05	28.1	1276	792	37.93
7	[25,1,9]	168	25	4491	3710	17.39	1.7	2.02	18.8	1540	836	45.71
8	[24,3,7]	205	24	4584	3748	18.24	1.9	2.09	10	1984	724	63.61
9	[27,1,9]	244	27	5687	4375	23.07	3.9	4.08	3.8	2356	1412	40.07
10	[33,1,9]	316	33	9026	6673	26.07	4.0	4.29	7.2	2914	1484	49.14
11	[31,11,6]	339	31	7362	5211	29.22	5.1	6.00	17.6	3355	1388	58.69
12	[36,7,6]	395	36	9805	7100	27.59	5.9	6.21	5.3	3520	1584	55.06
13	[30,20,4]	411	30	8626	6175	28.41	5.6	6.50	16.1	3654	1672	54.24
14	[40,3,10]	483	40	11405	7725	32.26	6.1	8.32	36.39	4480	3308	26.21
Average						22.25			20.24			46.52

VI. CONCLUSION

In this paper, a hierarchical method was proposed to generate physical layouts for quantum circuits. To develop the flow, a top-down approach was followed. Moreover, a new scheduling algorithm was proposed that is integrated into the layout generation process. Developing a layout generation algorithm along with an instruction scheduling in the proposed hierarchical method resulted in efficient physical layouts. The experimental results show that the method can explore the search space more efficient than the previous approaches. The proposed approach improves the latency of quantum circuits by up to 32.26% for the attempted benchmarks.

REFERENCES

[1] T. Metodi, D. Thaker, A. Cross, F. Chong, and I. Chuang, "A quantum logic array microarchitecture: scalable quantum data movement and computation," Proceedings of the 38th International Symposium on Microarchitecture, pp. 305-318, 2005.

[2] K. Svore, A. Cross, A. Aho, I. Chuang, and I. Markov, "Toward a software architecture for quantum computing design tools," Proceedings of the 2nd International Workshop on Quantum Programming Languages, pp. 145-162, 2004.

[3] T. Metodi, D. Thaker, A. Cross, F. Chong, and I. Chuang, "Scheduling physical operations in a quantum information processor," Proceedings of Society of Photo-Optical Instrumentation Engineers, pp. 6244: 62440T, 2006.

[4] M. Whitney, N. Isailovic, Y. Patel, and J. Kubiatowicz, "Automated generation of layout and control for quantum circuits," Proceedings of the 4th international conference on Computing Frontiers, pp. 83-94, 2007.

[5] D. Thaker, T. Metodi, A. Cross, I. Chuang, F. Chong, "Quantum memory hierarchies: efficient design to match available parallelism in quantum computing," Proceedings of the 33th International Symposium on Computer Architecture, pp. 378-390, 2006.

[6] N. Mohammadzadeh, M. Sedighi, and M. Saheb Zamani, "Quantum physical synthesis: improving physical design by netlist modifications," Microelectronics, vol. 4, pp. 219–230, 2010.

[7] S. Balensiefer, L. Kregor-Stickles, and M. Oskin, "An evaluation framework and instruction set architecture for Ion-trap based quantum micro-architectures," Proceeding of 32nd Annual International Symposium on Computer Architecture, pp. 186-196, 2005.

[8] N. Mohammadzadeh, M. Saheb Zamani, and M. Sedighi, "Improving latency of quantum circuits By gate exchanging," Proceedings of the 12th IEEE Euromicro Conference on Digital System Design, pp. 67-73, 2009.

[9] N. Mohammadzedeh, M. Sedighi, and M. Saheb Zamani, "Auxiliary qubit selection: a physical synthesis technique for quantum circuits," Quantum Information Processing, vol. 9, pp. 139-154, 2010.

[10] E. Magesan, J. Gambetta, and J. Emerson, "Scalable and robust randomized benchmarking of quantum processes," Physical Review Letters, vol. 106, pp. 180504, 2011.

[11] H. Haffner, C. Roos, and R. Blatt, "Quantum computing with trapped Ions," Physics Reports, vol. 469, pp. 155-203, 2008.

[12] D. Kielpinski, C. Monroe, and D. Wineland, "Architecture for a large-scale quantum computer," Nature, vol. 417, pp. 709-711, 2002.

[13] M. Breuer, "A class of min-cut placement algorithms," Proceedings of ACM/IEEE Design Automation Conference, pp. 284-290, 1977.

[14] A. Dunlop, and B. Kernighan, "A procedure for placement of standard-cell VLSI circuits," IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, vol. 4, pp. 92-98, 1985.

[15] C. Pearson, D. Leibrandt, W. Bakr, W. Mallard, K. Brow et al., "Experimental investigation of planar Ion traps," Physical Review Letters, vol. 73, pp. 03230.1-032307.12, 2006.

[16] M. Whitney, N. Isailovic, Y. Patel, J. Kubiatowicz, "A fault tolerant, area efficient architecture for Shor's factoring algorithm," Proceeding of 36th international symposium on computer architecture, 2009.

[17] M. Grassl, "Circuits for quantum error-correction codes," available online, https://iaks-www.ira.uka.de/home/grassl/QECC/index.html, visited on 2012-3-1.