Layout Optimization in Quantum Circuit using Simulated Annealing

Somayyeh Shahbazi, and Naser Mohammadzadeh

Abstract- The placement is one of the main steps of the quantum physical design process. It has prominent effects on the total area and the delay of a circuit. However, this process is an NP-complete problem. Therefore, we need some heuristics to solve the placement problem. Focusing on this issue, in this paper a placement algorithm is proposed based on Simulated Annealing heuristic. Experimental results show that the proposed technique decreases the average area of quantum circuits up to 43% against maximum 3% latency penalty for the attempted benchmarks.

Keywords- Quantum Circuits, Placement, Simulated Annealing (SA), Ion Trap Technology.

I. Introduction

IN layout generation, one of the main challenges of accurately modeling of large-scale quantum architecture is the ability to generate a layout for a quantum application by taking into account the area and the maximum parallelism that can be exploited [1]. Most of the important works [2]-[6] performed on quantum algorithms and its physical platform have focused on the constant layout and a few numbers of works [1][7]-[10] have been done in designing and producing layouts for quantum circuits and improving of them. In most of methods, the layout is generated and improved manually. Although the proposed methods in the primary steps and in an overall point of view seem efficient, but when the scale of the quantum circuit gets bigger, producing and improving of layouts manually seems to be impractical[11].

The circuit model for quantum computation allows us to represent any application as a sequence of logic gates [4],[12] to which we will refer to as operations or instructions interchangeably. Several major differences between quantum logic circuits and their classical counterparts make accurate placement and scheduling of a sequence of quantum operations a challenging task. First, the inability to copy quantum data[13] makes it impossible for the data to be transmitted on a wire or distributed to multiple destinations without error. In addition, the no-copy rule forces multi-qubit operations to bring the participating qubits physically together across some distance by traversing an empty channel as in the ion-trap technology [14][15] or by successive swapping with adjacent qubits as in the Kane silicon-based technology [16],[17]. The second important difference is that operations can occur simultaneously, and any or all physical qubits used in those operations may need to be physically transported. The choice of which qubits to transport affects the time of execution and the reliability of the application. The third major difference is that unlike classical data, quantum data is inherently very unstable. Experimental failure rates of multi-qubit operations for ion-traps are as high as 3% [18].

In this paper, we propose a heuristic for layout optimization of quantum circuits which takes a netlist and a physical layout [10], and optimizes the layout. We use a modified and improved version of dataflow-based layout generation approach proposed by Withney et al. [10].

By studying the physical limits of quantum computing, we can prevent from some problems such as obstruction and congestion between qubits which lead to inordinate increase of execute time and find the best location for placement of gates. Furthermore, qubit movement in channels and gate locations have direct effect on the execution time that is one of the main metrics in quantum circuit design.

The rest of this paper is organized as follows: We describe some basic concepts about quantum computation and ion trap technology in section II, followed by an overview of prior work in the field in Section III. Section IV contains our CAD flow and placement strategy. Experimental results are mentioned in Section V. Finally, Section VI concludes the paper.

II. Basic Concepts

A. Quantum Computation

Quantum computation and quantum information is the study of the information processing tasks that can be accomplished using quantum mechanical systems. Quantum mechanics is a mathematical framework or set of rules for the construction of physical theories. Quantum computation certainly offers challenges aplenty to physicists, but it is perhaps a little subtle what quantum computation and quantum information offers to physics in the long term. It is built upon an analogous concept, the quantum bit, or qubit for short. The beauty of treating qubits as abstract entities is that it gives us the freedom to construct a general theory of quantum computation and quantum information which does not depend upon a specific system for its realization [19].

Changes occurring to a quantum state can be described

Somayyeh Shahbazi is with the Department of Computer, Arak Branch, Islamic Azad University, Arak, Iran (corresponding author's e-mail: mehr.ssh@gmail.com).

Naser Mohammadzadeh is with the Computer Engineering Department, Shahed University, Tehran, Iran (e-mail: mohammadzadeh@shahed.ac.ir).

http://dx.doi.org/10.15242/IIE.E0214516

using the language of quantum computation. Analogous to the way a classical computer is built from an electrical circuit containing wires and logic gates, a quantum computer is built from a quantum circuit containing wires and elementary quantum gates to carry around and manipulate the quantum information [19].

B. Ion Trap Technology Abstraction

Ion trap technology [20],[21] was chosen as the underlying technology to study our proposed flow. Trapped ions have shown good potential for scalability [7]. Ion trap technology has been physically realized using universal elements for quantum computation with a clear scalable model[22]. Fig. 1 shows the main elements of quantum computing with ion traps [23].

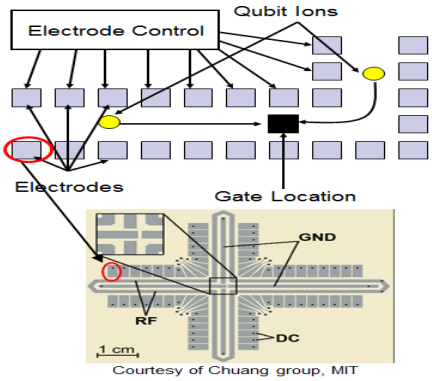


Fig. 1 Quantum Computing with Ion Traps [23]

There are two strictly quantum operation types that a low-level circuit is composed of: single- or multi-qubit logical gates and a single qubit measurement. All other operations are classical control operations, which includes communication instructions [1].

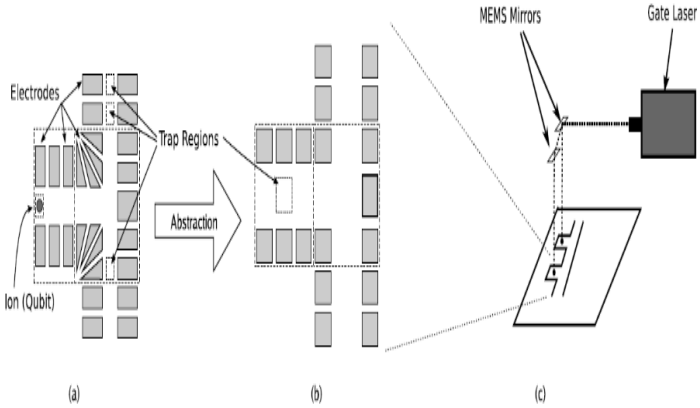


Fig. 2 (a) Physical layout demonstrated for a T-junction (three-way intersection). (b) Abstraction of the circuit in (a), built using the straight channel and three way intersection macro blocks shown in Fig. 3 (c) MEMS mirrors placed above the ion traps plane guide the laser beams to gate locations [10].

Fig. 2 shows a possible mapping of a demonstrated layout (Fig. 2a) to macroblock abstractions (Fig. 2b). As Fig. 2c shows, the laser pulses are guided to the gate locations by an array of MEMS mirrors located above the ion trap place to apply quantum gates [24]. Fig. 3 show the library defined in [10]. As this Fig. shows, each macroblock consists of a 3×3 structure of trap regions and electrodes with some ports to allow qubit movement between the macroblocks. The black squares are gate locations, which may not be performed at intersections or turns in the ion trap technology. Different orientations of each of these macroblocks can be used in a layout.

Some key characteristics of ion trap technology can be summarized as follows:

• Rectangular channels lined with electrodes make "wires" in ion traps. Atomic ions (qubits) can be suspended above the channel regions and moved ballistically by application of voltages on the channel electrodes [25]. Therefore, a movement control circuitry is required for each wire to handle any qubit communication.

• Any operation available in the ion trap technology can be performed at each gate location. This makes it possible to reuse gate locations for different operations within a quantum circuit.

• Fabrication and control of ion traps in the third dimension is difficult. Thus, scalable ion trap systems are two dimensional [17]. Therefore, routing channels should have T-junction(s) or cross-junction(s) to allow ions to move from one channel to another.

• Aside from Manhattan distance between the source and target locations for an ion movement, the geometry of the wire channel is also important in the calculation of movement latency. Experiments have shown that a right angle turn takes substantially longer than a straight channel over the same distance [25].

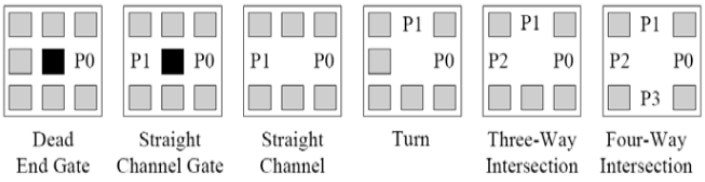


Fig. 3 Library of basic macroblocks used in this paper. Ports (P0–P3) and electrodes of each marcoblock make it possible for the ion to be moved and trapped. Some macroblocks contain a trap region where gates may be performed (black square) [10].

C. Simulated annealing

Simulated annealing is a method of local search algorithm. The process of annealing can be simulated with the Metropolis algorithm, which is based on Monte Carlo techniques. This technique can be used to generate a solution to combinatorial optimization problems assuming an analogy between them and physical many-particle systems with two equivalences: solutions in the problem are equivalent to states in a physical system, and the cost of a solution is equivalent to the "energy" of a state [26].

The search is started with a randomized state. In a polling loop we will move to neighboring states always accepting the moves that decrease the energy while only accepting bad moves accordingly to an acceptance probability can be given by $e^{\frac{-\Delta C}{T}}$ dependent on the "temperature" of the system (T).

Decrease the temperature slowly, accepting less bad moves at each temperature level until at very low temperatures. In this way, the algorithm converges to an optimal or near optimal configuration. The outline of the simulated annealing algorithm is shown in Fig. 4. The parameters and functions used in a simulated annealing algorithm determine the quality of the placement produced. These parameters and functions include the cooling schedule consisting of initial temperature (init-temp), final temperature (final-temp) and the function used for changing the temperature (SCHEDULE), inner- loop-criterion which is the number of trials at each temperature, the process used for shuffling a configuration (PERTURB), acceptance probability (F), and the cost function (COST). A good choice of these parameters and functions can result in a good placement in a relatively short time [27].

```
Algorithm SIMULATED-ANNEALING
Begin
    Temp = INIT-TEMP;
    Place = INIT-PLACEMENT;
    While (temp > FINAL-TEMP) do
        While (inner-loop-criterion = FALSE) do
            New-place = PERTURB (place);
∆C = COST (new-place) – COST (place);
            If (∆C < 0) then
                Place = new-place;
            Else if (RANDOM (0, 1) < e^(−∆C/temp)) then
                Place = new-place;
        Temp = SCHEDULE (temp);
End.
```

Fig. 4 The Simulated annealing algorithm[27]

D.Maze Routing Algorithm

Maze routing algorithms are used to find a path between a pair of points, called the source and the target respectively, in a planar rectangular grid graph. The objective of a maze routing algorithm is to find a path between the source and the target vertex without using any blocked vertex. The process of finding a path begins with the exploration phase, in which several paths start at the source, and are expanded until one of them reaches the target. Once the target is reached, the vertices need to be retraced to the source to identify the path[28].

III. Related Work

We are not the first to build a computer-aided design flow for quantum circuits. Svore et al. [2],[3] proposed a design flow that starts with a quantum program and generates its corresponding physical operations. Their work outlined various file formats and provided initial implementations of some of the necessary tools. Their design flow which has four phases, converts a high-level program specified in the mathematical abstractions of quantum mechanics and linear algebra into a low-level set of machine instructions scheduled on a fixed H-tree-based layout [2]. Balensiefer et al. [4],[5] proposed a design flow which takes a quantum description in QCL (Quantum Computation Language) [29] and generates a technology-dependent netlist. QCL that is defined by Omer [29] utilizes a syntax derived from C and provides a quantum simulator for code development and testing on a classical computing platform. In the physical design phase, the generated netlist is scheduled on a fixed layout by a list-scheduling algorithm [20]. No optimization is done on the laid out circuit. Metodi et al. [6] proposed a uniform Quantum Logic Array architecture, and extended and improved it later in [7]. The focus of the work was on the architectural research and the details of physical layout or scheduling were not explored. Whitney et al. [10] evaluate circuits by using the tools which allow us to lay out circuits. They proposed architectures for quantum computers have all consisted of computation regions connected by an interconnection network using quantum teleportation [18]. They also suggested [10] a quantum design flow that takes a description and generates its layout in ion trap technology. They proposed new heuristics for the layout generation and scheduling. Their physical design stage includes laying out and scheduling a fixed netlist.

Dousti et al. [30] focus on minimizing the total latency of the circuit to minimize the error in the circuit. A CAD tool, called Quantum mapper based on Scheduling, Placement, and Routing or QSPR, was developed to perform this task automatically. More precisely, the destination qubit is fixed in one trap while the source qubit is moved to reach the destination. Quantum physical operations scheduler (QPOS) distinguishes between the source and destination operands of a two-qubit instruction during the routing step [12]. QPOS extracts a routing path for each of the ready-to-issue instructions. If there are any overlaps among these paths, QPOS selects an instruction to execute based on the following criteria: 1) highest initial priority, 2) lowest among of congestion that is going to be introduced by using the path, and 3) shortest path length. Finally, QPOS maps these paths to the quantum circuit fabric and uses a deadlock prevention algorithm to prohibit qubits to locate in a position that further movement is impossible.

In earlier works, Mohammadzadeh et al. [31] introduced the physical synthesis concept for a quantum design flow and proposed a technique for it that exchanges the gates in the design after layout generation to improve the latency of quantum circuit execution. They also proposed[32] a new technique for physical synthesis using auxiliary qubit selection to improve the latency of quantum circuits. Recently, this group proposed an optimization technique using gate location changing to improve the latency of quantum circuits [33].

The main difference between our methodology and most another related works is using variable layout in quantum design. In most before works, a quantum design flow performs the synthesis and physical design processes separately. They take a fixed netlist and lay it out on a fix layout. But we consider a new placement that creates an optimum layout with suitable area in acceptable perform time for larger circuits

with more qubits. The goal of the placement is to minimize the total area.

IV. Our Placement Strategy

A vastly simplified version of typical classical circuit CAD flow is shown in Fig. 5. A user-specified application circuit specification into some sort of gate network is first synthesized then physical components are geometrically mapped to a substrate to make physical design. Verification steps ensure equivalence between stages. The purpose of the flow is to take in some sort of abstract circuit specification and produce a physical design that can be fabricated then [23].

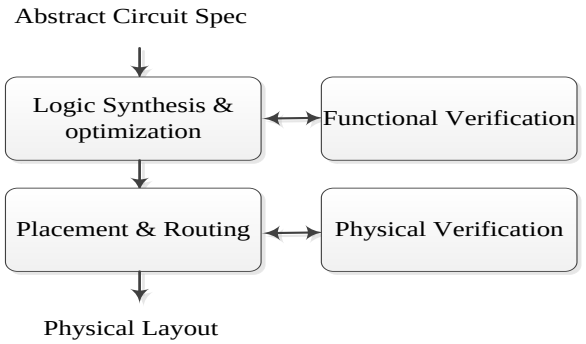


Fig. 5 Simplified view of a classical computer-aided design flow

We proposed our placement flow that presented in Fig. 6. This flow has two main steps. During the first stage, an initial placement is generated based on a modified version of dataflow graph based algorithm [10]. The second stage optimizes the initial layout to reach the final improved layout.

In the first stage, the algorithm provides an initial layout for the second stage. At the top, it begins with a netlist of the desired quantum circuit. Then, a dataflow graph is created that is equivalent to the instruction sequence. Each node represents an instruction, as labeled and each arc represents a qubit dependency. To increase the parallelism, the algorithm prioritized gate location. It means the instructions that have not any common qubits can have the same priority. Then, it prepares an initial placement of qubits and gate locations. Layout is constructed by going through dataflow graph. After that, long columns are folded into short columns to get a more compact layout, where every column is nearly the same height. Also, two vertical global routing channels are inserted between columns. The columns are then sorted to position gate locations that need to be connected roughly horizontal to one another. Thereafter, the algorithm inserts channel rout based on data flow graph between gate locations. It eliminates a weight to each edge of data flow. It is based on latency of qubit movement on that edge. The algorithm selects the longest edge on the longest critical path and merges these two node groups to eliminate this latency, in effect specifying that these two instructions should occur at the same gate location. Finally, it reroutes the paths and channels and the schedule again. Until create a triple gate in a location or the scheduler give us a time more than the previous layout, it does this work[10].

The generated layout is initial layout for our proposed methodology and is able to be fed into the SA algorithm stage. The SA algorithm includes four stages: 1) creating a new solution to find a close feasible solution (Perturb), 2) cost calculation for new solution, 3) analysis acceptance for new solution, 4) repeats management and periods changing (cooling rate).

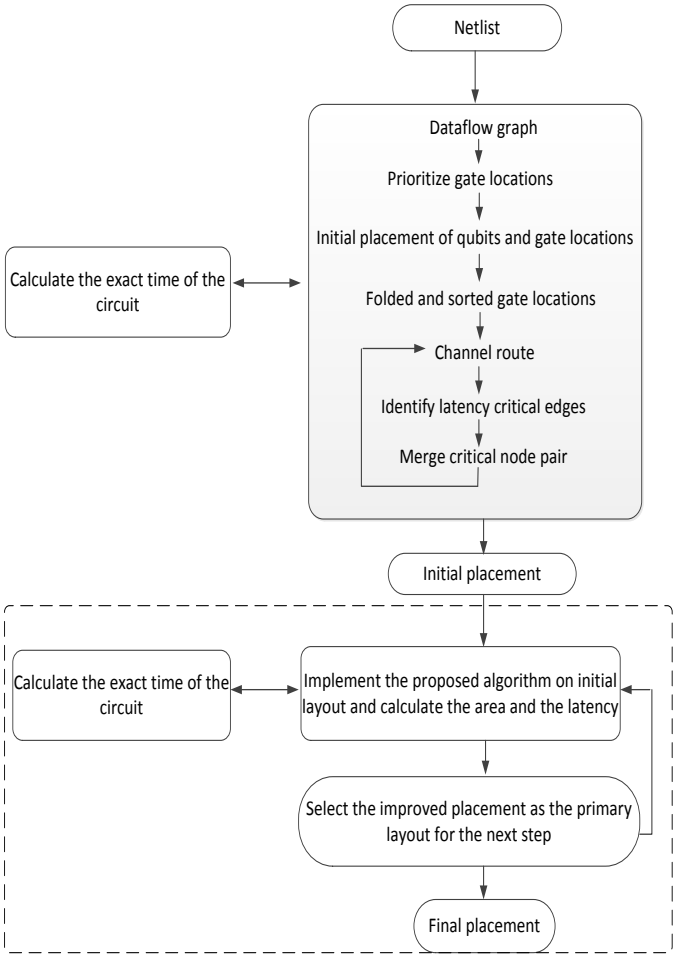
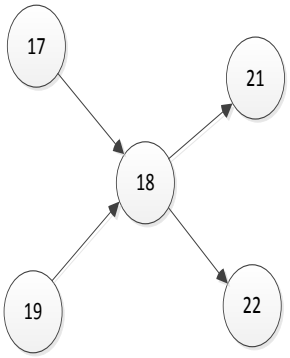


Fig. 6 The general idea

A. Perturb Procedure

For perturb, we propose merge gate locations method which is described below. Merge method is the best way to shrink circuit, but during it the latency of implementations should be considered. During each perturb call, merge process is done to determine the pair of gate locations for eliminating row or column that source gate locations are in it. Merge is based on gates dependency and consist of three parts: merge maximum three gates in a location, calculation the latency distance and arrival time of qubits of merging gates to location. If in merging process, we face with the gate location that include three gates in a row or column, that row or column will be set aside for elimination. Calculating the distance is done with the assumption that the channel is empty. For example, according to Fig. 7, if gate number 18 is source gate and gates number 22, 19, 17 and 21 are input or output gates for it, the gate number 18 can be merged with one of them. According to

Table 1, we determine the distance of gate number 18 to its depended gates and measure total distances. The lowest total is the highest priority for merging (Sum1).



			21			
19			17			22
			18			

Fig. 7 An example for calculating the latency distance

TABLE I
CALCULATING THE DISTANCE OF GATE NUMBER18 TO ITS DEPENDED GATES

18→17	18→19	18→22	18→21
18 to 19: 3 18 to 22: 3 18 to 21: 22	18 to 17: 3 18 to 22: 6 18 to 21: 23	18 to 17: 3 18 to 19: 6 18 to 21: 23	18 to 17: 22 18 to 19: 23 18 to 22: 23
Sum1 : 28	Sum2 : 32	Sum3 : 32	Sum4 : 68

Qubit of gate with larger number in merging gates may arrive earlier than expected and won't be allowed to enter in location. It can prevent entering required qubit for the first gate; as a result, congestion and latency increase in that place. To solve this problem, we computed the arrival time of each qubit to merger location. If the intruder qubit hadn't arrived to merger location merging is done, otherwise the algorithm goes back to pre-stage and selects the next priority.

B. *Cost Function*

The cost function is defined as follows:

$$\Delta C = (1-\lambda)\left[\frac{\Delta TCOST}{OLD\ TCOST}\right] + \lambda\left[\frac{\Delta ACOST}{OLD\ ACOST}\right] \qquad (1)$$

The cost function is weighted linear sum of area and time. λ is a trade-off variable to determine how much weight to give each component. Because of automatically adjusting the weights of the two components, algorithm is always allocating λ coefficient to changes in the area cost, and 1-λ coefficient to changes in the time cost. If λ is 1 then we have an algorithm that focuses only on area, but it ignores time improvement. If λ is 0, m focus is only on improving execution time.

C. *Acceptance Analysis*

After calculating ΔC, if the result is a negative number, we accept the new placement and replaces with the current layout. If ΔC is positive and Random (0,1) $< e^{\frac{-\Delta C}{T}}$, then new layout is accepted and replaced. Otherwise, new placement doesn't accept.

D.Cooling Rate

Final step in SA is to manage repetition and change periods flow. SA runs appropriate number of performed iterations during each period. These iterations are related to number of rows and columns in extant layout. Cooling rate consists of initial temperature, temperature steps and number of iteration at each temperature. The strategy of temperature reduction is calculated below:

$$T_{new} = T_{old} - 1 \qquad (2)$$

V. EXPERIMENTAL RESULTS

A. Experimental Setup

The proposed placement strategy is implemented in C#. The parameters used for the simulation are as follow: INIT-TEMP= initial layout size; INIT-PLACEMENT= upshot layout based on assumptions in [10]; FINAL-TEMP= 0; inner-loop= number of rows and columns in current layout; PERTURB (place) = described merge procedure; SCHEDULE (T) = T – 1. An Apple MacBook Pro laptop with Intel core 2 Duo CPU and 4 GB RAM was used to run the simulation.

B. Simulation Results

Table II and Fig. 8 and 9 show results of simulation for our purposed optimization algorithm with simulated annealing.

TABLE II
THE NON-MODIFIED PLACEMENT RESULTS

Circuit Name [34]	Area (Microblocks) [23]	Area after SA Optimization	Area Improvement (%)	Latency (μs)[23]	Latency after SA Optimization (μs)	Latency Improvement (%)
[[5.1.3]]	70	35	50	216	218	-0.93
[[7.1.3]]	70	35	50	206	192	6.8
[[9.1.3]]	143	91	36.36	298	285	4.36
[[11.1.3]]	208	117	43.75	515	481	6.6
[[12.1.4]]	285	209	26.67	570	489	14.21
[[19.1.7]]	651	420	35.48	1170	1044	10.77
[[20.1.6]]	782	646	17.39	1132	1120	1.06
[[28.2.8]]	1426	1150	19.35	1754	1697	3.25
[[29.1.11]]	1617	1334	17.5	1735	1800	-3.75
[[32.2.8]]	1518	1242	18.18	1750	1724	1.49
[[35.1.10]]	2262	1595	29.49	4612	4082	11.49
[[40.3.10]]	2752	1716	37.65	6039	5314	12.01
Average			31.81			5.61

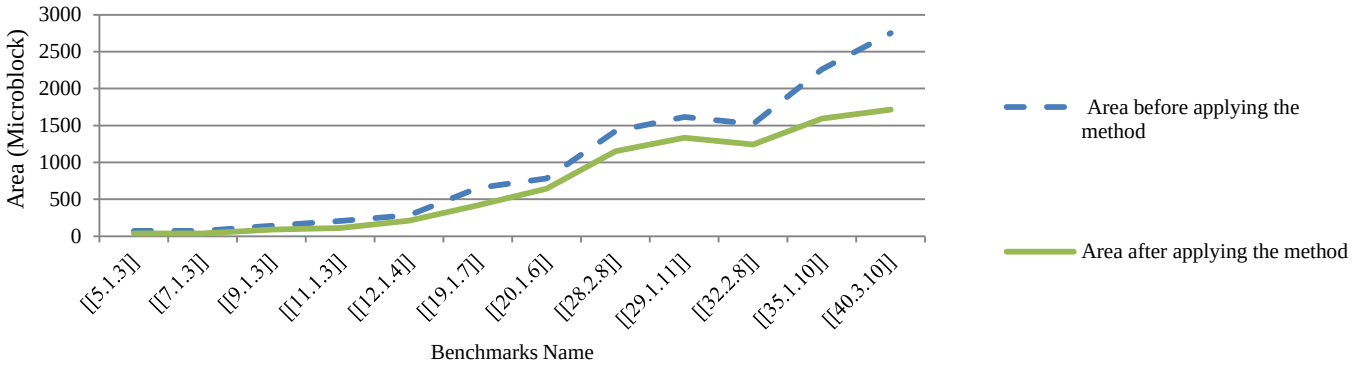


Fig. 8 Area Improvement

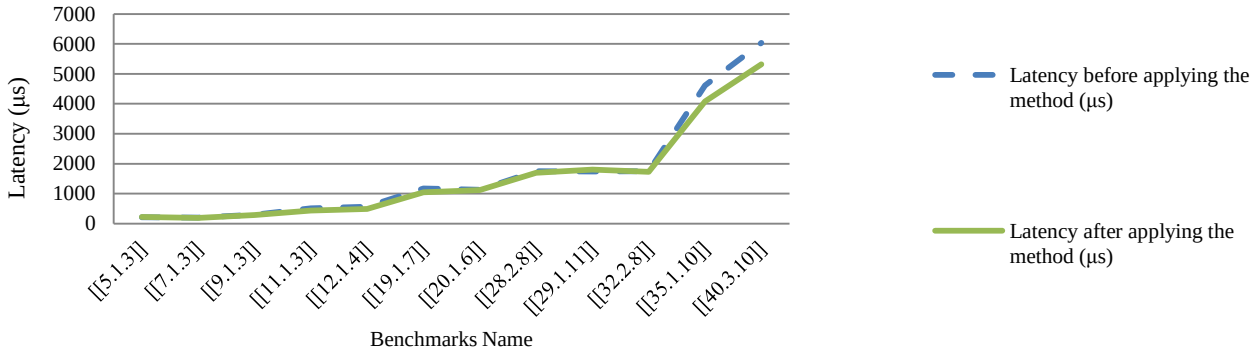


Fig. 9 Latency Penalty

VI. Conclusion

This paper includes a new technique for placement in quantum circuits. This style of placement is due to merge of gates location and repeat. In this way, we confront some problems such as, manner of leaving qubits from gate location, movement qubits in channel, blockage and congestion. We provided a solution for each of these problems.

When the goal is circuit run time, a skillful quantum physical designer can apply the special techniques to control entering and leaving qubits, prevent the achievement qubits that generate traffic in channel and calculate distances for shrinking the circuit to the best way and the least delay. In addition to mentioned problems, there are some physical limitations such as being impossible to merge more than three gates in a gate location or increase latency penalty in complex circuits toward reduction area. In general, simulated annealing placement outperforms the other placements regarding final results.

References

[1] T.S. Metodi, D. Thaker, A. Cross, F. Chong, and I. Chuang, "Scheduling physical operations in a quantum information processor," Proceedings of SPIE Defense and Security Symposium, Vol. 6244, 2006, pp. 62440T.1-62440T.12.

[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 (QPL), 2004, pp. 145–162.

[3] K. Svore, A. Aho, A. Cross, I. Chuang, and I. Markov, "A Layered Software Architecture for Quantum Computing Design Tools," Computer, Vol. 39, No. 1, 2006, pp. 74–83.

[4] S. Balensiefer, L. Kregor-Stickles, and M. Oskin, "An evaluation framework and instruction set architecture for ion-trap based quantum micro-architectures," Proceedings of International Symposium on Computer Architecture (ISCA), 2005, pp. 186 – 196.

[5] S. Balensiefer, L. Kreger-Stickles, and M. Oskin, "QUALE: quantum architecture layout evaluator," Proceedings of SPIE the international society for optical engineering, Vol. 5815, 2005, pp. 103-114.

[6] 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 (MICRO), 2005, pp. 305-318.

[7] D.Thaker, T. Metodi, A. Cross, I. Chuang, and F. Chong, "Quantum Memory Hierarchies: Efficient Designs to Match Available Parallelism in Quantum Computing," Proceedings of the 33rd International Symposium on Computer Architecture (ISCA), 2006, pp. 378-390.

[8] T. Metodi et al, "High-level interconnect model for the quantum logic array architecture," ACM Journal on Emerging Technologies in Computing Systems, Vol. 4 , No. 1, March 2008.

[9] T. S. Metodi and F. T. Chong, "Quantum Computing for Computer Architects," Morgan &Claypool Publishers, 2006.

[10] M. Whitney et al., "Automated Generation of Layout and Control for Quantum Circuits," Proceedings of Computing Frontiers, 2007, pp. 83 – 94.

[11] Naser Mohammadzadeh, "Design Process in Quantum Computing," PhD thesis, Amirkabir University of Technology, 2009.

[12] P. Aliferis, D. Gottesman, and J. Preskill, "Quantum accuracy threshold for concatenated distance-3 codes," Arxiv preprint quant-ph/0504218, 2005.

[13] J. I. Cirac and P. Zoller,"Quantum computations with cold trapped ions," Phys. Rev. Lett. 74, 1995, pp. 4091–4094.
[14] A. Cross. Qasm-tools. http://www.media.mit.edu/quanta/quanta-web/projects/qasm-tools/, 2006.
[15] L. Grover, "A fast quantum mechanical algorithm for database search," Proceeding of ACM Symposium on Theory of Computing, 1996, pp. 212–219.
[16] W. Hensinger, S. Olmschenk, D. Stick, D. Hucul, M. Yeo, M. Acton, L. Deslauriers, C. Monroe, and J. Rabchuk,"T-junction ion trap array for two-dimensional ion shuttling, storage, and manipulation," Applied Physics Lett. 88(3), 2006, pp. 34101.
[17] D. Hucul, M. Yeo, W. K. Hensinger, J. Rabchuk,S. Olmschenk, and C. Monroe, "On the transport of atomic ions in linear and multidimensional ion trap arrays," Quant-ph/0702175, 2008.
[18] N. Isailovic, Y. Patel, M. Whitney, and J. Kubiatowicz, "Interconnection Networks for Scalable Quantum Computers," Proceedings of the 33rd International Symposium on Computer Architecture (ISCA), 2006.
[19] M. A. Nielsen and I. L. Chuang, "Quantum computation and quantum information," Cambridge University Press, 10th publish, 2010.
[20] T. Yang and A. Gerasoulis, "List scheduling with and without communication delays," J. Parallel Comput. 19(12), 1993, pp. 1321–1344.
[21] 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 (MICRO), 2005, pp. 305–318.
[22] D. Kielpinski, C. Monroe, and D. Wineland, "Architecture for a Large-Scale Ion-Trap Quantum Computer," Nature 417, 2002, pp. 709–711.
[23] Mark Gregory Whitney, "Practical Fault Tolerance for Quantum Circuits," University of California, Berkeley, spring 2009.
[24] J. Kim, S. Pau, Z. Ma, H. McLellan, J. Gages, A. Kornblit, and R. Slusher,"System design for large-scale ion trap quantum information processor," Journal of Quantum Information and Computation 5, No. 7, 2005, pp. 515–537.
[25] J. Chiaverini, R. Blakestad, J. Britton, J. Jost, C. Langer, D. Leibfried, R. Ozeri, and D. Wineland, "Surface-Electrode Architecture for Ion-Trap Quantum Information Processing," Journal of Quantum Information and Computation, Vol. 5, No. 5, 2005, pp. 419-439.
[26] http://www.cs.washington.edu/education/courses/.../SimulatedAnnealing.ppt, 10/7/2005.
[27] N. Sherwani, "Algorithms for VLSI Physical Design Automation", 3rd edition, Kluwer Academic Publishers, Boston, MA, 1999.
[28] http://www.dte.uvigo.es/articulos/al4pda/mazerouting.html.
[29] B. Omer, "Quantum Programming in QCL," Master thesis, Technical University of Vienna, 2000.
[30] M. J. Dousti and M. Pedram, "Minimizing the latency of quantum circuits during mapping to the ion-trap circuit fabric," Proc. of Design Automation and Test in Europe, Mar. 2012.
[31] N. Mohammadzadeh, M. Sedighi, and M. Saheb Zamani, "Quantum Physical Synthesis: Improving Physical Design by Netlist Modifications," Elsevier Microelectronics Journal, Vol. 41, 2010, pp. 219-230.
[32] N. Mohammadzadeh, M. Saheb Zamani, and M. Sedighi, "Auxiliary Qubit Selection: A Physical Synthesis Technique for Quantum Circuits," To be published in Springer Quantum Information Processing Journal, 2011.
[33] N. Mohammadzadeh, M. Sedighi, M. Saheb Zamani, "Gate Location Changing: An Optimization Technique for Quantum Circuits," World Scientific International Journal of Quantum Information (IJQI), Vol. 10, No, 3, 2012.
[34] http://iaks-www.ira.uka.de/home/grassl/QECC/circuits/index.html, Copyright © Markus Grassl