Design of Efficient Quantum Circuits Using Nearest Neighbor Constraint in 2D Architecture

Leniency Marbaniang[1], Abhoy Kole[2], Kamalika Datta[1], and Indranil Sengupta[3(✉)]

[1] National Institute of Technology Meghalaya, Shillong, India
leniencym06@gmail.com, kdatta@nitm.ac.in
[2] B.P. Poddar Institute of Management and Technology, Kolkata, India
abhoy.kole@gmail.com
[3] Indian Institute of Technology Kharagpur, Kharagpur, India
isg@iitkgp.ac.in

Abstract. With the development in quantum computing, nearest neighbor constraint has become important for circuit realization. Various works have tried to make a circuit nearest neighbor compliant (NNC) by using minimum number of SWAP gates. To this end, an efficient qubit placement strategy is proposed that considers interaction among qubits and their positions of occurrence. Experimental results show that the proposed method reduces the number of SWAP gates by 3.3% to 36.1% on the average as compared to recently published works.

Keywords: Nearest neighbor · Qubit · 2D architecture · Quantum gate

1 Introduction

Quantum computing has drawn the attention of researchers over several decades. Unlike conventional binary logic systems that manipulate bits, quantum systems manipulate qubits that can exist as a state of superposition: $\phi = \alpha|0\rangle + \beta|1\rangle$, where $|\alpha|^2+|\beta|^2 = 1$. Qubits can be implemented using technologies like ion-trap [1], photonics [4], nuclear magnetic resonance [3], etc. In some technology like ion-trap, the operation requires that the interacting qubits must be adjacent to each other known as the *Nearest Neighbor Constraint*. This is achieved by inserting an appropriate number of SWAP gates for nearest neighbor compliance. Several works have been proposed for arranging qubits in 1D [2,5,6] and 2D [7–9] architectures where the main aim is to minimize the number of SWAP gates. 2D architectures require fewer number of SWAP gates for NNC. In this paper, a heuristic procedure for mapping qubits to a 2D grid is proposed, which considers gate position, degree of lookahead and strength of interaction among qubits.

The paper is organized as follows. Section 2 explains the proposed method and steps of algorithm using examples. In Sect. 3, experimental results and comparison with previous works have been presented followed by concluding remarks in Sect. 4.

I. Phillips and H. Rahaman (Eds.): RC 2017, LNCS 10301, pp. 248–253, 2017.
DOI: 10.1007/978-3-319-59936-6_19

2 Proposed Method

This section presents a qubit placement and SWAP gate insertion approach to make a quantum circuit NNC. This is based on a lookahead strategy that considers the frequency of occurrence of gates and their relative positions. Given the lookahead value LA, a window of LA gates $C = g_i g_{i+1} g_{i+2} \cdots g_{(i+LA-1)}$ is analyzed to determine the most interactive and frequently occurring qubits in the block. A data structure as shown in Fig. 1 is constructed for each qubit consisting of their interacting qubits, number of interactions and gate numbers. Using this structure, an interaction table is created as shown in Tables 1(a) and (b), from which the priority of the qubits is determined. Having the qubit priority list as in Table 1(g), qubit placement in the 2D grid as explained by Algorithm 1 is carried out such that the highest priority qubit is placed at the center and its interacting qubits are placed around it in the order <bottom, right, top, left>. Using the 2D grid, appropriate number of SWAP gates are inserted before the gate to bring the interacting qubits adjacent to each other and the new position of the qubit is retained. Finally, the total number of SWAP gates is counted and recorded. The same process is repeated for the next block of LA gates and also for the pair of blocks combined. This method is applied to other blocks of the same circuit and for different values of LA, and the configuration with minimum SWAP gate count is chosen as the best.

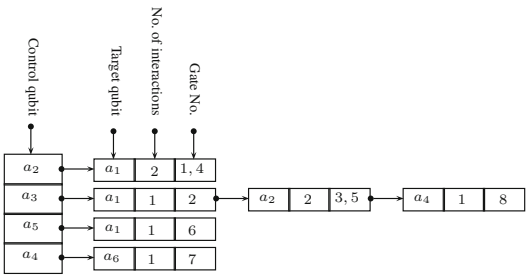

Fig. 1. Data structure of 1st block

2.1 The Proposed Algorithm

Knowing the LA value, different blocks of the circuit are defined by scanning the circuit from left to right. For each block, data structure and priority table are constructed and placement of the qubits in the 2D grid is performed (see Algorithm 1). Firstly, a qubit from the qubit priority table is selected and placed in the grid followed by its interaction qubits as shown in Fig. 2. If the qubit is already present, nothing is done. Initially it checks if the cell is empty; if not, it checks the next cell for space availability. This process is repeated until it finds an empty cell and inserts the qubit. Next SWAP gates are inserted as needed. Lastly the number of SWAP gates is calculated and recorded.

Algorithm 1. Qubit Placement

Input: Qubit Priority Table PT, Interaction Table IT
Output: Qubit placement GD in 2D grid
begin
  for $q_i \in PT$ **do**
    if $(q_i \notin GD)$ **then**
      $x = mid\_x(GD)$;
      $y = mid\_y(GD)$;
      if ($GD_{x,y}$ is NOT empty) **then**
        Find (x, y) such that $GD_{x,y}$ is empty and adjacent to maximum number of empty cells;
      enddif
      Place q_i at $GD_{x,y}$;
    else
      Retrieve location (x, y) of $q_i \in GD$;
    endif
    for $((q_j \in IT)$ **and** $(IT_{q_i,q_j} \geqslant 1))$ **do**
      Place q_j in one of the empty cell from $GD_{x\pm r,y\pm c}$ where $r, c = 1, 2, \ldots$;
    endfor
  endfor
  return GD;
end

Table 1. Illustration for the first block. (a) Random Interaction Table, (b) Interaction Table (after sorting), (c) Qubit Table, (d) Qubit Table (after sorting based on maximum interactions), (e) Qubit Table (after sorting the gate numbers), (f) Qubit Table with time interval, (g) Qubit Priority Table

Control	Target	Interactions	Gate no
a2	a1	2	1,4
a3	a1	1	2
a3	a2	2	3,5
a3	a4	1	8
a5	a1	1	6
a4	a6	1	7

(a)

Control	Target	Interactions	Gate no
a2	a1	2	1,4
a3	a2	2	3,5
a3	a1	1	1
a3	a4	1	8
a5	a1	1	6
a4	a6	1	7

(b)

Qubit	Interactions	Gate no
a2	4	1,4,3,5
a3	4	3,5,2,8
a5	1	6
a4	2	8,7
a1	4	1,4,2,6
a6	1	7

(c)

Qubit	Interactions	Gate no
a2	4	1,4,3,5
a3	4	3,5,2,8
a1	4	1,4,2,6
a4	2	8,7
a5	1	6
a6	1	7

(d)

Qubit	Interactions	Gate no
a2	4	1,3,4,5
a3	4	2,3,5,8
a1	4	1,2,4,6
a4	2	7,8
a5	1	6
a6	1	7

(e)

Qubit	Interactions	Gate no	Time Interval
a2	4	1,3,4,5	4
a3	4	2,3,5,8	6
a1	4	1,2,4,6	5
a4	2	7,8	1
a5	1	6	0
a6	1	7	0

(f)

Qubit	Interactions	Gate no	Time Interval
a2	4	1,3,4,5	4
a1	4	1,2,4,6	5
a3	4	2,3,5,8	6
a4	2	7,8	1
a5	1	6	0
a6	1	7	0

(g)

2.2 Illustrative Example

Consider the benchmark circuit *4gt4-v0_80* that consists of 6 qubits and 44 gates. We illustrate the steps of qubit mapping for $LA = 8$. In the first invocation of the lookahead mechanism the block will consist of the first 8 gates. In the second call it will consist of the next 8 gates, and in the last call it will consist of all the 16 gates. In the first invocation a data structure as shown in Fig. 1 is constructed to find out the interacting qubits, the number of interactions and gate numbers

where they interact within this block. Then a random interaction table is created by filling it randomly as shown in Table 1(a).

This random priority table is then sorted based on the interactions to get Table 1(b). If there is more than one gate with the same interacting qubits then we keep a record of just one gate, sum up the interactions, append the gate numbers and sort it again. Using the modified priority table, for each qubit, we calculate the total interactions and record all the gate numbers as shown in Table 1(c) followed by sorting as in Table 1(d). Next, the gate numbers of each qubit are sorted in ascending order as in Table 1(e). From this table, the qubits, their total interactions, the gate numbers, and the interval between the gates of each qubit is calculated as shown in Table 1(f). It is seen that qubits $a2$, $a3$ and $a5$ have four interactions but their frequencies of interaction are different. So qubit with the least time interval gets the highest priority, viz. $a2$, as seen in Table 1(g). Lastly, the circuit is scanned again to check if any qubit not in the block is left unfilled in the qubit priority table. If so, the qubit is appended in the table. Using this priority table, qubit placement is done as Algorithm 1 and illustrated in Fig. 2.

After qubit placement is completed in a 2D grid, SWAP gates insertion is performed. The process is illustrated for a benchmark *4gt11_84* that have five qubits, one of which (viz. a_4) is not involved in any gate interactions. The steps are shown in Fig. 3 which requires 2 SWAP operations.

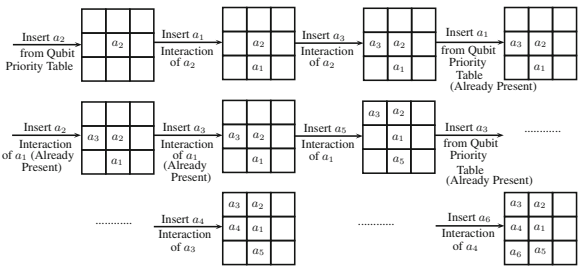

Fig. 2. Qubit placement of 1st block

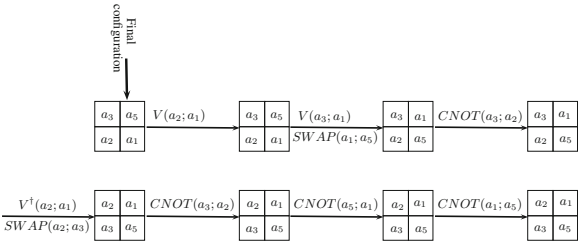

Fig. 3. Swap gate insertion

3 Experimental Results

The proposed method has been implemented in C and run on a core-i5 based desktop with 4 GB of RAM. Experiments have been carried out on NCV benchmarks that was used in [7–9] and results are shown in Table 2 along with previous results. The first two columns represent the benchmark name and number of qubits (n). SWAP gates count ($swap$) observed in [8] is presented next followed by number of SWAP gates for joint lookahead ($swap_{\triangle}$) and iterative lookahead ($swap_{\triangledown}$) reported in [9] in the next two columns respectively. After this, SWAP

Table 2. Improvements in SWAP gates of 2D qubit placement over [7–9]

Benchmark		[8]	[9]		[7]	Proposed 2D			Impr. (%)			
Name	n	$swap$	$swap_{\triangle}$	$swap_{\triangledown}$	$swap$	$swap$	l	grid	[8]	$[9]_{\triangle}$	$[9]_{\triangledown}$	[7]
QFT5	5	5	–	–	5	4	2	3 × 2	20.0	–	–	20.0
QFT7	7	14	13	22	18	14	7	3 × 3	0	−7.7	36.4	22.2
QFT8	8	23	17	25	18	20	4	3 × 3	13.0	−17.6	20	−11.1
QFT9	9	36	22	27	34	32	11	4 × 3	11.1	−45.5	−18.5	5.9
QFT10	10	51	37	43	53	43	2	4 × 3	15.7	−16.2	0	18.9
Shor3	10	1770	1010	1485	1710	828	16	3 × 4	53.2	18	44.2	51.6
Shor4	12	–	2757	3807	4264	2118	20	3 × 4	–	23.2	44.4	50.3
Shor5	14	–	6344	8504	8456	5566	15	4 × 4	–	12.3	34.5	34.2
Shor6	16	19980	12468	15970	20386	12905	25	4 × 5	35.4	−3.5	19.2	36.7
3_17_13	3	3	5	8	6	5	2	2 × 2	−66.7	0	37.5	16.7
4gt4-v0_80	6	15	–	–	17	10	8	2 × 3	33.3	–	–	41.2
4gt10-v1_81	5	15	15	22	16	14	4	2 × 3	6.7	6.7	36.4	12.5
4gt12-v1_89	6	18	–	–	19	14	4	3 × 4	22.2	–	–	26.3
4mod5-v1_23	5	7	–	–	11	6	6	3 × 3	14.3	–	–	45.5
aj-e11_165	5	22	16	37	24	11	3	3 × 4	50.0	31.3	70.3	54.2
alu-v4_36	5	11	–	–	10	8	5	3 × 3	27.3	–	–	20.0
cycle10_2_110	12	588	483	824	839	635	11	4 × 3	−8.0	−31.5	22.9	24.3
ham7_104	7	45	37	53	48	29	9	3 × 4	35.6	21.6	45.3	39.6
ham15_108	15	280	233	355	328	199	8	4 × 4	28.9	14.6	43.9	39.3
hwb4_52	4	9	–	–	9	5	6	3 × 2	44.4	–	–	44.4
hwb5_55	5	49	37	64	45	35	3	3 × 3	28.6	5.4	45.3	22.2
hwb6_58	6	76	59	85	79	52	5	3 × 3	31.6	11.9	38.8	34.2
hwb7_62	8	1500	1050	1703	1688	1093	6	4 × 4	27.1	−4.1	35.8	35.2
hwb8_118	9	7877	6316	11096	11027	5892	6	3 × 4	25.2	6.7	46.9	46.6
hwb9_123	10	11233	8522	14459	15022	8661	10	4 × 3	22.9	−1.6	40.1	42.3
mod5adder_128	6	36	33	45	41	30	6	3 × 3	16.7	9.1	33.3	26.8
mod8-10_177	6	43	–	–	45	36	4	3 × 4	16.3	–	–	20
plus63mod4096_163	13	13316	11764	22160	22118	15180	17	4 × 4	−14.0	−29.0	31.5	31.4
plus63mod8192_164	14	18987	15484	29939	29835	15931	20	4 × 4	16.1	−2.9	46.8	46.6
plus127mod8192_162	14	33299	27549	52333	53598	28520	30	5 × 4	14.4	−3.5	45.5	46.8
rd53_135	7	40	30	47	39	29	6	4 × 3	27.5	3.3	38.3	25.6
rd73_140	10	43	–	–	37	25	5	4 × 4	41.9	–	–	32.4
urf1_149	9	37722	29252	41058	38555	22358	10	3 × 4	40.7	23.6	45.5	42
urf2_152	8	16755	12872	18101	16822	9098	10	3 × 4	45.7	29.3	49.7	45.9
urf3_155	10	93558	69693	95485	94017	67034	30	3 × 4	28.4	3.8	29.8	28.7
urf5_158	9	34416	25887	36813	34406	19050	15	4 × 3	44.6	26.4	48.3	44.6
urf6_160	15	42910	31540	43100	43909	28147	15	5 × 4	34.4	10.8	34.7	35.9

gate count of [7] is presented. In the next three columns, SWAP gate count of proposed approach (*swap*), *LA* value and 2D configuration (*grid*) are reported. The last four columns show the % improvement of the proposed approach over [7–9]. On an average improvements of 22.4% (53.2% in the best case) over [8], 3.3% and 36.1% (31.3% and 70.3% in the best case) over joint and iterative lookahead strategy from [9], and 32.4% (54.2% in the best case) over [7] are observed.

4 Conclusion

In this work, a new lookahead approach for qubit placement in a 2D grid to minimize the number of SWAP gates for NN-compliance is proposed. Prioritization of the qubits has been worked out to determine which qubit should be placed earlier by considering the qubit's number of interactions and position in the circuit. The most frequent qubit with less interval gets a higher priority. The results obtained are found to be better than those reported in existing works.

Acknowledgement. This work was partially supported by Department of Science and Technology, Government of India under Grant No. YSS/2015/001461.

References

1. Blatt, R.: Quantum information processing with trapped ions. In: Quantum Information and Measurement, p. Th1.1 (2013)
2. Chakrabarti, A., Sur-Kolay, S., Chaudhury, A.: Linear nearest neighbor synthesis of reversible circuits by graph partitioning. arXiv preprint (2011). arXiv:1112.0564
3. Lu, D., Brodutch, A., Park, J., Katiyar, H., Jochym-O'Connor, T., Laflamme, R.: NMR quantum information processing. In: Takui, T., Berliner, L., Hanson, G. (eds.) Electron Spin Resonance (ESR) Based Quantum Computing. BMR, vol. 31, pp. 193–226. Springer, New York (2016). doi:10.1007/978-1-4939-3658-8_7
4. Nemoto, K.: Photonic architecture for scalable quantum information processing in diamond. Phys. Rev. X **4**(3), 031022 (2014)
5. Rahman, M.M., Dueck, G.W., Chattopadhyay, A., Wille, R.: Integrated synthesis of linear nearest neighbor ancilla-free MCT circuits. In: 46th International Symposium on Multiple-Valued Logic (ISMVL), pp. 144–149, May 2016
6. Shafaei, A., Saeedi, M., Pedram, M.: Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures. In: 50th ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6, May 2013
7. Shafaei, A., Saeedi, M., Pedram, M.: Qubit placement to minimize communication overhead in 2D quantum architectures. In: 2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 495–500, January 2014
8. Shrivastwa, R.R., Datta, K., Sengupta, I.: Fast qubit placement in 2D architecture using nearest neighbor realization. In: IEEE International Symposium on Nanoelectronic and Information Systems, pp. 95–100, December 2015
9. Wille, R., Keszocze, O., Walter, M., Rohrs, P., Chattopadhyay, A., Drechsler, R.: Look-ahead schemes for nearest neighbor optimization of 1D and 2D quantum circuits. In: 21st Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 292–297, January 2016