A Parallel Approach to Convert Quantum Circuits to an LNN Architecture

Edgar Meza, Joni Fernández, Bengamín Barán, Joaquín Lima

Universidad Nacional de Asunción
San Lorenzo, Paraguay
Email: edgar.meza.franco@gmail.com, jonifernandezc@gmail.com
Email: bbaran@pol.una.py, joaquin.lima@pol.una.py

***Abstract*—This paper describes four algorithms implemented to solve the problem of converting general quantum circuits to a *Linear Nearest Neighbor* (LNN) architecture. All the implemented algorithms are based on the HIRATA II algorithm and consider two improvements: (i) the use of parallel computing, and (ii) branch & bound technique. The proposed parallel algorithms are tested with the largest test circuit presented in the work of Hirata et al., this circuit correspond to Shor's factorization algorithm (named as Shor10 circuit). Experimental results show a speedup of an order of magnitude from hours to seconds, improving slightly the quality of the converted circuit, measured as the number of inserted swap gates.**

Keywords*— *LNN architecture; Quantum Circuits; Parallel Computing.

I. Introduction

The design of a general quantum circuit allows the interaction of non-adjacent qubits; however, the current technology may not allow the interaction between non-adjacent qubits [13]. Therefore, quantum circuits require an architecture that facilitates implementation. *Linear Nearest Neighbor* (LNN) architecture [16] facilitates the implementation of quantum circuits. The conversion of a general quantum circuit to an LNN architecture is a hard task for conventional heuristics.

Among several alternatives, HIRATA II algorithm [11] provides a general conversion scheme applicable to non-trivial quantum circuits. Therefore, this paper proposes several parallel versions of HIRATA II algorithm. Proposed parallel versions implement a branch and bound scheme [19] to improve algorithm performance.

The algorithms implemented in this work are tested with the largest circuit considered in the work of Hirata et al. [11] to prove the advantage of the proposed alternatives.

Parallel computing seems an interesting alternative for this work given the size of computation, and availability of multi-core processors in today servers. This way, the studied problem may be solved considerable faster with a cluster of computers or even a multi-core Central Processor Unit (CPU) as far as the problem can be efficiently partitioned in smaller subproblems.

This work is organized as follows: Section II presents the general conversion problem while Section III describes HIRATA II algorithm. Then, the implemented algorithms are presented in Section IV while experimental results are presented in Section V. Finally, conclusions are left for Section VI, where future works are also presented.

II. Conversion of general quantum circuits to an LNN architecture

There are already known methods to convert a quantum circuit to an LNN architecture. Fowler et al. [5] describe a construction scheme of quantum circuits in LNN architecture. On the other hand, Hirata et al. [11] present a general conversion scheme applicable to any quantum circuit that is considered in this work for parallelization.

The process of converting a quantum circuit to LNN architecture involves the insertion of SWAP gates to the original circuit to change the order of the qubits in such a way that needed gates only opperate on neighboring qubits.

The conversion of a general quantum circuit to the LNN architecture is defined by Hirata et al. [11] as:

- **Input:** a general quantum circuit, composed of N qubits and K gates.
- **Output:** an equivalent LNN quantum circuit.
- **Objective:** to minimize the total SWAP gates added.
- **Restriction:** the equivalent circuit output should have all qubits in the same original order.

Quantum gates are in LNN architecture if the qubits necessary to operate a gate are adjacent. A quantum circuit is in a LNN architecture when all its gates are LNN.

Circuit in Figure 1 represents a quantum circuit that is not in an LNN architecture. The circuits in Figures 2 and 3 are in LNN architecture and they are equivalent to the circuit in Figure 1. Moreover, the circuit in Figure 3 represents the best solution of the two alternatives because it adds fewer SWAP gates to the original circuit. In fact, the number of inserted SWAP gates needed to convert a general quantum circuit to an LNN architecture is here considered as the main quality indicator of the convertion process.

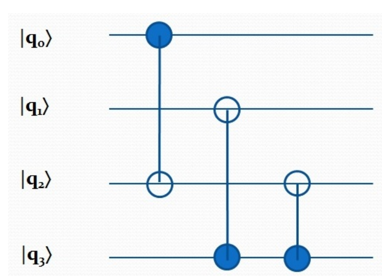

Figure 1. A quantum circuit not LNN

 ISBN: 978-1-61208-431-2

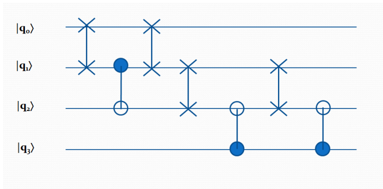

Figure 2. A quantum circuit equivalent to the one presented in 1 in an LNN architecture

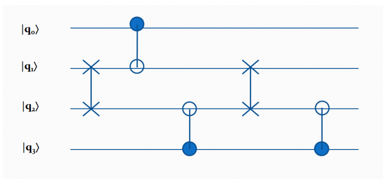

Figure 3. LNN quantum circuit equivalent to circuit of 1 with fewer SWAP gates

III. HIRATA II ALGORITHM

Hirata et al. [11] presented the HIRATA II algorithm to reduce the number of candidates to be evaluated in the convertion of a general quantum circuit to on LNN architecture with respect to classic heuristics as greedy algorithms.

As explained in detail in [11], HIRATA II algorithm defines an objective function f to be minimized wich can be evaluated for each candidate solution. The objective function to evaluate candidates is given by:

$$f(n_{i,j}) = f_1(n_{i,j}) + f_2(n_{i,j}) + f_3(n_{i,j}) \quad (1)$$

with:

$$f_1 = local\_search_w(n_{i,j}, w),$$

$$f_2 = calc\_swaps(c\_order, n_{i,j}) \quad and$$

$$f_3 = \frac{c_k}{k-i+1} calc\_swaps(n_{i,j}\, l\_order)$$

where $n_{i,j}$ represents the current candidate j for the current gate i; c_order is a list that represents the current order of the qubits while l_order is a list representing the initial order of the qubits. $local\_search_w(n_{ij})$ represents the lowest cost of converting the following w gates if j is chosen. $calc\_swaps(c\_order, n_{ij})$ is the number of SWAP gates necessary to obtain n_{ij} from c_order. $\frac{c_k}{k-i+1} calc\_swaps(n_{ij}\, l\_order)$ represents an estimation of the cost necessary to re-order the final order to the original order (see restriction in Section I). This term receives more preponderance in the conversion when the process progresses and it gets closer tho the end. The ck constant is chosen a priori.

IV. IMPLEMENTED ALGORITHMS

In this paper, three algorithms are proposed based on the original HIRATA II algorithm. The H2-S version is a sequential algorithm that implements a branch and bound technique

Algorithm 1 HIRATA II

```
Require: N,K,w
l_order ← {0, 1, 2, ..., N − 1}
c_order ← l_order
swaps ← 0
i ← 0
MIN ← ∞
while i < K do
   n ← makeCandidates(c_order, i)
   j ← 0
   while j ≤ |n| − 1 do
      AUX ← evaluate_objective_function()
      if AUX < MIN then
         S ← ∅ // List of candidates n_ij, the amount of
         swap gates was minimal.
         S ← S ∪ n_ij
         MIN ← AUX
      else if AUX = MIN then
         S ← S ∪ n_ij
      end if
      j + 1
   end while
   S_a ← random(S) // Candidate n_ij the set S chosen
   randomly.
   swaps ← swaps + calc_swap(c_order, S_a)
   n ← ∅
   S ← ∅
   c_order ← S_a
   i ← i + 1
end while
swaps ← swaps + calc_swap(c_order, l_order)
return swaps
```

Figure 4. Hirata II sequential algorithm

in the evaluation of f_1 needed for the calculation of objective function (2).

All parallel algorithms implement a branch and bound technique. Figure 7 shows an example of the number of nodes evaluated by the original algorithm. Figure 8 shows a decrease in the number of evaluated nodes when implementing a branch and bound technique.

Algorithm H2-P is a parallel version based on the scheme of task division. A task is an evaluation of a candidate, i.e. the calculation of the term f_1 of the objective function given by (2).

H2-X algorithm is a hybrid parallel version scheme based on task division and problem partitioning. This algorithm first divides the problem into X parts. Then, the algorithm is applied in parallel to each part of the problem. In what follows, two values of X are used: $X = 2$ and $X = 5$.

Following Frutos suggestion [6], objective function is modified to use different weights as follows:

$$f(n_{i,j}) = P_1 * f_1(n_{i,j}) + P_2 * f_2(n_{i,j}) + P_3 * f_3(n_{i,j}) \quad (2)$$

where P_1, P_2 and P_3 are weights satisfying the relation $P_1 + P_2 + P_3 = 1$. This paper only considers the special cases presented in Table I.

 ISBN: 978-1-61208-431-2

Algorithm 2 H2-P

Require: N,K,w
- $l\_order \leftarrow \{0, 1, 2, ..., N-1\}$
- $c\_order \leftarrow l\_order$
- $i \leftarrow 0$
- **while** $i < K$ **do**
  - **if** $Gates_i$ is not LNN **then**
    - $n \leftarrow makeCandidates(c_order, Gates_i)$
    - **foreach** n **in** n_{ij} **do in parallel**
      - $evaluate\_objective\_function(n_{ij})$
    - **end foreach**
    - $barrier()$ //awaiting finalization of processes
    - $S \leftarrow \emptyset$
    - $S \leftarrow getBestCandidates(n_i)$ //S is the set of best candidates
    - **if** $Gates_i$ is not LNN **then**
      - $c\_order \leftarrow random(S)$
    - **else**
      - $c\_order \leftarrow S_0$ //S_0 is the only element of S
    - **end if**
  - **else**
    - $i \leftarrow i + 1$
  - **end if**
- **end while**

Figure 5. First proposed parallel algorithm H2-P

Algorithm 3 H2-X

Require: N,K,w,X
- $l\_order \leftarrow \{0, 1, 2, ..., N-1\}$
- $c\_order \leftarrow l\_order$
- $swaps \leftarrow 0$
- **for** $i = 1$ to $i = X$ **do**
  - executed in parallel $H2\_P(N, \frac{K}{X}, w)$ //problem is divided into X parts
- **end for**
- barrier() //awaiting finalization of Initiators processes
- $swaps = swaps_1$ //improves performance and allows the construction of the solution in a single cycle
- **for** $i = 2$ to $i = X$ **do**
  - $swaps = swaps + dist(swaps_i, swaps_{i-1})$ //distance between the partial solutions
  - $swaps = swaps_i + swaps$ //$swaps_i$ is the number of gates needed to resolve the i section of the problem
- **end for**

Figure 6. Second parallel algorithm proposed H2-X

TABLE I. COMBINATION OF P_1, P_2 Y P_3 USED IN THE REPORTED TESTS

P_1	P_2	P_3	Comments
1	0	0	only $f_1()$ is use
0	1	0	only $f_2()$ is use
0	0	1	only $f_3()$ is use
1/3	1/3	1/3	HIRATA II
0.5	0.25	0.25	$f_1()$ is more important
0.25	0.5	0.25	$f_2()$ is more important
0.25	0.25	0.5	$f_3()$ is more important

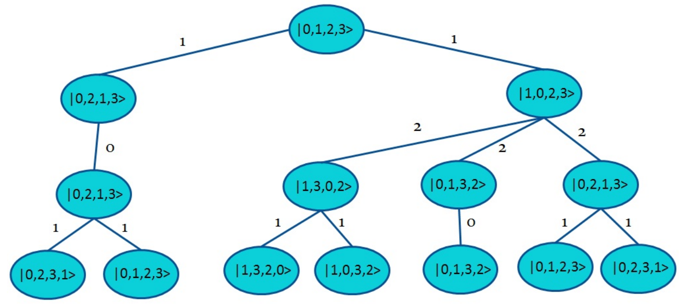

Figure 7. Nodes evaluated with the original $local\_search$ function

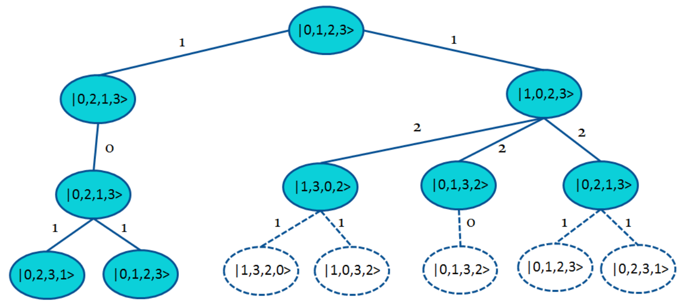

Figure 8. Nodes evaluated with the $local\_search$ function and branch and bound technique

Clearly, combination of weights where $P_1 = P_2 = P_3 = \frac{1}{3}$ corresponds to the original optimization function proposed in [11].

The nature of the parallel algorithm conversion requires the development of a communication protocol as OpenMPI [7] used in this work. The communication protocol has an Initiator, a Router and Worker processes:

- **Initiator:** the Initiator process determines the extent of the problem;
- **Router:** the Router process manages the pool of worker processes;
- **Worker:** worker processes run tasks, and even the Initiator process is also a worker process.

This protocol avoids the scheme "master / slave" implementing a scheme of *pool of workers*.

Table II summarizes the implemented algorithms and techniques used for each algorithm.

TABLE II. COMPARISON ALL IMPLEMENT ALGORITHM

Algorithm	Branch and Bound	Parallelization of calculation of objective function	Circuit partitioning	Comments
Hirata II	No	No	No	Hirata et al. [11]
H2-S	Yes	No	No	[11] with branch and bound: Algorithm (1)
H2-P	Yes	Yes	No	Objective function calculated in parallel
H2-X	Yes	Yes	Yes	Algorithm (2)

TABLE III. RESULTS OBSERVED FOR SPECIAL WEIGHTS IN THE SHOR10 CIRCUIT

Weight	Alg.	w=10		w=12		w=14	
		$\overline{swaps}$ / $\sigma(swaps)$	$\overline{t_{seg}}$ / $\sigma(t_{seg})$	$\overline{swaps}$ / $\sigma(swaps)$	$\overline{t_{seg}}$ / $\sigma(t_{seg})$	$\overline{swaps}$ / $\sigma(swaps)$	$\overline{t_{seg}}$ / $\sigma(t_{seg})$
1/3; 1/3 ;1/3	H2-S	163615,4 / 2552,19	136,851 / 2,80	150355,8 / 1050,52	558,765 / 2,28	148364,1 / 982,27	2656,19 / 2,32
	H2-P	153674,2 / 1954,26	43,694 / 1,93	140800,8 / 779,10	177,602 / 3,91	141167,4 / 761,14	745,632 / 40,03
	H2-X2	154718,8 / 2134,27	22,073 / 0,87	142723,4 / 902,84	88,617 / 4,25	141018,2 / 823,06	365,461 / 28,01
	H2-X5	152410,6 / 1709,77	10,448 / 1,21	145624,6 / 1583,82	41,144 / 2,35	145109,2 / 1498,90	147,06 / 8,44
0.5 ;0.25; 0.25	H2-S	145609 / 749,12	98,886 / 1,21	139809,2 / 633,56	388,584 / 1,63	140433,6 / 513,01	3186,251 / 4,08
	H2-P	154321,8 / 1612,67	44,781 / 1,28	140918,8 / 799,92	175,399 / 5,54	140989,6 / 580,96	744,693 / 56,84
	H2-X2	154944,6 / 1650,27	22,191 / 0,65	142394,6 / 1284,04	88,942 / 3,99	141539,6 / 466,87	359,981 / 21,28
	H2-X5	152727,8 / 937,09	9,453 / 0,43	144842,4 / 1704,23	38,127 / 1,46	144829 / 1072,02	140,265 / 14,25
0.25 ;0.5 ;0.25	H2-S	171704,6 / 625,71	146,504 / 2,25	171464,4 / 617,81	570,564 / 1,96	165006,2 / 1344,39	2039,95 / 3,02
	H2-P	154272,6 / 2316,71	43,648 / 2,07	140634,4 / 458,33	178,132 / 4,24	141376,8 / 416,88	768,926 / 78,55
	H2-X2	154401,6 / 1717,95	22,29 / 0,95	142341,4 / 639,11	92,126 / 3,39	141263,6 / 389,22	366,553 / 19,26
	H2-X5	153910,7 / 1978,02	9,881 / 0,74	143864,7 / 931,65	39,732 / 2,36	145080,2 / 1535,01	137,357 / 15,53
0.25 ;0.25 ;0.5	H2-S	162204,9 / 1729,59	118,83 / 1,36	153733,5 / 1042,29	685,323 / 1,79	157097,5 / 464,64	3097,569 / 2,31
	H2-P	153339,6 / 1755,63	43,842 / 2,36	140970 / 810,46	178,819 / 3,85	141207,2 / 757,31	718,053 / 56,23
	H2-X2	155090,2 / 1216,85	21,899 / 0,83	142839,2 / 652,68	90,129 / 3,60	141261 / 715,41	357,025 / 29,05
	H2-X5	153845,9 / 1507,76	9,774 / 0,58	145869,5 / 1366,39	45,245 / 1,98	144738,2 / 1066,47	148,415 / 15,39
1 ;0 ;0	H2-S	141604,1 / 327,71	101,027 / 0,98	136209,1 / 699,13	379,494 / 2,56	136456,7 / 396,55	3515,705 / 3,51
	H2-P	153671,4 / 1582,54	44,062 / 2,50	140613,4 / 567,89	176,709 / 4,68	141174 / 688,45	730,881 / 30,94
	H2-X2	154514,4 / 1738,62	21,916 / 0,94	142319,8 / 706,41	91,789 / 3,41	140975,6 / 548,79	361,095 / 29,26
	H2-X5	153853,9 / 1720,86	9,758 / 0,59	144962,9 / 1528,26	42,392 / 2,88	144832,2 / 1110,51	146,448 / 15,09
0 ;1 ;0	H2-S	256266,7 / 1113,46	454,973 / 1,94	258657,4 / 661,86	1851,703 / 2,12	259102,6 / 539,50	5851,167 / 5,14
	H2-P	272585,9 / 2034,55	189,944 / 13,12	274059,9 / 1370,88	700,717 / 19,88	282117,3 / 918,42	2442,572 / 72,16
	H2-X2	270386,3 / 3346,48	93,088 / 5,22	278743,7 / 1706,26	351,268 / 18,94	281370,6 / 2738,40	1454,563 / 132,50
	H2-X5	270821,2 / 1947,30	43,842 / 3,70	276104,8 / 1870,61	183,48 / 8,10	282341,2 / 3062,77	632,463 / 10,58
0 ;0 ;1	H2-S	252368,7 / 930,42	408,199 / 1,38	257750 / 625,07	2216,577 / 3,39	257017,4 / 289,33	5996,26 / 2,80
	H2-P	274425 / 1725,57	195,747 / 7,80	277003,2 / 3693,99	721,374 / 9,74	281502,6 / 2514,29	2518,742 / 38,91
	H2-X2	271276,2 / 1927,95	91,79 / 2,09	277446,3 / 1016,30	355 / 15,92	283420,9 / 2629,49	1.457 / 120,07
	H2-X5	271754,1 / 3659,20	41,78 / 3,27	276948,9 / 1597,44	186,845 / 1,44	283434,3 / 1688,50	625 / 12,03

V. EXPERIMENTAL RESULTS

The largest circuit presented in [11] is used in this work, this circuit correspond to Shor's factorization algorithm [14]. It is composed of 24 qubits and 132,204 quantum gates (named as Shor10 circuit). Given that the algorithms are probabilistic when there is a tie, experiments are run 10 times for each algorithm implemented considering three values of w, giving a total of 10x7x4x3 = 840 experimental runs.

Algorithm H2-S was run in a computer with Intel processor I7 Quadcore 2.3 GHz and 16 GB of Random Access Memory (RAM). On the other hand, parallel algorithms were executed on a cluster of computers with Intel processor I5 Quadcore 2.3 GHz and 4 GB of RAM. It is important to note that the equipment used for the execution of H2-S is clearly better than the one used for the parallel implementations.

TABLE IV. SPEEDUP AND QUALITY MEASURE COMPARISON

Weight	Algorithm	w=10		w=12		w=14	
		Q	S_p	Q	S_p	Q	S_p
1/3;1/3;1/3	H2-S	1,00	1,00	1,00	1,00	1,00	1,00
	H2-P	0,94	3,13	0,94	3,15	0,95	3,56
	H2-X2	0,95	6,20	0,95	6,31	0,95	7,27
	H2-X5	0,93	13,10	0,97	13,58	0,98	18,06
0.5;0.25;0.25	H2-S	1,00	1,00	1,00	1,00	1,00	1,00
	H2-P	1,06	2,21	1,01	2,22	1,00	4,28
	H2-X2	1,06	4,46	1,02	4,37	1,01	8,85
	H2-X5	1,05	10,46	1,04	10,19	1,03	22,72
0.25;0.5;0.25	H2-S	1,00	1,00	1,00	1,00	1,00	1,00
	H2-P	0,90	3,36	0,82	3,20	0,86	2,65
	H2-X2	0,90	6,57	0,83	6,19	0,86	5,57
	H2-X5	0,90	14,83	0,84	14,36	0,88	14,85
0.25;0.25;0.5	H2-S	1,00	1,00	1,00	1,00	1,00	1,00
	H2-P	0,95	2,71	0,92	3,83	0,90	4,31
	H2-X2	0,96	5,43	0,93	7,60	0,90	8,68
	H2-X5	0,95	6,60	0,95	15,15	0,92	20,87
1;0;0	H2-S	1,00	1,00	1,00	1,00	1,00	1,00
	H2-P	1,09	2,29	1,03	2,15	1,03	4,81
	H2-X2	1,09	4,61	1,04	4,13	1,03	9,74
	H2-X5	1,09	10,35	1,06	8,95	1,06	24,01
0;1;0	H2-S	1,00	1,00	1,00	1,00	1,00	1,00
	H2-P	1,06	2,40	1,06	2,64	1,09	2,40
	H2-X2	1,06	4,89	1,08	5,27	1,09	4,02
	H2-X5	1,06	10,38	1,07	10,09	1,09	9,25
0;0;1	H2-S	1,00	1,00	1,00	1,00	1,00	1,00
	H2-P	1,09	2,09	1,07	3,07	1,10	2,38
	H2-X2	1,07	4,45	1,08	6,24	1,10	4,12
	H2-X5	1,08	9,77	1,07	11,86	1,10	9,59

Table III describes the mean values and standard deviations observed for different weight combinations. In particular, Table III shows that H2-X5 is strictly better in running time and number of SWAP gates than the sequential version of HIRATA II algorithm (found when all weights are equal $\frac{1}{3}$). Even more, H2-X5 has a shorter running time than any other implemented algorithm, proving its effectiveness. At the same time, all implemented parallel algorithms proved to be quite competitive with respect to the sequential algorithm H2-S.

The standard deviation observed in Table III is small in general, showing that no peaks are seen neither in the running time nor in the necessary amount of quantum gates.

Table IV shows the SpeedUp (S_p) [12] and quality (Q) calculated for each parallel algorithm with respect to algorithm H2-S. Clearly, parallel algorithms have the highest acceleration and H2-X5 is the best parallel alternative.
Finally, it can be noticed in Table III that the worst results were obtained when $P_1 = 0$, confirming same conclusion reported in [6].

 ISBN: 978-1-61208-431-2

VI. Conclusion and Future Works

Experimental results show a speedup of an order of magnitude with respect to the resolution times (from hours to seconds), even improving slightly the quality of the converted circuit, measured by the number of inserted swap gates.

Experiments corroborate the importance of the term f_1 in (1) and (2) for the quality of results. Combinations where $P_1 = 0$ obtained the worst experimental results, confirming Frutos conclusion [6].

The communication protocol designed and implemented in this work using Open MPI seems very efficient and it can be used in building other non-trivial parallel algorithms.

For future work, the authors are working on the following improvements: (i) apply meta-heuristic techniques, possibly based on a strategy of task division, using the communication protocol designed and implemented in this work; (ii) modify HIRATA II algorithm to avoid the randomness in the selection of candidates in tie situations and (iii) apply further partitioning to solve a give partition to increase the potential of using parallelism in new multi-core cluster of cumputers.

Acknowledgment

Special thanks to the Nara Institute of Science and Technology (NAIST) in the person of Yasuhiko Nakashima, Hiroyuki Seki, Masaki Nakanishi, Shigeru Yamashita and Yuichi Hirata, for all help and for providing the test circuits.

References

[1] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines," Journal of Statistical Physics, vol. 22, no. 5, pp. 563–591, 1980.
[2] H. Charles Bennett, et al. "Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels," Physical Review Letters, vol. 70, no. 13, p. 1895, 1993.
[3] P. A. Maurice Dirac, "A new notation for quantum mechanics," In Mathematical Proceedings of the Cambridge Philosophical Society, vol. 35, Cambridge Univ Press, pp. 416-418, July 1939.
[4] R. P. Feynman, (1982). "Simulating physics with computers," International journal of theoretical physics, vol. 21, no. 6, pp.467-488, 1982.
[5] A. G. Fowler, S. J. Devitt and L. C. Hollenberg, "Implementation of Shor's algorithm on a linear nearest neighbour qubit array," arXiv preprint quant-ph/0402196, 2004.
[6] L. Frutos, "Conversion de circuitos cunticos a la arquitectura LNN," Enginering Thesis at Universidad Nacional de Asuncion - Facultad Politecnica, 2012.
[7] E. Gabriel, et al. "Open MPI: Goals, concept, and design of a next generation MPI implementation," In Recent Advances in Parallel Virtual Machine and Message Passing Interface, pp. 97-104, Springer, 2004.
[8] R. L. Graham, T. S. Woodall and J. M. Squyres, "Open MPI: A flexible high performance MPI," In Parallel Processing and Applied Mathematics, pp. 228-239, Springer, 2006.
[9] W. Gropp, E. Lusk and R. Thakur, "Using MPI-2: Advanced features of the message-passing interface," MIT press, 1999.
[10] L. K. Grover, "A fast quantum mechanical algorithm for database search," In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pp. 212-219, ACM, 1996.
[11] Y. Hirata, M. Nakanishi, S. Yamashita and Y. Nakashima, "An efficient method to convert arbitrary quantum circuits to ones on a linear nearest neighbor architecture," In Quantum, Nano, and Micro Technologies, First International Conference, pp. 26-33, IEEE, 2009.
[12] V. Kumar, A. Grama, A. Gupta and G. Karypis, "Introduction to parallel computing: design and analysis of algorithms," Benjamin/Cummings Publishing Company Redwood City, CA, 1994.
[13] M. A. Nielsen and I. L. Chuang, "Quantum computation and quantum information," Cambridge university press, 2010.
[14] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM journal on computing, vol. 26, no. 5, pp. 1484-1509, 1997.
[15] M. Snir, "MPI the Complete Reference: The MPI core," vol. 1, MIT press, 1998.
[16] S. Yamashita and I. L. Markov, "Fast equivalence-checking for quantum circuits," In Proceedings of the 2010 IEEE/ACM International Symposium on Nanoscale Architectures, pp. 23-28, IEEE Press, 2010.
[17] R. L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, vol. 21, no. 2, pp. 120-126, 1978.
[18] D. Deutsh, "Quantum theory, the Church-Turing principle and the universal quantum computer," The Royal Society, 1985.
[19] J. Clausen, "Branch and bound algorithms-principles and examples," Department of Computer Science, University of Copenhagen, 1999.

 ISBN: 978-1-61208-431-2