GPGPU Based Simulations for One and Two Dimensional Quantum Walks

Marek Sawerwain and Roman Gielerak

Institute of Control & Computation Engineering
University of Zielona Góra, ul. Podgórna 50, Zielona Góra 65-246, Poland
{M.Sawerwain,R.Gielerak}@issi.uz.zgora.pl

Abstract. Simulations of standard 1D and 2D quantum walks have been performed within Quantum Computer Simulator (QCS system) environment and with the use of GPGPU (General Purpose Graphics Processor Unit) supported by CUDA (Compute Unified Device Architecture) technology. In particular, simulations of quantum walks may be seen as an appropriate benchmarks for testing computational power of the processors used. It was demonstrated by a series of tests that the use of CUDA based technology radically increases the computational power compared to the standard CPU based computations.

Keywords: one and two dimensional quantum walks, simulation of quantum walks on gpgpu, CUDA technology.

1 Introduction

Recently the concept of quantum walks attracted a big attention as they provide us with a very promising source of ideas for constructing new quantum algorithms [1], [2], [3]. In particular exponential speedups of some classical problems have been discovered again (like in the Shor's algorithm case), the exponentially faster hitting described in [4] and [5] seems to be a good example for this. Additionally, certain although not so particular as exponential speedups are, speedups of some classical problems like k-distinction problem, triangle and k-clique algorithms are examples of them [6],[1],[7] have been obtained by the use of quantum walks concept. Another inspirations for studying quantum walks based applications may come from the observation [8], [9] that quantum walks might play the role of universal quantum calculation tool.

It is the main purpose of the present contribution to test some particular properties of quantum walks by using certain simulating tools. The main tools used for our simulations are the Zielona Góra Quantum Computer Simulator (as the main tool) and the GPGPU equipped with new computational technology offered by CUDA.

It is one of the main result of this note the demonstration of powerful incrementation of calculational abilities if we use GPGPU of new generation compared with standard CPU based computations.

A. Kwiecień, P. Gaj, and P. Stera (Eds.): CN 2010, CCIS 79, pp. 29–38, 2010.

Organisation of this note is as follows: in Sect. 2 we provide a reader with the basic definitions and constructions connected with quantum walks on general graphs. Numerical implementations of quantum walks on certain lattice structures coming from that of $\mathbb{Z}^1$ and $\mathbb{Z}^2$ together with the corresponding numerical algorithms are presented in Sect. 3. Additionally several examples of our simulations will be presented and discussed briefly there.

2 Some Mathematical Preliminaries

Let $G = (\mathbb{V}, \mathbb{E}, l)$ be a graph and where as obviously $\mathbb{V}$ stands for the set of vertices of G the number of elements $|\mathbb{V}|$ of which is equal to N, $\mathbb{E}$ stands for the set of (undirected) edges of G and $l : \mathbb{E} \to \{\mathbb{V}, \mathbb{V}\}$ is the edges labelling function. The corresponding incidence matrix of G will denoted as M_G. With the use of M_G the continuous time Markov walk on G can be described by the corresponding, canonical Markovian transition semi-group $T_t = \exp(-tM_G)$ and (eventually) an initial distribution π_0. For any $v \in \mathbb{V}$ we denote by $d(v)$ the corresponding degree of vertex v and let then $\mathbb{P}(v) = (P_1, P_2, \ldots, P_{d(v)})$ be a discrete probability measure assigning a probability that the walker jump with probability p_i by the use of i-th edge e_i connecting the vertex v with $l(e_i)(2) = \omega$. The system $(\mathbb{P}, \pi_0)$, where π_0 is an initial distribution gives rise to the discrete step random walk on G.

The corresponding quantum walks on G can be constructed in the following way.

The continuous time quantum walk on G: by the very definition starting from the vertex v_0 (with probability π_0) after time t we arrive at the vertex obtained by the action of the unitary group $U_t^G = \exp(itH_G)$ (where H_G stands for the corresponding graph Hamiltonian) acting in the Hilbert space $\mathcal{H}^G = \oplus_{v\in\mathbb{V}}|v\rangle$. However only the discrete time processes will be discussed in this note.

2.1 The Discrete Time Markovian Quantum Walks

Let$\mathcal{H}^G = \oplus_{v\in\mathbb{V}}|v\rangle$ be the canonical Hilbert space associated with G, obviously $\mathcal{H}^G \simeq \mathbb{C}^N$. For any $v \in \mathbb{V}$ let $d(v)$ be degree of v. Then the local Hilbert space $\mathcal{H}_v$ is defined as a space isomorphic to $\mathbb{C}^{d(v)}$, explicitly $\mathcal{H}_v = \oplus_{e_u}|e_u\rangle$, e_u runs over all edges connecting the vertex v with others. A collections $\mathbb{C} = (C_v, v \in \mathbb{V})$ of unitary maps acting on the spaces $\mathcal{H}_v$, $v \in \mathbb{V}$ and fulfilling additionally certain natural coincidence conditions, see i.e. [5]; will be called a "coin flip transformation sequence". In other words, for any $v \in \mathbb{V}$:

$$\begin{array}{c} C_v : \ |v\rangle \otimes \mathcal{H}_v \to |v\rangle \otimes \mathcal{H}_v \\ \text{where } |v\rangle \otimes |\omega\rangle \to |v\rangle \otimes C_v|\omega\rangle \ . \end{array} \tag{1}$$

The global Hilbert space $\mathcal{H}$ is defined as $\mathcal{H} = \oplus_{v\in\mathbb{V}}|v\rangle\otimes\mathcal{H}_v$ and the corresponding discrete quantum walk on G, providing the family $\mathbb{C}$ is given, can be defined by its one step transformation:

$$U = S(I \otimes \mathbb{C}) \tag{2}$$

where the shift transformation S is defined as:

$$S|v,e\rangle = |v',e\rangle \quad \text{if} \quad l(e) = \{v,v'\} \ . \tag{3}$$

Several questions (with analogy to the classical case, especially the problems connected to the mixing and hitting times on a large class of graphs have been studied intensively) can be studied, the question about limiting probability distributions and hitting times are amoung the most popular one [10].

Although intensive simulations of quantum walks on many complex graphs are planned to be done we have concentrated first on some simplest quantum walks on the infinite (the finite amount of with appropriate boundary condition are considered in real simulation tasks of course) graphs $\mathbb{Z}^1$ and $\mathbb{Z}^2$ that we describe now.

2.2 Quantum Walks on Lattice $\mathbb{Z}^1$ and on Lattice $\mathbb{Z}^2$

With the lattice $\mathbb{Z}^1$ we associate the Hilbert space $l_2(\mathbb{Z}) = \oplus_{n\in\mathbb{Z}}|n\rangle$ and the coin flip transformation $\mathbb{C}$ acting in $\mathbb{C}^2 \equiv |R\rangle \oplus |L\rangle$ symbolising the possible steps in the right ($|R\rangle$) or left direction ($|L\rangle$) is given. Then the corresponding Hilbert space $l_2(\mathbb{Z}) \otimes \mathbb{C}^2$ can be seen as a space of infinite sequences $|\psi\rangle \approx ((\alpha_{jk}), j \in \mathbb{Z},\ k \in \{R,L\})$, i.e. any vector $|\psi\rangle \in l_2(\mathbb{Z}) \otimes \mathbb{C}^2$ can be given by:

$$|\psi\rangle = \sum_{j\in\mathbb{Z},k=L,R} \alpha_{jk}|j,k\rangle \quad \text{where} \quad \sum_{j,k} |\alpha_{jk}|^2 = 1 \ . \tag{4}$$

Different choices of $\mathbb{C}$ and shift operators lead to different models of quantum walks on the lattice $\mathbb{Z}^1$.

The graph Hilbert space $\mathcal{H}^{\mathbb{Z}^2}$ for the 2D lattice $\mathbb{Z}^2$ is defined as

$$\mathcal{H}^{\mathbb{H}_2} = \oplus_{j,m\in Z}|j,m\rangle = l_2(\mathbb{Z}^2) \ . \tag{5}$$

The degrees of all vertices are equal to 4 and therefore the local Hilbert spaces are isomorphic with $\mathbb{C}^4$ to be identified with R,L,U,D (right, left, up, down) steps on the lattice. The total space $\mathcal{H} = l_2(\mathbb{Z}^2) \otimes \mathbb{C}^4$ and the typical vector $|\psi\rangle \in \mathcal{H}$ can be decomposed as

$$|\psi\rangle = \sum_{j,k=1}^{4} \sum_{m,n=-\infty}^{+\infty} \alpha_{j,k,m,n}|j,k\rangle|m,n\rangle \quad \text{with} \quad \sum_{j,k=1}^{4} \sum_{m,n=-\infty}^{+\infty} |\alpha_{j,k,m,n}|^2 = 1 \ . \tag{6}$$

A different versions of the corresponding coin flip transformation $\mathbb{C}$ and the shift operators (reflecting some additional topological constraints) then lead to different quantum walk models on $\mathbb{Z}^2$ lattice. Some of them will be presented for simulations performed in the next section including some quantum walk models on $\mathbb{Z}^1$ as well.

3 The Algorithm for Simulating Quantum Walks on GPGPU

The calculation routine for simulation of quantum walks can be build directly from the definition of state of quantum walker walking on the lattice $\mathbb{Z}^2$. In general a state of the quantum walker at time t is given in the following way:

$$|\psi(t)\rangle = \sum_{j,k=0}^{1} \sum_{m,n=-\infty}^{+\infty} \alpha_{j,k,m,n}(t)|j,k\rangle|m,n\rangle \tag{7}$$

where $\alpha_{j,k,m,n}(t) \in \mathbb{C}$ and $\sum_{j,k}\sum_{m,n}|\alpha_{j,k,m,n}(t)|^2 = 1$. The evolution of the quantum walker system over time t is expressed by following unitary operator

$$U = S(C \otimes I) \tag{8}$$

where S is the shift operator, I represents the identity operator and C is the coin operator (in most cases we can assume that the coin is represented by Hadamard operator, but there exist other representations of the coin operator e.g. Fourier and Grover coins) which acts on the local $\mathcal{H}_2 \otimes \mathcal{H}_2$ subspace of whole walker Hilbert space system $l_2(\mathbb{Z}^2) \otimes \mathcal{H}_2 \otimes \mathcal{H}_2$.

In this contribution we propose rather special definition of shift operator for two-dimensional quantum walks. A comparison of our definition with those used frequently can be found in [11], [12] and [13]. We also use the random broken links (termed RBL) technique, first developed for one dimensional quantum walks and introduced in [14]. The RBL technique was generalised for two-dimensional case in [11].

The used definition of shift operator which coincides with physical and mathematical lattice is the following

$$S = \sum_{j,k=0}^{1} \sum_{m,n=-\infty}^{+\infty} |j,k\rangle\langle j,k| \otimes |m + (-1)^j(1-\delta_{j,k}), n + (-1)^j\delta_{j,k}\rangle\langle m,n| \tag{9}$$

where $\delta_{j,k}$ is Dirac discrete delta function.

The following function includes the possibility of appearance of the broken line in a path in between site (j,k) and (m,n) in example depicted on Fig. 1:

$$\mathcal{RBL}(j,k,m,n) = \begin{cases} (-1)^j & \text{– if link to } m + (-1)^j(1-\delta_{j,k}), \\ & \quad n + (-1)^j\delta_{j,k} \text{ is closed} \\ 0 & \text{– if link is open} \end{cases} \tag{10}$$

where $j,k \in \{0,1\}$.

After applying shift operator (9) to state (7) the evolution can be summarised in following way:

$$\psi_{(1-j,1-k,m,n)}(t+1) = \sum_{j',k'=0}^{1} C_{j+\mathcal{RBL}(j,k,m,n),k\oplus\mathcal{RBL}(j,k,m,n),j',k'} \cdot \psi_{(j',k',m+\mathcal{RBL}(j,k,m,n)(1-\delta_{j,k}),n+\mathcal{RBL}(j,k,m,n))\delta_{j,k}}(t) \tag{11}$$

where $\oplus$ represents addition modulo two.

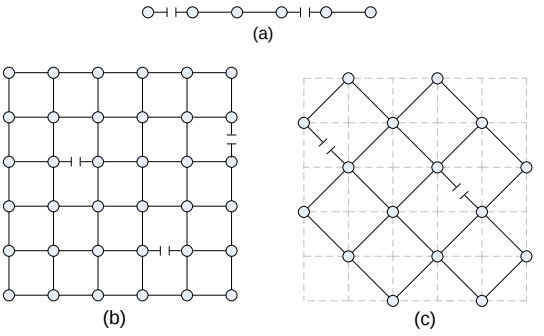

Fig. 1. The idea of broken links for one-dimensional quantum walk (a) where we show broken links between sites, part (b) and part (c) represent examples of natural and diagonal lattices for two-dimensional quantum walk

The algorithm to simulate two-dimensional quantum walks is directly basing on the evolution given by Equation (11). It can be implemented on the traditional architecture for standard CPU and of course for GPGPU based solutions.

Figure 2 shows a fragment of calculation routine for GPGPU which task is to compute values for the next iteration of quantum walk in the segment case. In each iteration all points in the segment attain a new values. This means that the all available GPGPU cores are fully used. Additionally, the efficiency can be increased by better usage of threads available in CUDA architecture.

```
__global__ void one_iteration_segment(
            cuFloatComplex *A0, cuFloatComplex *A1,
            cuFloatComplex *Atemp0, cuFloatComplex *Atemp1, int *RBL0,
            int *RBL1, cuFloatComplex *C, int N) {
    int m = blockIdx.x * blockDim.x + threadIdx.x; int L, cidx1, cidx2;

    if (m<N) {
        L = RBL0[m]; cidx1=(L*2); cidx2=(L*2)+1;
        Atemp1[m] = cuCaddf(cuCaddf(Atemp1[m], cuCmulf(C[cidx1], A0[m+L])),
                            cuCaddf(Atemp1[m], cuCmulf(C[cidx2], A1[m+L])));

        L = RBL1[m]; cidx1=((1+L)*2); cidx2=((1+L)*2)+1;
        Atemp0[m] = cuCaddf(cuCaddf(Atemp0[m], cuCmulf(C[cidx1], A0[m+L])),
                            cuCaddf(Atemp0[m], cuCmulf(C[cidx2], A1[m+L])));
    }
}
```

Fig. 2. The kernel function to compute trajectories of one-dimensional quantum walks on segment

The function to calculate the probability distribution for one-dimensional quantum walk on the line is very similar to the segment case with one important difference. In the i-th iteration the quantum walker cannot be farther than i sites from its initial position. The necessary change in GPGPU routine is expressed as:

```
int m = blockIdx.x * blockDim.x + threadIdx.x; int left, right;
left = max(midpoint - extra - iteration, 1);
right = min(midpoint + extra + iteration, N-1);
if (m>=left && m<=right) { ... }
```

The probability distributions connected to a calculated trajectory for segment and line (and also for two-dimensional cases) are not the same, which is illustrated on the Fig. (3).

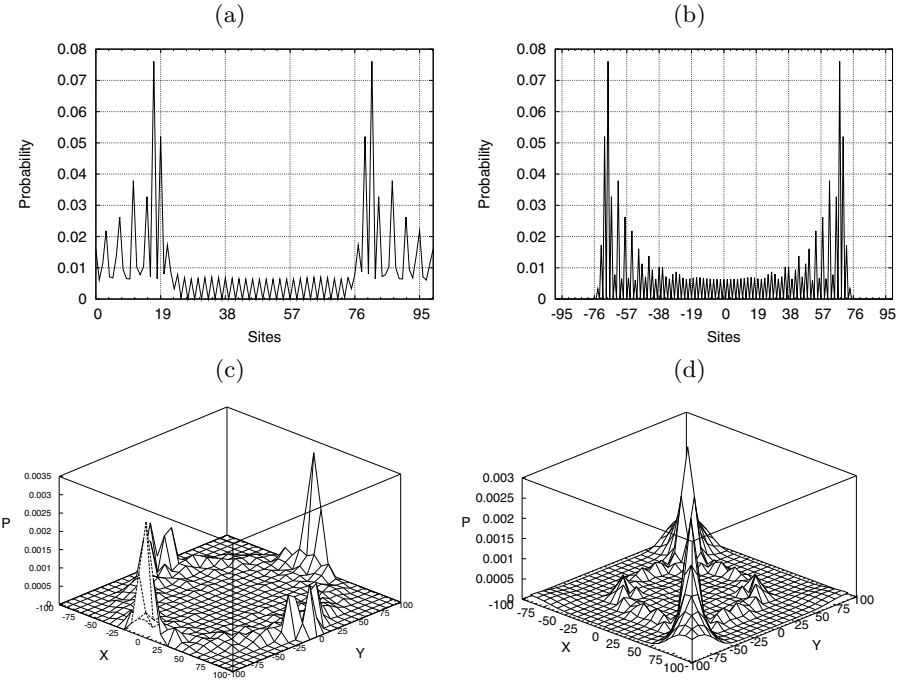

Fig. 3. The one-dimensional quantum walk on segment (a) and on line (b). In both cases number of iterations are the same and equal to 100 and probability distribution of two-dimensional quantum walks with Fourier walker and Fourier's initial state on diagonal lattice depicted on plot (c) and natural lattice depicted on plot (d).

3.1 Complexity Analysis Briefly

The computational complexity T^{1d} of simulation of quantum walk for one-dimensional case strictly depends on the size of segment or line used. Let N be a size of segment for arising in i-th iteration, then:

$$T_i^{1d}(N) = \sum_{m=0}^{N-1} \sum_{j=0}^{1} \left(T_{\mathcal{RBL}_m} \sum_{k=0}^{1} (T_{\mathrm{OP}_m}) \right) = 4 \cdot T_{\mathcal{RBL}_m} \cdot T_{\mathrm{OP}_m} \cdot N \ . \qquad (12)$$

It is possible to write Equation (12) in such a form because operations $\mathcal{RBL}_m$ and OP_m need the same constant time of work which is independent on the number of iteration. The symbol $T_{\mathcal{RBL}_m}$ stands for amount of time necessary to process information about broken links, the second symbol T_{OP_m} stands for amount of time necessary to process the probability amplitude of selected point. The use of GPGPU means, that the processing time can be divided by the number N_c of available cores, because operations $\mathcal{RBL}_m$ and OP_m are independent for each point in segment or line (as well as in two-dimensional case):

$$T^{1d}_{\mathrm{GPGPU}}(n_i) = T^{1d}_i(N)/N_c \ . \tag{13}$$

The simulation of quantum walk on line shows one important difference comparing to segment case. The point is that the new values of points are calculated only in part and the size of simulated portion strictly depends on the number of iteration. The complexity can be denoted as (where $T_{\mathcal{RBL}_m}$ and T_{OP_m} have the same sense as in previous, segment case):

$$T^{1d}_i(N) = \sum_{m=\mathrm{l}_i}^{\mathrm{r}_i} \sum_{j=0}^{1} \left(T_{\mathcal{RBL}_m} \sum_{k=0}^{1} (T_{\mathrm{OP}_m}) \right) = 4 \cdot T_{\mathcal{RBL}_m} \cdot T_{\mathrm{OP}_m} \cdot ((\mathrm{r}_i - \mathrm{l}_i) + 1) \tag{14}$$

where the l_i and r_i are given by:

$$\mathrm{l}_i = \max(\mathrm{MP} - \mathrm{ES} - \mathrm{i}, 1), \quad \mathrm{r}_i = \min(\mathrm{MP} + \mathrm{ES} + \mathrm{i}, N - 1) \tag{15}$$

where MP is the midpoint index of line, the value ES is used to pad and therefore prevents from range errors. However, these values are constant so the difference of r_i and l_i for i-th iteration can be expressed as

$$(r_i - l_i) = (2 \cdot i) + 1 \ . \tag{16}$$

In the case of small systems (ten, twenty or fifty points), Equation (16) shows that the most of available cores in GPGPU are not fully utilised. In the case of bigger systems this problem is not arising. The difference in (16) for higher iteration number is bigger and what is more the values of this difference for all iterations form the arithmetic sequence, which means that in sense of complexity theory that only linear speedup is achieved, however for segment with size of 10000 points the obtained speedup is nearly hundredfold. The corresponding times have been depicted in Tables 1, 2 and 3.

The use of Equation (11) allows to estimate computational complexity of two-dimensional quantum walk trajectory calculations in i-th iteration:

$$\begin{aligned} T^{2d}_i(N) = \sum_{m=\mathrm{lb}_i}^{\mathrm{rb}_i} \sum_{n=\mathrm{lb}_i}^{\mathrm{rb}_i} \sum_{j=0}^{1} \sum_{k=0}^{1} T_{\mathcal{RBL}_{(m,n,j,k)}} 4 \cdot T_{\mathrm{OP}_{(m,n,j,k)}} = \\ > 16 \cdot T_{\mathcal{RBL}_{(m,n,j,k)}} \cdot T_{\mathrm{OP}_{(m,n,j,k)}} \cdot ((\mathrm{rb}_i - \mathrm{lb}_i) + 1)^2 \ , \end{aligned} \tag{17}$$

where N means the length of trajectory calculated.

Table 1. The measured times of calculations of one-dimensional quantum walks trajectories for segments with different sizes (without broken links)

	Core 2 Duo 8400 (1 core)	Geforce 9600 GT (64 cores)	Geforce 280 (240 cores)
Size	Time in ms	Time in ms	Time in ms
100	6.00	4.51	2.48
1000	663.00	40.186	24.171
5000	22685.00	612.67	281.57
10000	96362.00	2324.3002	884.297

Table 2. The measured times of calculations of two-dimensional quantum walks trajectories for diagonal lattice without broken links. The measured times for the case of Core 2 Duo and two-thread computational routine are presented in bracket.

	Core 2 Duo 8400 (1 core)	Geforce 9600 GT (64 cores)	Geforce 280 (240 cores)
Size	Time in ms	Time in ms	Time in ms
100	700 (413)	195	64
200	5480 (2945)	1003	363
300	19140 (10512)	3423	1137
400	47230 (25785)	8123	3127
500	92530 (50274)	14462	4706

Table 3. The measured times of calculations of two-dimensional quantum walks trajectories for diagonal lattice without broken links. The simulation was performed on two Intel Xeon E5420 2.50 Ghz processors, the multi-threaded calculation subroutine was compiled with GCC compiler with "-O3" option.

	(1-thread)	(2th)	(4th)	(8th)
Size	Time in ms	Time in ms	Time in ms	Time in ms
100	770	409	299	206
200	6740	3281	2190	1382
300	22910	12029	7130	4848
400	56610	29853	17699	12069
500	114060	65292	33275	24602

The variables lb_i and rb_i have the same meaning as l_i, r_i introduced before and are calculated in the following way:

$$\mathrm{lb_i} = \max(\mathrm{MP} - \mathrm{ES} - \mathrm{i}, 1), \quad \mathrm{rb_i} = \min(\mathrm{MP} + \mathrm{ES} + \mathrm{i}, 2 \cdot \mathrm{MP} - 1) \ . \tag{18}$$

The expression $((\mathrm{rb}_i - \mathrm{lb}_i)+1)^2$ in Equation (17) can be expressed in the following way

$$((\mathrm{rb}_i - \mathrm{lb}_i) + 1)^2 = ((2 \cdot i) + 1)^2 \ . \tag{19}$$

Speedups obtained for 2D quantum walks presented in Fig. 4 are based on results presented in Tables 2 and 3.

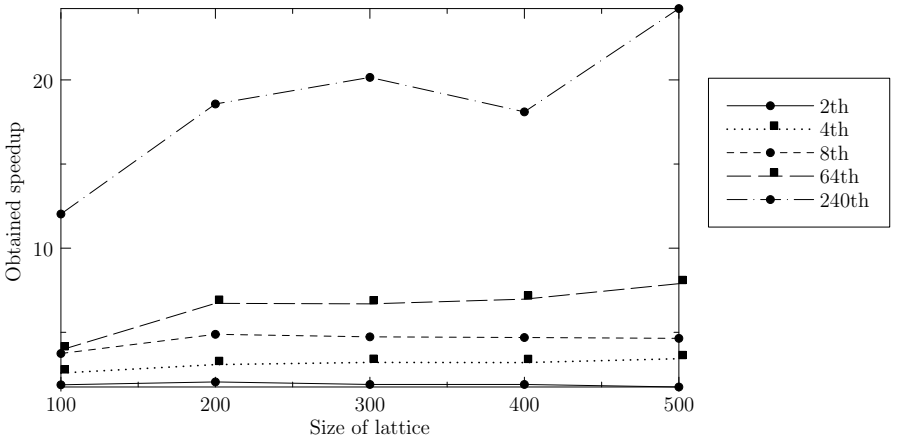

Fig. 4. The obtained values of speedup for simulations of two dimensional quantum walks on diagonal lattice. Graphs labelled as 2th, 4th and 8th are compared to one-thread computational routine. Graphs marked as 64th and 240th represent the speedup obtained by computational routine executed on the Geforce video card 9600 GT and GTX 280 respectively.

4 Conclusions and Further Work

The specialised software to simulate one and two dimensional random quantum walks without and with broken links has been presented in this article. The used software is a part of the Quantum Computing Simulator presented in [15], [16].

A significant speedup of the simulations process comparing to previous paper [13] have been achieved. The used technologies enables to simulate effectively much more complex quantum walks then previously known. Additionally, certain more deeper notions connected to the analysis of quantum walks behaviour can be analysed using computer simulations as an appropriate tool.

Acknowledgments. We acknowledge useful discussions on the QCS with the *Q-INFO* group at the Institute of Control and Computation Engineering of the University of Zielona Góra, Poland.

References

1. Szegedy, M.: Quantum Speedup of Markov Chain Based Algorithms. In: Proc. of 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 32–41 (2004)
2. Aharonov, D., Ambainis, A., Kempe, J., Vazirani, U.: Quantum walks on graphs. In: Proceedings of 33th STOC, pp. 50–59. ACM, New York (2001)
3. Nayak, A., Vishwanath, A.: Quantum Walk on the Line, `http://arxiv.org/abs/quant-ph/0010117`
4. Childs, A.M., Cleve, R., Deotto, E., Farhi, E., Gutmann, S., Spielman, D.A.: Exponential algorithmic speedup by quantum walk. In: Proc. 35th ACM Symposium on Theory of Computing, pp. 59–68 (2003)

5. Kempe, J.: Quantum random walks hit exponentially faster. In: Proceedings of 7th Intern. Workshop on Randomization and Approximation Techniques in Computer Science. LNCS, pp. 354–369. Springer, Heidelberg (2003)
6. Ambainis, A.: Quantum walks and their algorithmic applications. International Journal of Quantum Information 1(4), 507–518 (2003)
7. Wocjan, P., Abeyesinghe, A.: Speedup via quantum sampling. Phys. Rev. A 78, 042336 (2008)
8. Oliveira, A., Portugal, R., Donangelo, R.: Simulation of the single- and double-slit experiments with quantum walkers (2007), http://arxiv.org/abs/0706.3181
9. Childs, A.M.: Universal computation by quantum walk. Phys. Rev. Lett. 102, 180501 (2009)
10. Kempe, J.: Quantum random walk algorithms. Contemp. Phys. 44(3), 302–327 (2003)
11. Oliveira, A.C., Portugal, R., Donangelo, R.: Decoherence in two-dimensional quantum walks. Phys. Rev. A 74, 012312
12. Kendon, V.: Decoherence in quantum walks: a review. Math. Struct. in Comp. Sci. 17(6), 1169–1220 (2006)
13. Marquezino, F.L., Portugal, R.: The QWalk Simulator of Quantum Walks. Computer Physics Communications 179(5), 359–369 (2008)
14. Romanelli, A., Siri, R., Abal, G., Auyuanet, A., Donangelo, R.: Decoherence in the quantum walk on the line. Physica A 347C, 137–152 (2005)
15. Sawerwain, M.: Parallel algorithm for simulation of circuit and one-way quantum computation models. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J., et al. (eds.) PPAM 2007. LNCS, vol. 4967, pp. 530–539. Springer, Heidelberg (2008)
16. Sawerwain, M., Gielerak, R.: Natural quantum operational semantics with predicates. Int. J. Appl. Math. Comput. Sci. 18(3), 341–359 (2008)