Programming gate-based hardware quantum computers for music

Kirke, A

Published in:
Muzikologija

DOI:
10.2298/muz1824021k

Publication date:
2018

DOI https://doi.org/10.2298/MUZ1824021K
UDC 789.983
004.9:78

Programming Gate-based Hardware Quantum Computers for Music

Alexis Kirke[1]
University of Plymouth, School of Humanities and Performing Arts

Received: 17 April 2018
Accepted: 7 May 2018
Original scientific paper

Abstract

There have been significant attempts previously to use the equations of quantum mechanics for generating sound, and to sonify simulated quantum processes. For new forms of computation to be utilized in computer music, eventually hardware must be utilized. This has rarely happened with quantum computer music. One reason for this is that it is currently not easy to get access to such hardware. A second is that the hardware available requires some understanding of quantum computing theory. This paper moves forward the process by utilizing two hardware quantum computation systems: IBMQASM v1.1 and a D-Wave 2X. It also introduces the ideas behind the gate-based IBM system, in a way hopefully more accessible to computer-literate readers. This is a presentation of the first hybrid quantum computer algorithm, involving two hardware machines. Although neither of these algorithms explicitly utilize the promised quantum speed-ups, they are a vital first step in introducing QC to the musical field. The article also introduces some key quantum computer algorithms and discusses their possible future contribution to computer music.

Keywords: quantum computer music, algorithms, D-Wave

Introduction: Quantum Computing

Why Quantum Computing? The typical answer is speed. Quantum mechanics models the world by considering a physical state as a sum of all its possible configurations. For example, the physical state of an electron is modeled as a weighted sum of a large number of vectors (called eigenvectors), each of which represents something that could possibly happen in the physical world. This sum of vectors varies over

1 alexis.kirke@plymouth.ac.uk

time as the electron's state changes. When the electron's state is measured at a particular time in the lab, the result will be one of the vectors. Performing an operation on a physical state in quantum mechanics thus means operating on a large number of configurations simultaneously. This natural parallelism, when combined with a property known as entanglement, provides the potential for calculations whose speed far exceed those of classical computers. Certain quantum algorithms have been shown to be orders of magnitude faster than their classical versions (Shor 2006).

Another feature of quantum mechanics is its probabilistic nature. The results of measurements on an electron in general cannot be predicted with certainty. Quantum mechanics simply provides a means to calculate the probability of the electron being in a certain state. Surprising results emerge from this. In classical physics if an electron is fired at a sufficiently strong electromagnetic barrier, it will fail to penetrate it. In quantum mechanics, there is a small probability that it will be observed on the other side of the barrier. This is known as quantum tunneling. In quantum computing this tunnelling becomes relevant when building quantum annealers. Annealers can be thought of as traversing a fitness landscape looking for the global minimum. One main weakness is that the solver can get trapped in a local valley – i.e. it thinks it's at the bottom of a valley, but in fact just over the hill is a much deeper valley. However the solver can not "see" it, because of the hill. In the quantum version of this algorithm, the quantum solver can tunnel through the mountain to the lower valley, leading to potential speed-ups in solving (Neven 2016).

The non-deterministic nature of quantum computing is another reason to examine it from an artistic point of view. Artistic algorithms have utilized pseudo-random algorithms since the first computer arts up to some of the most recent. This is because randomness helps to prevent the algorithm getting stuck in an attractor or producing repeated uninteresting output. Many computer artists prefer to use complexity algorithms rather than randomness, to avoid these pitfalls – for example cellular automata. However, at the heart of many of these systems is a pseudo-random choice still. The same parameters will create the same result. So the parameters of the complex algorithm are sometimes pseudo-randomized. It has been argued that the human brain itself is at many levels non-deterministic as well as complex. Quantum computing is not pseudo-random. It is random. Like the brain may have, and many of the complexity arts algorithms, it has randomness at its heart. A quantum algorithm for which there is an expected deterministic result needs to be run multiple times to get a final output. The final output is some averaging of all the intermediate outputs. Such a form of computation provides a new way of thinking about computer arts. Rather than trying to create complexity and randomness from determinism – as in classical computing, quantum computing requires us to build determinism and complexity from randomness. The implications of this reversal of thinking for the arts are hard to imagine at this stage, but must be investigated.

It is the concept of hard-to-imagine implications that further drives research in quantum computer arts and quantum computer music. Quantum computing is, to a degree, a solution looking for a problem. Three main potentially useful algorithms have been identified, but have only been implemented in a limited sense. Developing quantum algorithms requires a new way of thinking: rotations in complex vector

spaces, probabilistic results, entanglement and superposition. But it must be asked: what are the implications of this way of thinking for the arts? We can only begin to answer these question by starting to apply basic quantum algorithms to the arts.

The structure of this article will be to provide an overview of related work. Then gate-based quantum computing will be introduced, and the algorithm GATEMEL. Finally this system will be combined with a D-WAVE quantum annealer to create the system qGEN: the first hybrid hardware quantum computer music system.

Related Work

Most previous designs for performances and music involving quantum mechanical processes have either been metaphorical, based on simulations (online or offline), or – in the case of actual real-time physics performances – not directly concerned with quantum effects.

In terms of offline simulations, the most closely related to this chapter is the web page *Listen to Quantum Computer Music* (Weimer 2010). Two pieces of music are playable online through MIDI simulations. Each is a sonification of two key quantum computation algorithms. The offline sonification of quantum mechanics equations have also been investigated by Sturm (2000; 2001) and O'Flaherty (2009), with the third being an attempt to create a musical signature for the Higgs Boson at CERN before its discovery. Another paper defines what it calls Quantum Music (Putz and Svozil 2017), though once again this is by analogy to the equations of quantum mechanics, rather than directly concerned with quantum physics. Certain equations of quantum mechanics have also been used to synthesize new sounds (Cadiz and Ramos 2014). The orchestral piece *Music of the Quantum* (Coleman 2003) was written as an outreach tool for a physics research group, and has been performed multiple times. The melody is carried between violin and accordion. The aim of this was as a metaphor for the wave particle duality of quantum mechanics, using two contrasting instruments.

The most impressive quantum simulation performance has been *Danceroom Spectroscopy* (Glowacki et al. 2012) in which quantum molecular models generate live visuals. Dancers are tracked by camera and their movements treated as the movement of active particles in the real-time molecular model. Thus the dancers act as a mathematically accurate force field on the particles, and these results are seen in large scale animations around the dancers.

There have been performances and music that use real-world quantum-related data. However most of these have been done offline, rather than using physics occurring during the performance. These include the piece *Background Count:* a prerecorded electroacoustic composition that incorporates historical Geiger counter data into its creation (Brody 1997). Another sonification of real physics data done offline was the LHChamber Music project (Anon. 2014). It was instrumented for a harp, a guitar, two violins, a keyboard, a clarinet and a flute. Different instruments played data from different experiments. Flute and guitar were CMS, Clarinet and Violin I were ATLAS, Violin II was LHCb, Piano was ALICE, and harp was CCC.

The first real-time use of subatomic physics for a public performance was *Cloud Chamber* (Kirke 2011). In *Cloud Chamber* physical cosmic rays are made visible in real-time, and some of them are tracked by visual recognition and turned in to sound. A violin plays along with this, and in some versions of the performance, the violin triggered a continuous electric voltage that change the subatomic particle tracks, and thus the sounds (creating a form of duet). *Cloud Chamber* was followed a few years later by a CERN-created system which worked directly, without the need to use a camera. Called the *Cosmic Piano* it detects cosmic rays using metal plates and turns them into sound (Culpan 2015). The previous two discussed performances were live, and the data was not quantum as such. It was quantum-related in that the cosmic rays and cloud chambers are subatomic quantum processes. But they do not incorporate actual quantum computation in their music.

The first use of hardware quantum computers to make music was the algorithm qHARMONY (Kirke 2016) which was implemented on an adiabatic quantum computer and also utilized in a live music performance with a mezzo-soprano.[2]

In this paper I will present qGEN which is the first designed from the ground-up QC music algorithm using both adiabatic and gate-based quantum computers.

Music and Gate-based Quantum Computers

Gate-based quantum computers are the most well-known, but least commercially developed quantum computers. One gate-based quantum computer is available commercially, a 17 qubit machine by IBM. But even this is a commercial proof-of-concept rather than retail quantum computing. In this section some results of music generation with a hardware gate-based QC are presented. The promise of gate-based quantum computers is, although they are not yet available, they have been theoretically demonstrated to be incredibly powerful. Gate-based QCs utilize many of the elements familiar to those who know about traditional computation – for example NOT-type logic gates. However they are also more complex in that the advantages of the gate-based approach over the classical requires some mathematical understanding of complex vector spaces and linear algebra. This article will endeavor to introduce the concepts to a broader audience using the simplest possible music generation system.

I will start by introducing the quantum gates that will be used. The system GATEMEL will be implemented in IBMQASM. This language can be used by expert users to access a small hardware quantum computer.

Five Qubit Computer

The equivalent to the basic unit of classical computation – the bit – in quantum computing is the qubit. A qubit is a quantum bit. As was mentioned the quantum

2 Alexis Kirke and Juliette Pochin, "Superposition" https://www.youtube.com/watch?v=-S5hU40MWag

electron is a sum of all its possible states, and only a measurement shows which state it is in. Similarly a qubit is a weighted combination of both possible bit values: 0 and 1. The precise value of a qubit is not know until measured by something outside of the quantum system.

The IBM quantum computer used in this paper is a 5 qubit system. It is housed in a large dilution refrigerator, supported by multiple racks of electronic pulse-generating equipment. The qubits used are known as fixed-frequency superconducting transmon qubit, and are Josephson-junction based to reduce noise effects. They use fixed-frequency qubits to minimize sensitivity to external magnetic field fluctuations that could corrupt the quantum information.

The superconducting qubits are made on silicon wafers with superconducting metals such as aluminium and niobium. The processor is contained in a printed circuit board package shielded within a light-tight, magnetic-field shielding can. The dilution refrigerator cools the device down to around 15 miliKelvin. It works by circulating a mixture of two Helium isotopes. Electromagnetic impulses at microwave frequencies are sent to the qubits through coaxial cables with a particular phase, duration, and frequency. These enact the quantum gates.

To measure the qubits, each is coupled to a microwave resonator. A microwave tone is sent to the resonators, and the qubits state can be retrievied from the phase and amplitude of this reflected signal. Signals in the resonator are amplified within the dilution refrigeration layers: a quantum-limited amplifier at 15 mK, and a high-electron mobility transistor amplifier at 4K. The system is re-tuned three times a day, which takes up to an hour.

The IBM gate-based computer has topological limitations. Specifically the only controllable connectivity between qubits is via qubit 4. So qubit 4 is connected to qubits 0, 1, 2 and 3; but none of 0,1,2,3 are connected to each other. This also needs to be taken into account when designing GATEMEL. The gates which make up GATEMEL are now introduced, which also provides an introduction to gate-based QC.

Despite the simplicity of the IBMQASM 1.1 system, it has been used for practical quantum computing research, for example Google's post-quantum cryptography (Malloy et al 2016).

Quantum States and Gates

The general form of a simple qubit, the weighted sum of states, is written in the form:

$$q = a|0> + b|1>$$

where a and b are the weights. $|0>$ and $|1>$ are known as 'kets' and represent vectors in a complex vector space. This is called a superposition of a 0 and a 1 state. The axioms of quantum mechanics say that the probability of measuring the qubit as 0 is $|a|^2$ and the probability of measuring the qubit as 1 is $|b|^2$. This is as much as we can know about this qubit. So QC is essentially non-deterministic. In fact the

non-determinism is deeper than that. It has been shown that this non-determinism is not because there are hidden factors we are not taking into account. For example, the bizarre orbits of the planets around the earth were eventually explained and simplified by the knowledge that they were orbiting around the sun not the earth. There are no properties of the qubit or its surrounding system (usually called 'hidden variables') which would enable us to 100% determine the qubit's measured value before measuring.

A quantum computing system with two qubits would actually be represented by:

$$Q = a|00> + b|01> + c|10> + d|11>$$

Quantum mechanics once again says that the probability of measuring |00> (both qubits 0) is $|a|^2$ and of |01> is $|b|^2$ and so forth. Quantum "gates" act on qubits. One of the fundamental quantum gates is the CNOT gate, where:

$$CNOT(Q) = a|00> + b|01> + c|11> + d|10>$$

Comparing the weights, it can be seen that CNOT swaps around the c and d coefficients. To understand this more clearly, look at the truth table in Table 1 below for a classical CNOT gate. A classical CNOT gate is similar to an exclusive-or gate (XOR) but unlike the XOR is reversible. This reversibility is achieved by having two outputs (which can be used to reconstruct the input) and is key to all quantum gates, the reasons for which are outside the scope of this article. A CNOT can also be viewed as the A input controlling the B output: if the A input is one is switches on a NOT gate acting on B, otherwise the B signal just passes through unchanged.

In essence all rows of the truth table are acting simultaneously in the quantum version, with the first digit in each ket is A and the second digit in each key is B. It is the CNOT gate that allows two qubits to be entangled. The concept of entanglement is beyond the scope of this article, but has been discussed in detail in relation to sonification and computation (Kirke and Miranda 2017).

In A	In B	Out A	Out B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0

Table 1. *Classical CNOT truth table*

A convenient way of writing qubits and gates is in vector / matrix notation. This will simplify our discussion moving fowards, as the key elements that matter in the gate processing are how the coefficients a, b, c, etc of the qubits change. q = a|0> + b|1> is written as the vector:

$$\begin{bmatrix} a \\ b \end{bmatrix}$$

and Q = a|00> + b|01> + c|10> + d|11> as the vector:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Then the CNOT gate can be written as:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

for the following reason. By standard matrix multiplication, if CNOT multiples Q, the result gives the same weightings as the CNOT definition from earlier. The coefficients c and d are swapped around:

$$CNOT(Q) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} a \\ b \\ d \\ c \end{bmatrix}$$

Another fundamental gate is the Hadamard gate. Unlike CNOT – it has no classical equivalent because it can result in qubits which have two values simultaneously. In matrix form it is:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

It can be thought of as a gate that transforms a single qubit into a superposition of qubits because:

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0\\ 1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ -1\end{bmatrix} = \frac{1}{\sqrt{2}}\left(\begin{bmatrix}1\\ 0\end{bmatrix} - \begin{bmatrix}0\\ 1\end{bmatrix}\right) = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

This reason for the sqrt(2) is to ensure that putting a qubit through a Hadamard gate and then another Hadamard gate will output the original qubit. This is useful and important for further development. As an aside, it is interesting to note that if another quantum gate, the Rotational R gate, is added to the set of the CNOT and the Hadamard, then the three make a universal quantum gate set. In the same way that NAND gates can be used to build any classical function, these three gates can be used to build any quantum function. An R gate has an exponential term as one of its matrix entries. This R-gate is not used in GATEMEL, the algorithm introduced here. However the Hadamard and the CNOT are. However there is one final gate that needs to be added to create GATEMEL. It is called the Pauli X gate:

$$X = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$

It is the quantum equivalent of a NOT gate because:

$$X(|0\rangle) = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix} = \begin{bmatrix}0\\ 1\end{bmatrix} = |1\rangle$$

$$X(|1\rangle) = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}0\\ 1\end{bmatrix} = \begin{bmatrix}1\\ 0\end{bmatrix} = |0\rangle$$

$$X(a|0\rangle + b|1\rangle) = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}a\\ b\end{bmatrix} = \begin{bmatrix}b\\ a\end{bmatrix} = b|0\rangle + a|1\rangle$$

The commands for these gates in the IBMQASM language are x (NOT), cx (CNOT) and h (Hadamard), and qubits in IBMQASM are referenced as q[0], q[1], etc. for qubits 0 and 2 up to 4. Another key command is "measure", which returns the result of physically measuring a qubit. All qubit inputs are set to |0> by default. Thus the IBMQASM:

measure q[0]
will return the value 0 from q[0] with high probability. In 1024 runs of this code on the hardware QC it returned 0 with 0.976 probability, and 1 with 0.024 probability.

The program:

```
x q[0]
measure q[0]
```

will return value 1 with high probability, since the input is always 0 and the x command is the bit-flipping Pauli X-gate. In 1024 runs of this code on the hardware QC it returned 1 with 0.962 probability, and 0 with 0.038 probability. In this case, the non-pure probabilities are a result of an imperfect quantum computer hardware implementation, due to what is known as decoherence. However, this implementation is state-of-the-art as of this writing.

Finally:

```
cx q[1], q[2]
measure q[1]
measure q[2]
```

will return the result of a CNOT of q[1] on q[2]. Since the input state is |00> to start, the measured output will be |00> with a high degree of probability. One run of 1024 examples lead to 00 with probability 0.979, 01 with probability 0.005, 10 with probability 0.007, and 11 with probability 0.009.

Quantum Annealing

As of the time of writing this article, there is only one company making quantum computers available for purchase. (Though IBM has made a commercial gate-based machine available on a timeshare basis.) These computers are based on adiabatic quantum computing (Albash et al. 2015). An adiabatic quantum computer implements a form of computation reminiscent of connectionist computing: what is known as an Ising model (Lucas 2014). Ising models were originally used to describe the physics of a magnetic material based on the molecules within it. As well as electrical charge, each of these molecules has a property known as spin; their spin can be +1 or -1. An adiabatic quantum computer attempts to find spin values to minimize the total energy. The user sets the values of the connections between the simulated molecules so as to define the problem to be solved. Such a minimizer can be implemented using non-quantum hardware. However significant speedups are expected through the use of quantum hardware. Such hardware is now being sold by the Canadian company D-Wave.

On the face of it, it may not seem significant that quantum computers can be built to solve only this problem type. However over a period of 28 years, more than 10,000 publications came out in areas as wide as zoology and artificial intelligence on applications of the Ising model (Bian et al. 2010). Any problem that can be modeled using elements interacting pairwise with each other, and involves minimizing some measure of the interaction, has the potential for being modeled as an Ising problem.

There is an ongoing debate about how the D-Wave adiabatic computer truly functions and what speedup it can provide; but results indicated quantum effects occurring in subgroups of nodes in the computer and results by Google have claimed large speed increases for quantum hardware. As has been mentioned, this is thought by some to be due to quantum tunnelling (Katzgraber 2015). When searching for low energy states, a quantum system can tunnel into nearby states.

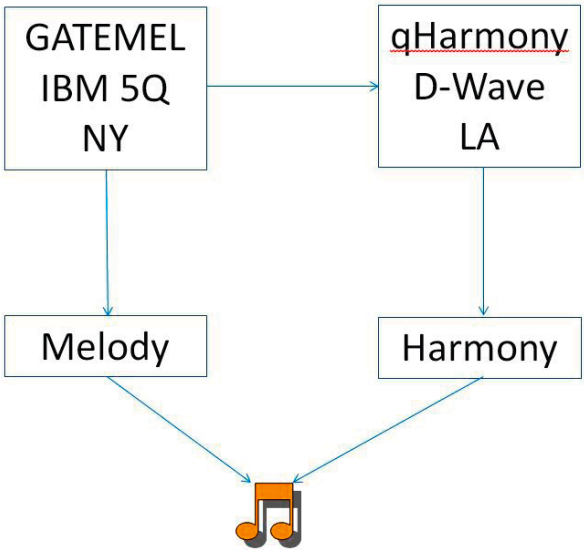


Figure 1. *qGen architecture*

qGEN

qGEN is a hybrid quantum algorithm using both hardware gate-based and adiabatic quantum computers. The gate-based algorithm is GATEMEL – the simplest possible gate-base quantum music algorithm. It has been used to generate simple melodies for media demonstration purposes.[3] The adiabatic algorithm is qHarmony which, given a note, attempts to harmonize it (Kirke and Miranda 2017).

qGen on D-Wave – qHarmony

A basic harmony tool called qHarmony has been developed on a D-Wave 2X. It generates options for a set of white piano notes that can be constructed as a "reasonably"

3 Alexis Kirke, "Futureproofing." *BBC Radio 4,* https://soundcloud.com/alexiskirke/alexis-kirke-talks-quantum-on-futureproofing-bbc-radio-4

assonant chord, and which can harmonize a user-provided white piano note. The problem is approached by mapping the notes of the scale of C Major to qubits. The qubits connections in the D-Wave are designed so that qubits representing notes that are closer together on the keyboard, contribute to a higher energy than qubits representing notes that are further away from each other on the keyboard qHarmony is described in detail in (Kirke and Miranda 2017).

qGen on IBM 5Q – qHarmony

When thinking about the simplest quantum computer music algorithm, it is useful to imagine a logic electronics engineer designing a melody generator back at the dawn of classical computers. For the simplest classical computation based melody algorithm, we can look back into computer music history where systems actually use pseudo-random number generators to "create music", for example. A pseudo-random number generator is in effect a function of classical logical gates, memory and a timer. Anything simpler will only produce the same notes or the same pattern repeatedly. Constraints might be added, for example, by saying that tunes can't have too long a rest, or too long a run of notes without a rest. This requires a more complex set of logic gates. But it is expressible.

When working with a small number of bits, it is simplest to encode relative up and down movement, rather than use multiple bits to encode larger numeric note values. To allow more interesting note movements, a two bit register can be used where the first bit is up or down, and the second bit is the size of the jump: 1 or 2 pitch degrees. Thus 00 would be down 1 degree, 01 down 2 degrees, 10 up 1 degree and 11 up two degrees. As already mentioned, melodies also have rests, there are not notes every metronome beat. So the system can have another bit to indicate play or don't play a note.

The up or down and play note flags in classical computing would be based on a pseudo-random number generator. To repeat the above simplistic compositional constraint, it will be required that if the melody note played for the last two metronome beats, then there should be a rest for the next beat; whereas if the melody rested for the last two beats, it must play for the next beat. In classical computation the equations could be written as below. The first 3 are random number generators, the last three are the play note constraint:

Play_note_flag = PSEUDORANDOM_BIT
Pitch_direction = PSEUDORANDOM_BIT
Pitch_size = PSEUDORANDOM_BIT
Last_two_play_note_flags_equal = NOT(XOR(prev_play_note_flag, prev_prev_play_note_flag))
Current_and last_play_note_flags_equal = NOT(XOR(play_note_flag, prev_play_note_flag))
Play_note_flag = CNOT(Play_note_flag, AND(Last_two_play_note_flags_equal, Current_and last_play_note_flags_equal))

So the last line flips the play_note_flag bit if it is the same as the last two flag values. Looking now at a quantum version, the non-deterministic element is given, without the need for pseudo-random number generation. It is truly random with no underlying hidden process. What is more complex is implementing constraints and memory. In GATEMEL the memory is implemented outside of the quantum computer. The gates can be implemented using those already introduced. To convert to a quantum system, the following needs to be observed. A NOT gate can be implemented using a Pauli X gate. An XOR and a CNOT gate are equivalent. And that randomness can be generated by creating a superposition of any input using a Hadamard and then observing it.

So the written version of GATEMEL has two input qubits labelled *prev_play_note_flag* and *prev_prev_play_note_flag*, and three output qubits labeled *pitch_change_direction*, *pitch_change_size* and *play_note_flag*. Note that the function CNOT(a, AND(b,c)) is called a Toffoli gate. The equations are written in words as:

Pitch_change_direction = Hadamard(|0>)
Pitch_change_size = Hadamard(|0>)
Note_play_flag = Hadamard(|0>)
Note_play_flag = Toffoli(Note_play_flat, X(CNOT(prev_play_note_flag, prev_prev_play_note_flag)), X(CNOT(play_note_flag, prev_play_note_flag)))

The three hadamard statements are essentially random number generators. prev_play_flag is the whether the previous pitch was played or not, 1 for player, 0 for not. prev_prev_play_flag is whether the note before that was played. Thus these equations have the effect that if the last two play note instructions were the same (both 0 or both 1) then the current note play flag is set to the opposite.

Simplifying these equations to make them more IBMQASM, and assuming the previous two play_note flags have been input on q[2] and q[3] we have:

q[0] = H(q[0])
q[1] = H(q[1])
q[4] = H(q[4])
q[4] = Toffoli(q[4], X(CNOT(q[3],q[2])), X(CNOT(q[3],q[4])))

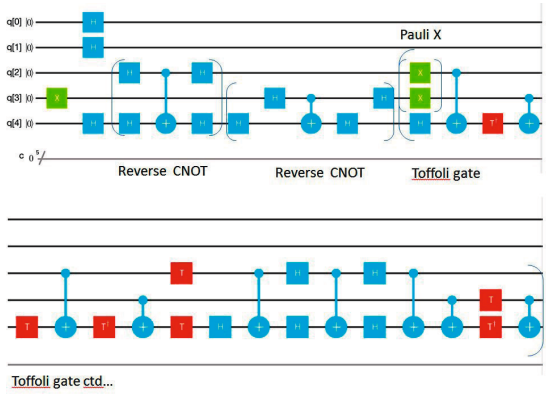
Making this circuit useable in hardware IBMQASM requires a number of adjustments. In particular, the topology means that CNOTs can only be performed of the form cx q[i], q[4]. In other words all CNOTs must have q[4] as their second parameter. In fact, CNOT(|q[i]q[j]>) can be calculated as a function of CNOT(|q[j]q[i]>) using what is known as a change of basis. In essence the whole input is rotated around a complex vector space, CNOTed, and then rotated back. The Hadamard gate can be used to transform the state |xy> so that when it is put through a CNOT gate, and then Hadamard transformed again, it behaves as though it were|yx> in the CNOT gate. Hence the final hardware IBMQASM code for GATEMEL is in Table 2. It includes the Toffoli gate build, and the various change of bases:

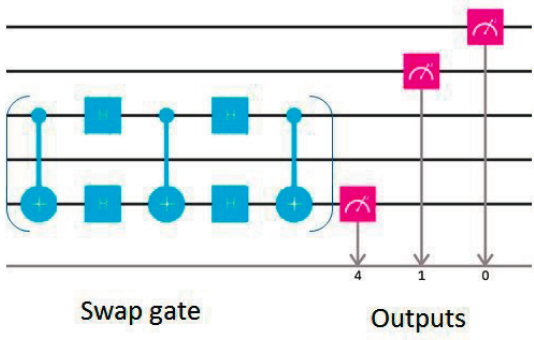
```
include"qelib1.inc";
qreg q[5];
creg c[5];
//move up or down?
//q[0] = 0 means move down
//q[0] = 1 means move  up
//q[1] = 0 move one degree
//q[1] = 1 move two degrees
//randomly select q[0]q[1]
h q[0];
h q[1];
//q[4] = 1 means play note
//q[4] = 0 means don't
//select q[4] randomly
h q[4];
//check if q[2] = q[4]
h q[2];
h q[4];
cx q[2],q[4];
h q[2];
h q[4];
//check if q[3] = q[4]
h q[4];
h q[3];
cx q[3],q[4];
h q[4];
h q[3];
//check rest or play flag not
the same for last two
notes (Toffoli/AND gate)
x q[2];
x q[3];h q[4];
cx q[2],q[4];
tdg q[4];
cx q[3],q[4];
t q[4];
cx q[2],q[4];
tdg q[4];
cx q[3],q[4];
t q[2];
t q[4];
h q[4];
cx q[2],q[4];
h q[2];
h q[4];
cx q[2],q[4];
h q[2];
h q[4];
cx q[2],q[4];
cx q[3],q[4];
t q[3];
tdg q[4];
cx q[3],q[4];
//end of Toffoli/AND
//switch the "move" flag
//back to q[4] output
//for consistency
cx q[2],q[4];
h q[2];
h q[4];
cx q[2],q[4];
h q[2];
h q[4];
cx q[2],q[4];
//collapse the quantum
//bits to classical bits
measure q[4] -> c[4];
measure q[1] -> c[1];
measure q[0] -> c[0];

//output q[0]q[1] is two bit number
defining move up or down
//output q[4] is whether to play a note
this beat or not
```

Table 2. *The final hardware IBMQASM code for GATEMEL*

Comments have been added just for readability here, but they are not included in IBMQASM1.1. The gate diagram version is shown in Figure 2.

Figure 2. *Gate diagram for GATEMEL*





qGEN Example

The D-Wave 2X processor will now be used to generate a harmony for a simple note sequence generated by the ibmqx2. qHarmony is called at the start of each bar of a GATEMEL tune. Figure 3 shows an example output. An audio version can be heard on Soundcloud.[4] The GATEMEL start note is middle C, and each beat is an eighth note. So if GATEMEL says not to play a note for 2 beats after a note has been played, then that last note played will go on for a dotted quarter. If GATEMEL says not to play a note for 1 beat after a note has been played, then that last note will go on for a quarter. Note that the half note in bar 1 of Figure 3 highlights the probabilistic nature of quantum computation. For a half-note to appear means that a new note is not being triggered for 3 beats in a row. This should be very low probability, as recall the circuit is designed to stop this happening, but it does occur this once.

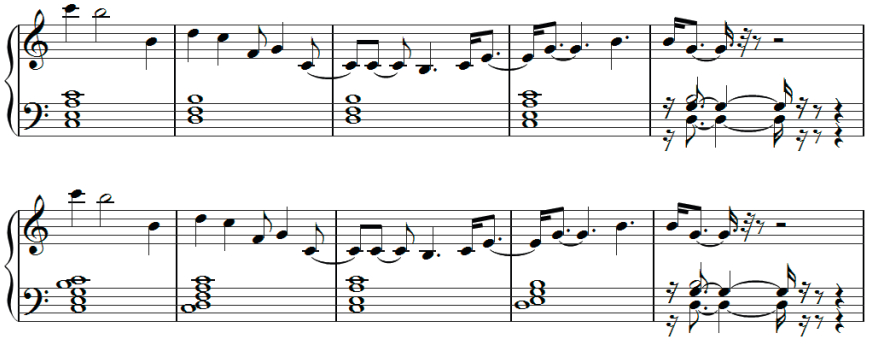


Figure 3. *Example qGen output*

4 https://soundcloud.com/alexiskirke/quantummelharm2

The quality of music produced by qGen is not particularly high. There are two main reasons for this. The first is that only a 5 qubit and an 8 qubit system are used - to simplfy this introduction to quantum programming (Kirke and Miranda 2017). For example - this limits the harmonies from the D-Wave to the 8 white notes. Such a constraint is rare in most mainstream composition. However, using more than 8 qubits on a D-Wave would have required much concentration on qubit connectivity issues in the D-Wave, rather than the quantum-related issues. The connectivity of the D-Wave 2X outside of 8 qubit segments is fairly complex.

The second reason – once again used for simplification purposes in an introductory paper – is that qGen takes no account of its previous harmonies and melodies when generating its next ones. For example if a composer uses an Am/C chord to harmonize a melody segment, then that choice of chord will affect the next chord. Not so in qGen.

qGen only takes advantage of one aspect of QC: its non-deterministic nature and ability to return multiple results. However the quantum part of the algorithm is so simple that it does not require the potential speed-ups available from quantum computers. The D-Wave 2X has over 1000 qubits available, and enters states of superposition and entanglement during its calculations. Even the simple 8 qubit algorithm above will have utilized these quantum states in coming to the results. In fact, despite the debates about how quantum effects occur in the D-Wave, it has been shown that entanglement does occur *at least* within the 8 qubit groups.

However a much more complex and constrained problem would be required to utilize all advantages of QC. Constraint-based and spectral composition, together with musical/sonic pattern matching algorithms are areas which may benefit from QC, due to their potential computational complexity. In essence, any complex musical problem that involves a database search, or can be fully or partially modelled as an Ising system, could benefit from quantum computation.

List of References

Albash, Tameem, Vinci, Walter, Mishra, Anurag, Warburton, Paul A. and Lidar, Daniel A. (2015) "Consistency tests of classical and quantum models for a quantum annealer." *Physical Review* A 91(4): 042314.

Anon. (2014) "Scientists 'sonify' LHC data to Chamber Music." *ALICE Matters — A Large Ion Collider Experiment*, 30 October 2014, http://alicematters.web.cern.ch/?q=content/node/776

Bian, Zhengbing, Chudak, Fabian, Macready, Willliam G. and Rose, Geordie (2010) "The Ising model: teaching an old problem new tricks." *D-Wave Systems,* 30 August 2010, https://www.dwavesys.com/sites/default/files/weightedmaxsat_v2.pdf

Brody, James (2003) "Background Count", for percussion and 2 channel electroacoustic. *Background Count. Electroacoustic Music by James Brody.* CD Innova 600116680624, https://www.innova.mu/albums/james-brody/background-count

Cádiz, Rodrigo F. and Ramos, Javier (2014) "Sound Synthesis of a Gaussian Quantum Particle in an Infinite Square Well." *Computer Music Journal* 38(4): 53–67.

Coleman, Jaz (2003) *Music of the Quantum*. New York: Columbia University, http://musicofthequantum.rutgers.edu/musicofthequantum.php

Culpan, Daniel (2015) "CERN's 'Cosmic Piano' uses particle data to make music." *Wired*, 8 September 2015, http://www.wired.co.uk/article/cern-cosmic-piano

Glowacki, David, Tew, Philip, Mitchell, Thomas and McIntosh-Smith, Simon (2012) "Danceroom Spectroscopy: Interactive quantum molecular dynamics accelerated on GPU architectures using OpenCL." *The fourth UK Many-Core developer conference (UKMAC 2012),* Bristol, http://eprints.uwe.ac.uk/18268/

Katzgraber, Helmut G. (2015) "Seeking Quantum Speedup Through Spin Glasses: Evidence of Tunneling?" *American Physical Society Meeting Abstracts* id. L53.005.

Kirke, Alexis, Miranda, Eduardo, Chiaramonte, Antonino, Troisi, Anna R., Matthias, John, Fry, Nicholas and McCabe, Catherine (2013) "Cloud Chamber: A Performance with Real Time Two-Way Interaction Between Subatomic Particles and Violinist." *Leonardo Journal* 46(1): 84–85.

Kirke, Alexis (2016) *Superposition Symphony. Port Eliot Festival*, 29 July 2016, https://porteliotfestival.com/wp-content/uploads/2016/07/Port-Eliot-Programme.pdf

Kirke, Alexis and Miranda, Eduardo R. (2017) "Experiments in Sound and Music Quantum Computing." In: Eduardo Reck Miranda (ed.), *Guide to Unconventional Computing for Music.* Cham: Springer, 121–157.

Lucas, Andrew (2014) "Ising formulations of many NP problems." *arXiv.org*, Cornell University Library, preprint https://arxiv.org/pdf/1302.5843.pdf

Malloy, Ian and Hollenbeck, Dennis (2016) "Inversions of New Hope." *arXiv.org*, Cornell University Library, preprint https://arxiv.org/pdf/1608.04993.pdf

Neven, Hartmut (2016) "Quantum Annealing at Google: Recent Learnings and Next Steps." *American Physical Society (APS) Meeting Abstracts,* March 2016, http://adsabs.harvard.edu/abs/2016APS..MARF45001N.

O' Flaherty, Eric (2009) "LHCsound: Sonification of the ATLAS data output." *Science & Technology Facilities Council — Small Awards Scheme*, https://stfc.ukri.org/news/the-sounds-of-the-lhc/

Putz, Volkmar and Svozil, Karl (2017) "Quantum Music." *Soft Computing* 21(6): 1467–1471.

Shor, Peter W. (2006) "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer." *SIAM Journal of Computing* 26(5): 1484–1509.

Sturm, Bob L. (2000) "Sonification of Particle Systems via de Broglie's Hypothesis." In: Peter R. Cook (ed.), *Proceedings of the 6th International Conference on Auditory Display (ICAD2000), Atlanta, GA, April 2–5, 2000*. Atlanta: Georgia Institute of Technology / International Community for Auditory Display, https://smartech.gatech.edu/bitstream/handle/1853/50683/Sturm2000.pdf?sequence=1&isAllowed=y

Sturm, Bob L. (2001) "Composing for an Ensemble of Atoms: The Metamorphosis of Scientific Experiment into Music." *Organised Sound* 6(2): 131–145.

Weimer, Heindrik (2010) "Listen to Quantum Computer Music." *Quantenblog*, http://www.quantenblog.net/physics/quantum-computer-music.

Алексис Кирк

Програмирање квантних рачунара базираних на употреби логичких кола за потребе рада са музиком

(Сажетак)

Досад су забележени значајни покушаји да се једначине квантне механике користе за генерисање звука и да се озвуче симулирани квантни процеси. Али, за нове облике рачунања који би се користили у компјутерској музици, мора се употребити одговарајући хардвер. Ово се досад ретко дешавало са квантном компјутерском музиком, најпре зато што такав хардвер није широко доступан. Други разлог јесте околност да овакав хардвер захтева извесно познавање теорије квантног рачунарства. Овим чланком померамо овај процес унапред помоћу два хардверска квантна рачунарска система: IBMQASM v1.1 и D-Wave 2X. Такође уводимо неке идеје из IBM-овог система заснованог на логичким колима, на начин доступан рачунарски писменим читаоцима. Ово је презентација првог хибридног квантног компјутерског алгоритма, који укључује две хардверске машине. Иако ниједан од ових алгоритама експлицитно не користи обећана квантна убрзања, они представљају виталан први корак у увођењу квантног рачунарства у поље музике.

Чланак започињемо кратким прегледом квантног рачунарства и указујемо како се оно може применити на подручју уметности. Следи истраживање претходних пројеката у којима су коришћени стварни или симулирани квантни процеси у музичким делима или извођењима. У следећем одељку се говори о најпознатијој врсти квантних рачунара, заснованих на логичким колима, и описује се хардвер једног од мањих квантних рачунара компаније IBM. Следи кратак увод у теорију квантног рачунарства; ове идеје су потом пројектоване на језик који користе IBM рачунари: IBMQASM.

Следећи одељак доноси кратак преглед друге врсте квантног рачунара који се користи: D-Wave. Детаљнији описи мог алгоритма доступни су у другим чланцима на које се позивам. На крају је описан qGen: IBM генерише мелодију, а D-Wave је хармонизује. Фокус је на мелодијском алгоритму, пошто је алгоритам D-Wave описан у поглављу из књиге на коју референрам. Развијен је "најједноставнији могући" мелодијски алгоритам, уз који је приложен и одговарајући пример.

Кључне речи: квантна компјутерска музика, алгоритми, D-Wave