Simulation and validation of qubits based on Phase estimation algorithm

Elanchezhiyan T*, Dheerendra Singh Tomar**

* VIT University, Vellore, Tamil Nadu, India
** Intel Corporation, Bengaluru, Karnataka, India

DOI: 10.29322/IJSRP.12.01.2022.p12149
http://dx.doi.org/10.29322/IJSRP.12.01.2022.p12149

Abstract- After the advent of Quantum Physics, new opportunities were created for research and development in the world of science and technology. Quantum computation is one such emerging technology which allows for notable speed-ups compared to conventional computational methods. Here a quantum bit simulator is presented. This simulator is an engineering work, and no deep understanding of quantum mechanics is required from the user. The well-known phase estimation algorithm is presented using this simulator. The qubit simulator will be a useful tool for the study of quantum computer circuits, quantum computing, and the development of new quantum algorithms

Index Terms- Quantum algorithms, Quantum computing, Quantum entanglement, Quantum superposition, Phase estimation, Validation, Qubits, Python.

I. INTRODUCTION

Though the present day's supercomputers offer incredible computational power, still there exists few problems which cannot be tackled by classical computing. With these limitations the apt solution to the computational challenges which might arise in the future remains undiscovered. Quantum computation which is a combination of physics, engineering and computer science has become a promising technology which has been found to be multiple times faster than the present-day classical computers. The basis of the quantum computing arises from the quantum physics concepts namely the quantum superposition, and the quantum entanglement. Generally, a computer can be defined as a device which accepts data as an input, perform some operations with the data and gives out the result. The quantum computers also nearly do the same, but they perform some operations on the input data with a process which depends on quantum physics concepts as the basis. But we cannot the use the expeditious quantum computers for performing general purpose tasks as they come with little bit of hard limitations on the environments which quantum computers require to operate seamlessly. Quantum computers demands a near absolute zero temperature environment which can be provided only by cryogenic fridges as in Fig.1, because of this limitation quantum computers in the present-day scenario are available only at the datacenters and laboratories. Even after the commercialization of quantum computers they would be most likely accessed through cloud. Classical computing will be still around and will be used for all general-purpose applications. In the present-day scenario, the quantum computers which are commercially available have only less than 100 qubits which is very much smaller compared to the today's Terabytes and gigabytes used in the classical computers. Problems which are having small input and output data sizes but having huge number of computations will most likely utilize the quantum advantage. Before we jump into qubits - the quantum analogue of a classical bit we need to understand about bits which are the smallest units of data in a classical computer. Using classical bits, we cannot represent few real-life scenarios like chemical reactions, genetics etc. For these types of scenarios, the quantum advantage will be useful. A classical bit is a physical system that has two distinct states namely 0 and 1. When we measure the classical bit, we copy the information contained in the state of that bit. A qubit is the smallest unit of data in a quantum computer which can be physically implemented on systems having two states. They are like the classical bits, but the difference is they behave according to the laws of quantum physics which allows them to perform few operations which classical bits is not capable of. The qubits have a special characteristic that it can be at a superposition state which

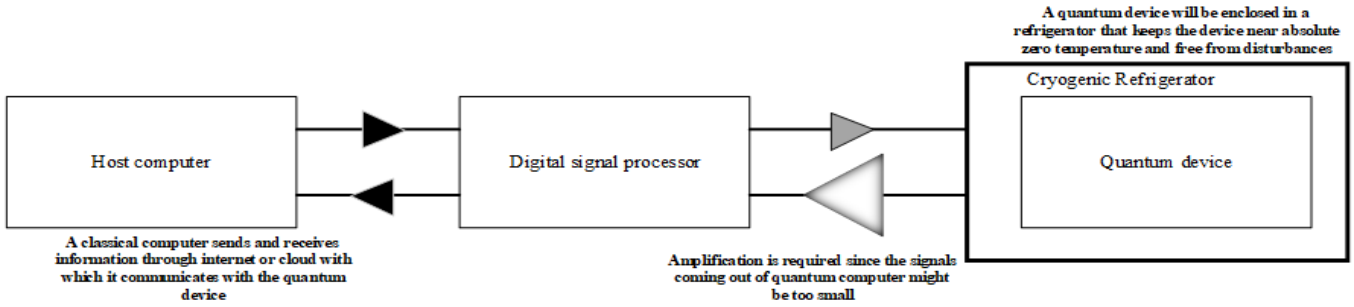

Fig 1: Quantum computer operation

is neither zero nor one. But it is important to note that a qubit can be in any indeterministic state but while we measure a qubit, we always end up getting a deterministic result. And if a group of qubits are entangled state of a qubit can affect the state of the other qubit. This property of qubit can be better understood by taking a quick look at the Schrodinger's cat experiment. Measuring a qubit's state will surely have an impact on the state of the qubit. These qubits can be simulated using the classical computing resources to understand them better. But there are lot of barriers which limits the qubit simulation. Even with the maximum usage of classical computing resources available we can only simulate up to 40 qubits. But the qubits have an interesting property called the reversibility which turns out to be a fundamental for quantum computing.[1] This property states that if we rotate a quantum state, we can get back to the same state by rotating it backward. We already know that the classical NOT operation is reversible, but it must be noted that classical AND and classical OR are irreversible. In contrary all the quantum operation is reversible. This reversible nature of the quantum operations helps us to reuse the same qubits for multiple operations. It is also important to note that only the quantum operations are reversible but not the quantum initialization and quantum measurement. Due to their high-speed efficient computing, quantum computers can create a revolution in various fields like cryptography, genetics, etc.

II. QUANTUM ALGORITHMS

Before we start programming our quantum devices, we need define our problems at a high level in a more generic manner. An algorithm may be defined as the step-by-step procedure written mostly in high level to solve a given problem. The algorithm will be independent of the programming language and the operating system used. We use quantum simulators to test our quantum programs and algorithms which is analogous to the simulators we use to test specialized hardware like FPGA. The only difference is we can only use a classical computer to simulate a quantum computer that too for only few restricted kinds of programs due to the sparse availability of qubits. The quantum algorithms will help us to organize the steps which need to happen in classical and quantum devices. The quantum algorithms will be using the quantum effects such as superposition, quantum entanglement. The quantum algorithm will be implemented by a quantum program which consists of a classical program that send instructions to a quantum device to initialize or measure a particular state. We cannot copy the state of a qubit like we do with classical bits, so the idea is to prepare those qubits using the quantum program. This preparation steps can be copied as much time as required to get the required qubit state. Some of the quantum algorithms are explained in brief to have a better understanding. The Deutsch-Jozsa algorithm [1] is a quantum algorithm which requires lesser resources to solve a given computational problem than any classical algorithm in the present-day world. This algorithm is a deterministic quantum algorithm which uses a single query to arrive at the results using a unitary matrix called oracle. The algorithm determines whether a hidden function is constant or balanced in a quick manner. A function is said to be constant when the qubit provides an output of 0 or 1 all the time irrespective of the inputs. On the other hand, a function is said to be balanced when the qubit provides an output of 0 and 1 with equal probability. Classically, to determine if a function is constant or balanced, we must learn the entire function by building up its truth table. But with quantum algorithms we can determine this by measuring one qubit. Since quantum algorithms use the concepts of superposition and entanglement which allows the system to calculate all probabilities at once this is made possible. A quantum computer allows us only to measure individual qubits. To measure more complex observables like the energy described by a Hamiltonian, we look after the next quantum algorithm-phase estimation algorithm. This algorithm is mostly used within other quantum algorithms such as Shor's algorithm to build up larger and more complex programs. The phase estimation algorithm is used to find the phase of an eigenvalue corresponding to the eigenvector of a unitary operator. Here a large list of input values is taken, and they are scaled with the secret angle and probability of getting a one for each scale factor is measured. Using these data, the angle with which the qubit has been rotated can be found. Extending the idea, we can also use quantum algorithms to solve chemistry problems. Many chemists would love to understand how the molecule arranges itself as a function of energy based on simulations than performing experiments. This can be done with the help of a simulator within a simulator – a classical computer will be used to simulate a quantum computer which is again used to simulate a quantum chemical system. When we think quantum mechanics in terms of electrons, nuclei and other things, the quantum mechanics heavily relies on the concept of energy. The energy can be represented using a special type of matrix called Hamiltonian matrix.[2] To simulate a chemical molecule, the molecule is first described in terms of Hamiltonian matrix, then the operators are constructed. With these operators we can simulate the Hamiltonians in our quantum device. The operations are decomposed by representing them using tensor product of Pauli matrices and to compose everything again we use Trotter-Suzuki method.[1] Finally phase estimation algorithm can be used to estimate an eigenstate's energy. The energy estimates help us to understand much in detail about chemical interactions. Shor's algorithm is a combination of classical and quantum algorithm which combines the pre and post processing required with phase estimation on a quantum computer to aid in quick factorization of integers using modular arithmetic. Shor's algorithm is a best example to understand that quantum hardware and algorithms fit well as subroutines for combined quantum-classical algorithms. To factor an integer N, Shor's algorithm uses phase estimation and a quantum computer to find the period of a function that takes powers of another integer a using modular arithmetic mod N. After some classical postprocessing, that period can be used to find the factors of N. Since Shor's algorithm can break through the present-day security encryptions, it is not true that the cryptography is doomed. There are technologies in both classical and quantum computing to resist algorithms like Shor's algorithm.

III. SIMULATORS

Computer simulation is performed on the computer by designing mathematical models which predict the outcomes of a physical system. They have become a useful tool since it's easier to understand and analyze many physical systems using simulators and mistakes can be rectified without much damage on the real hardware. Computer simulation developed rapidly right from the

days of World War II and it has its impact over many fields in the current day scenario. Computer simulations are widely used in medicine, aeronautics, aviation, automotive, software etc. For instance, flight simulators help the pilot trainees to get used to the real-life scenarios. This helps them to perform better with real flight machines using all their experience learnt using simulators. The input sources to a simulator can vary widely depending upon the nature of the simulator. Simulators allow us to analyze systems before they are constructed, reduce the number of design mistakes, optimize the design, analyze operational systems, Create virtual environments for training, entertainment etc. We cannot observe an operational system at different abstraction levels, but this is made possible with the help of computer simulations. This is really a big plus while dealing with very large-scale integrated circuits as the higher level of abstraction can reduce the complexity and helps us to understand better. Using simulations, an individual will be able to handle large amounts of data seamlessly comparing to the conventional physical systems. Unexpected behaviors and phenomenon can be handled at ease with the use of computer simulations as it gives us a prior alarm. We can also come up with best conditions in order to achieve best results by making a analysis after multiple simulations. Another big advantage with computer simulation is that it guarantees correct and precise results all the time so that it's not necessary to repeat the simulation to check whether we got the correct results. Computer simulations are classified in a bipolar manner based on their nature - stochastic or deterministic, state - steady state or dynamic, time - continuous or discrete and execution - local or distributed. To enjoy all these advantages provided by the simulators they need to be designed in a precise manner. The typical cost of a simulation study is substantially less than 1% of the total amount being expended for the implementation of a design or redesign. Since the cost of a change or modification to a system after installation is so great, simulation is a wise investment. There are lot of challenges which needs to be tackled in building a simulator. A designer needs to make sure that he has understood the physical system completely and has required resources before he starts building a simulator. After the simulator is built, its efficiency and reliability need to be checked before we perform our first simulation. There are lot of simulation programming languages available which aid us to work with the simulators. Simulations have already registered their importance in the field of quantum computing too. We will use the concept of simulation to enter a deep level understanding of qubits.

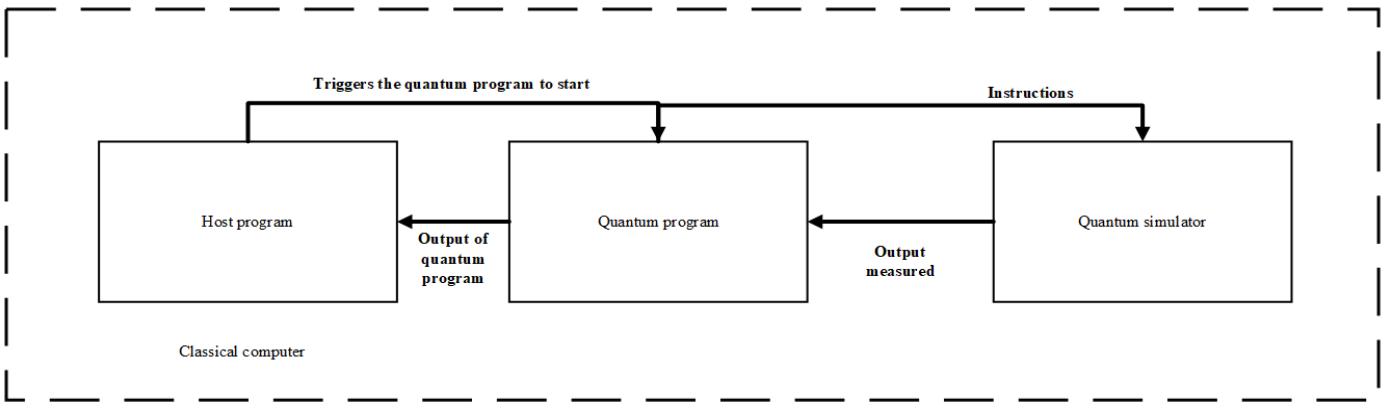

Fig 2: Implementation of qubit simulator

IV. VALIDATION

Validation refers to a process of testing a regulated computerized system to make sure that it works exactly what it is designed to do in a consistent and safe manner. With respect to the system on chip designs, validation involves assuring their functional correctness, robustness, meeting the power and timing constraints etc. It is very interesting to note that nearly 70% of the overall time, efforts and resources are spent in SoC validation and verification and hence it has been acknowledged as a major bottleneck in SoC design. SoC design validation has been broadly divided into three categories: pre-silicon validation, post-silicon validation and in-field debugging. Pre-silicon validation refers to the verification done before the design is sent for the fabrication. Post-silicon validation refers to the validation efforts carried out on the manufactured chips in the real application environment before proceeding with mass production. Post-silicon validation is responsible for detecting various design flaws which includes the bug escaped from pre-silicon validation by using various tests and debug frameworks. Since the post-silicon validation involves a manufactured chip the controllability and observability of the resources are limited which posts a huge challenge to the validation engineers. Post-silicon validation is the final frontier to check the functional correctness and integrity of the design before mass production. So, the most important duty of post silicon validation engineer is to make sure that no bugs escape to the customers. The silicon observability can be increased by using extra memory units which store the values of few selected registers during the runtime called the trace buffers. But the depth and width of these trace buffers are small which limits the number of registers that can be traced. This emphasizes on the fact that trace-based validation and debug quality depends upon the quality of the selected signals. Here, a simulation-based approach can be used to select the signals as they are more precise, but they are extremely slow and are not suitable for complex designs. Simulation -based design environments have a nearly perfect observability however the limited amount of data they can generate during the post silicon validation is a problem. And in few cases when a defect occurs,

it's very hard to reproduce the data on simulator. The first and foremost step in post-silicon validation is the test plan generation which includes the test architecture, debug software, functionality requirements, corner cases etc. It is followed by building a right debug infrastructure, generation of tests and post silicon debug which involves localization of the error. It is important to understand that post-silicon validation is usually performed under tight schedule with respect to time-to-market constraints and a significant step in product development cycle.

V. IMPLEMENTATION

Imagine quantum computers being used by everyone as handheld devices, it feels astonishing right - but unfortunately this is not possible with the current day technologies available. Quantum devices are built with components which require surroundings with a near zero temperature. Even a little bit of noise and temperature rise can disturb the quantum states and lead to unpredictable behaviors. To avoid this scenario, quantum devices are always maintained in an environment which maintains required temperature and resists all kinds of noise and disturbances. This is made possible with the use of cryogenic refrigerators which enclose the quantum computer. Due to these constraints, quantum computers cannot be portable or used like classical computers in present day scenario. So how to access these complex machines? Luckily, we have got internet and cloud services which aid us to access quantum devices from every nook and corner of the world by connecting with them. This makes quantum devices like any specialized extra hardware attached to our personal computer which can be used for specific operations. Thus, a quantum computer will be accessed through internet by our classical machines for specialized applications. A host program written in python sends a quantum program to the target quantum device through cloud which connects with the quantum hardware through instructions and measures the results. The quantum program will be written in a manner such that at the end of the program the result measured in sent back to the classical computer through the cloud. This is how a quantum computer works in real time - but we are going to implement a quantum bit simulator. For our task we should be able to finish everything within the classical computer itself. We will implement a simulator which works on the same classical computer where the host program runs but still, we use our quantum program to send instructions to the quantum bit simulator and measure the results back as in Fig.2. These simulators use linear algebra such as matrix multiplication heavily as the qubits are represented using Dirac notation and quantum operators are represented using unitary matrices. But a challenge remains before us as we start building our quantum simulator - as the number of qubits grow, the number of states a register can be in grows exponentially. With our laptops and personal computers, we can simulate up to 30 qubits seamlessly and the good thing is that will be enough to understand most of the quantum operations. The python or Q# programs we write on our classical computer is sent to an interface for a quantum device. This interface defines what instructions will be available for a quantum device. For example, based on the program sent the interface might get exposed with instructions like "Hadamard" or "measure". The qubits are represented in the python code using NumPy library as vectors. The phase estimation algorithm which is used to measure complex observables like the energy described by a Hamiltonian can be implemented by the qubit simulator. The angle with which the qubit is rotated is measured by estimating the probability of measuring a one for each scale value. A qubit should be prepared in the $|1\rangle$ state and will be used as a target for all the measurements as it is the eigen state for the rotation. Now, a control qubit will be allocated [3] and for each scale value it will be prepared in the superposition state. This control qubit will be rotated by the product value calculated by multiplying the scale value with the secret angle. After the rotation, the control qubit is unprepared, and the measurement is made. This technique helps us to reuse the same control qubits for multiple rotations. This secret angle with which the qubit is rotated cannot be concluded in a single measurement due to the limitation of no cloning theorem. Hence, the scale values are multiplied with the angle and the respective probabilities are measured. Analyzing the recorded values, the angle with which the qubit is rotated can be concluded. This is done by writing a python host program in the classical computer which calls the Q# operations which are already defined in the same classical computer. The Q# library has inbuilt functions which allocate, deallocate, measure the qubits and to convert them to classical format. The Q# program written in the classical computer uses the Q# library and defines functions to rotate a qubit, hide the rotation angle and estimate the probability of the scale value after controlled rotations. So, when the host program is executed it calls the Q# functions and the angle value which is calculated based on the probability and scale values is returned.

VI. FUTURE WORK

The simulator implemented needs to be validated to assure the functional correctness, robustness etc. This can be carried out by providing a set of inputs to the qubit simulator and comparing the outputs with the Hamiltonian which is specified as the design goal for the qubit simulator. One another way which can be used to increase the confidence into the output of quantum simulation is simulating different algorithms in the qubit simulator and comparing their execution time and bit error rates.

REFERENCES

[1] Sarah c. Kaiser and Christopher e. Granade. 2021. Learn Quantum Computing with Python and Q# - a hands-on approach. ISBN: 9781617296130

[2] Jose Carrasco, Andreas Elben, Christian Kokail, Barbara Kraus and Peter Zoller. 2021 Theoretical and Experimental Perspectives of Quantum Verification.

[3] Jinyoung Ha. and Jun Heo. 2021. Performance Comparison of Quantum Phase Estimation Algorithm with Different Number of Register Qubits on Noisy Quantum Processor. 2021 IEEE Region 10 Symposium (TENSYMP) | 978-1-6654-0026-8/21.

AUTHORS

First Author – Elanchezhiyan T, MTech (VLSI Design), VIT University, Vellore, Tamil Nadu, India.
elanchezhiyan.t2020@vitstudent.ac.in
Second Author – Dheerendra Singh Tomar, Silicon Validation Lead, Intel Corporation, Bengaluru, Karnataka, India.
dheerendra.s.tomar@intel.com