Implementing qubit validation framework 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.12.04.2022.p12410
http://dx.doi.org/10.29322/IJSRP.12.04.2022.p12410

Paper Received Date: 15th March 2022
Paper Acceptance Date: 1st April 2022
Paper Publication Date: 6th April 2022

***Abstract*-** Phase estimation algorithm is an ineluctable algorithm in the field of quantum computing as it has significant applications for itself, and it is frequently used as a subroutine in other quantum algorithms like Shor's and Grover's. Here, a quantum bit simulator is presented using which the well-known phase estimation algorithm is simulated and validated as an example. This quantum bit simulator helps us to achieve an understanding on quantum computers, explore existing quantum concepts like quantum cryptography, quantum chemistry etc. and to develop new quantum applications.

***Index Terms*-** Quantum algorithms, Quantum computing, Quantum entanglement, Quantum superposition, Phase estimation, Validation, Qubits, Simulation, Python, Q#, Phase kickback.

I. INTRODUCTION

Quantum computing is a skyrocketing technology which makes use of the laws of the quantum physics to solve complex problems which cannot be solved using presently available classical computing resources. Quantum computing has emerged as a popular field for research and source of hype over the last few years because of the impact they can make in the society. Given the hype around the quantum computing, it is necessary to understand the capability and limitations. Quantum physics involves understanding of matter and energy at the basic level which leads to a better understanding on basic building blocks of nature. Few concepts which help in understanding quantum computations include the wave-particle duality, quantum superposition, quantum entanglement, uncertainty principle, etc. Since most of the quantum concepts are hard or sometimes even impossible for us to visualize, mathematical representation of them becomes significant. Quantum phenomena and their probabilistic nature is better understood with the help of mathematical operators and equations. Quantum superposition explains the nature of a quantum object to exist in two different states at a same time which can be seen analogous to a ripple in the surface of a pond which is a combination of two waves overlapping. Quantum entanglement explains about the connectivity of two objects which can be thought as a single system even if they are distant from each other. It states that, by knowing information about one object we can retrieve the information about the other and vice versa. The quantum entanglement can be well understood by looking on Schrodinger's cat experiment. Uncertainty principle states that no two properties of any object, cannot be measured precisely at a same point of time. Even with the high level of performance offered by the present-day supercomputers, few of the real-world problems cannot be solved by the existing classical computing methods. For example, problems like the fastest matrix multiplication algorithm, polynomial integer factorization, simulation of chemical reactions cannot be solved precisely and would be much time consuming. To tackle all these limitations human mankind has now started to look quantum computing as the apt solution. But there are lot of challenges associated in making it happen since working with quantum concepts and computers is always quite complicated and challenging. For the quantum machines to work in compliance with the laws of quantum physics, it requires a nearly absolute zero temperature which can only be provided in laboratories. Hence, this makes quantum computer usage well limited to specific places which can provide cryogenic fridges. But thanks to the cloud technology we have today, using which the quantum computers in some data centers or laboratories can be made easily accessible from every nook and corner of the world. The most basic building block of data in telecommunication and computing is called a bit- short form for binary digit. Likewise, the basic unit of data in quantum computing is termed as a qubit - short form for quantum bit. A qubit is a quantum analogue of a classical bit which represents an indeterminate state called the superposition state. When a qubit which is in superposition state is measured, it always yields a deterministic result. As per present day trends, it is not possible to use a quantum computer for general purpose applications due to various limitations. But lot of researchers and companies is on track to achieve the goal of creating a commercially useful quantum computer by this decade. Problems having numerous computations, small input and output data sizes are the best fit to utilize the quantum computers. Researchers and students who love to explore more about quantum computing have limitations since availability of

the actual quantum hardware will be less due to its cost and complexity. Those limitations are shattered by simulating the quantum computing concepts using classical computing resources. Though quantum simulations have several of its own limitations, it is seen as a gateway for the better understanding of quantum computing. With the present-day classical computing resources available, we can maximum simulate up to 40 qubits. This limitation won't be a major issue as quantum bits have a fascinating property called the reversibility which allows reuse of the qubits. All the quantum operations performed are reversible like the classical NOT operation. Quantum initialization and quantum measurement are the two operations which cannot be reversed. The exploration of this captivating technology began way back in 1980 when physicist Paul Benioff showed that the simulation of Turing machine is possible by proposing a quantum mechanical model for the same.[5] Yuri Manin and Richard Feynan followed it up with their perspective that a quantum computer can simulate computations which a classical computer won't be able to do. Many researchers and scientists came up with lot other proposals and many quantum algorithms were invented. After some heavy investments and prolonged efforts in research and development, we have a quantum computer with a capacity of 127 qubits at the present day. Constant efforts have been made for the commercialization of quantum computers and still quantum computing remains a wide-open area for active research. In our previous work [1], the idea of implementing a quantum simulator was proposed and the basics of quantum computing, validation, simulation, and quantum algorithms have been explained in a detailed manner. As an extension to that idea and we have implemented the simulator in real time and have validated the results in this work. The organization of this paper is as follows: The quantum Fourier transform, and the phase estimation algorithm has been explained in a detailed manner in Section 2 and 3. In Section 4 and 5, more information on simulations and validation has been provided. Section 6 discusses the implementation of the proposed quantum simulator model. The results arrived are posted in the Section 7 followed by conclusion in Section 8.

II. Quantum Fourier Transform

One of the remarkable discoveries in quantum computing is the finding which stated that tasks which are not feasible to be performed on a classical computer can be performed efficiently in a quantum computer. Taking prime factorization of an n-bit integer as an example, it would require the number of operations as huge as the exponential of the number of bits of the given integer which makes the problem nearly impossible to be solved using a classical computer [4]. In contrast, the quantum computer can solve the same problem in an efficient manner which is as fast as exponential number of times compared to the classical computation. Fourier transform is an important concept in classical computing and has applications ranging from signal processing to complexity theory. We extend the concept to develop the quantum Fourier transform which will be the basic building block for phase estimation and many other quantum algorithms. Noticing the speedup and increase in efficiency one might think that the quantum Fourier transform speeds up the Fourier transform computations on classical data, but it is the quantum implementation of the Fourier transform performed on the quantum mechanical amplitudes.[7] It also enables the phase estimation which estimates the phase of unitary operator by approximation which would eventually allow us to solve problems like order-finding problem, factoring problem, the problem of counting solutions etc. The most preferred way to solve a problem in computer science or mathematics is to transform the given problem into some other problem for which the solution is already known. Quantum computers has the marvelous advantage that these types of transformations are computed swiftly using them. Discrete Fourier transform is one such transformation which takes in an input vector of complex numbers, $c_0, c_1, \ldots c_{M-1}$ where M is the length of the input vector. The output will be the transformed data of input vector of complex numbers $d_0, d_1, \ldots d_{M-1}$, as follows:

$$d_j = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} c_k \, e^{2\pi i j k / M} \tag{2.1}$$

The quantum Fourier transform is an analogous operation to the Fourier transform but it is performed on an orthonormal basis $|0\rangle \ldots |M-1\rangle$ with the following action on the basis states,

$$|c\rangle \rightarrow \frac{1}{\sqrt{M}} \sum_{d=0}^{M-1} e^{2\pi i c d / M} \, |d\rangle \tag{2.2}$$

The transformation explained above is a unitary transformation and hence can be implemented using a quantum computer. Assuming, M = 2^m where m is an integer, and the orthonormal basis $|0\rangle, \ldots, |2^m - 1\rangle$ will be the computational basis for an m-qubit quantum computer. Now, the state $|c\rangle$ can be represented as $c = c_1 2^{m-1} + c_2 2^{m-2} + \ldots + c_m 2^0$. The quantum Fourier transform can be expressed as a product representation by applying some steps of linear algebra to equation 2.2 as follows,

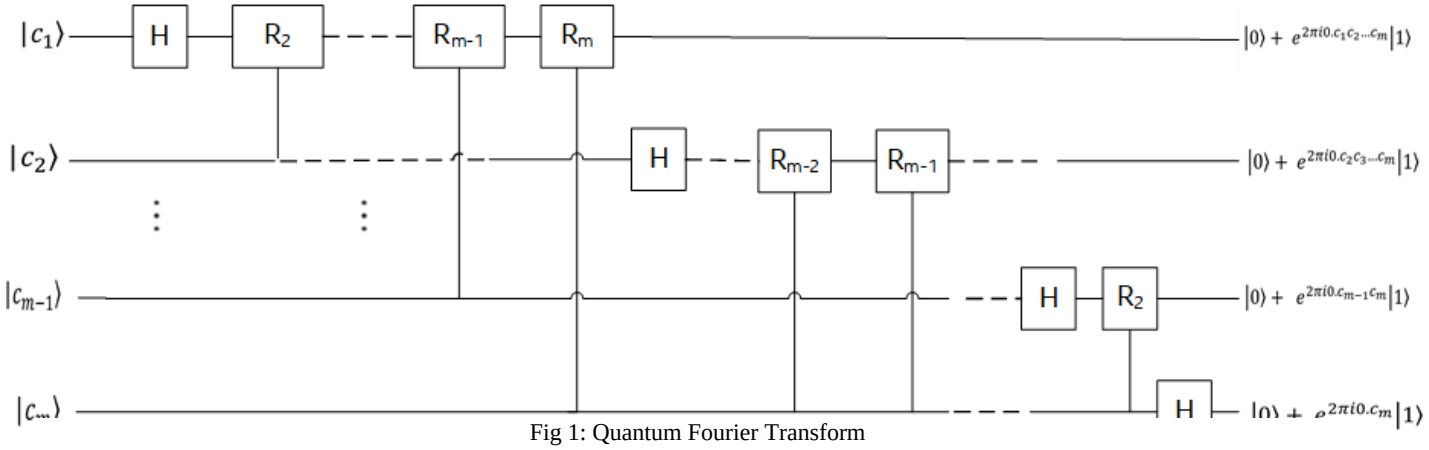

Fig 1: Quantum Fourier Transform

$$|c_1 \dots c_m\rangle \rightarrow \frac{1}{2^{m/2}} \left(|0\rangle + e^{2\pi i 0.c_m}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.c_{m-1}c_m}|1\rangle\right) \dots \left(|0\rangle + e^{2\pi i 0.c_1 c_2 \dots c_m}|1\rangle\right) \quad (2.3)$$

An efficient quantum circuit as in Fig.1 computing quantum Fourier transform can be constructed using the above explained product representation from equation 2.3. The circuit makes use of Hadamard gate and a controlled rotation operation which is a unitary operation denoted by R_z as follows:

$$R_z = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^z} \end{bmatrix}$$

This circuit provides us an algorithm for performing the quantum Fourier transform which is exponentially faster than the classical computing methods. As discussed earlier, we might not have a way to speed up the Fourier transform using this quantum Fourier transform but we can develop more algorithms by applying the quantum Fourier transform.

III. Quantum Algorithms – Phase Estimation Algorithm

A higher-level description of problems using algorithms will be the right approach before we start programming our quantum devices. An algorithm is independent of the programming language and operating system used which would aid in better understanding of problems. The quantum simulator which we have developed is like any other simulator which is used to simulate any specialized hardware like FPGA or a microcontroller. But the key difference to note is that the quantum computations will be simulated using only a classical computer. Since, we use classical computers to simulate quantum hardware the organization of steps which needs to happen in classical computer and quantum devices becomes more significant which will be provided by the quantum algorithms. The quantum algorithms will abide by the laws of quantum physics and will be implemented using a quantum program which comprises of a classical program which initializes or measures data from a particular state. Few of the impactful quantum algorithms include - Deutsch-Jozsa algorithm, Grover's Algorithm, Phase estimation algorithm, Shor's algorithm. Of these few, phase estimation algorithm is taken as an example and was simulated and validated. Using quantum measurements, we can only measure qubits but not phases. It should be noted that phases cannot be measured but can be estimated. [3] The phase or eigenvalue of an eigenvector of a unitary operator can be estimated using the phase estimation algorithm. Phase estimation algorithm is often used as a subroutine in other quantum algorithms like Shor's and Grover's to build large complex programs. It is the outset for many other quantum algorithms and depends on the inverse quantum Fourier transform. Phase estimation estimates the phase of an eigen vector $|+\rangle$ performing a unitary operation with an eigen value $e^{2\pi i\theta}$ where θ is the unknown phase value to be estimated. The idea is to use two registers one with the target qubit in the stable state and other with the control qubit in the superposition state. Basic quantum operations like Hadamard operations and controlled rotations are handy implementing this algorithm. Hadamard gate is used to prepare a qubit in the superposition state. Oracle concept which is analogous to the black boxes and can prepare a qubit in the superposition state and rotating it in a controlled manner is also used to perform phase estimation. Quantum Fourier transform is a key to the phase estimation subroutine which is in turn a significant part of many quantum algorithms. Phase estimation involves preparation of a qubit in the superposition state using Hadamard gate, then applying the hidden black box operation on the qubit and ultimately measuring the qubit after application of inverse quantum Fourier transform.[7] The inverse quantum Fourier transform can be realized by reversing the circuit in Figure 1. The first stage of the phase estimation involves application of Hadamard operation to the target qubit and performing controlled rotations on the control qubit in the second register. It must be noted that the inverse quantum Fourier transform will be applied to the target qubit. The next stage involves the measurement of the target qubit, and the control qubit is unprepared for reuse. The results of phase estimation provide an output state close to $\sum_+ c_u |\theta`_+\rangle$, where $\theta`_+$ approximates the phase θ_+ and c_+ denotes the probability at which the + is chosen. The whole process of phase estimation can be well understood from the schematic diagram Figure 2. It's

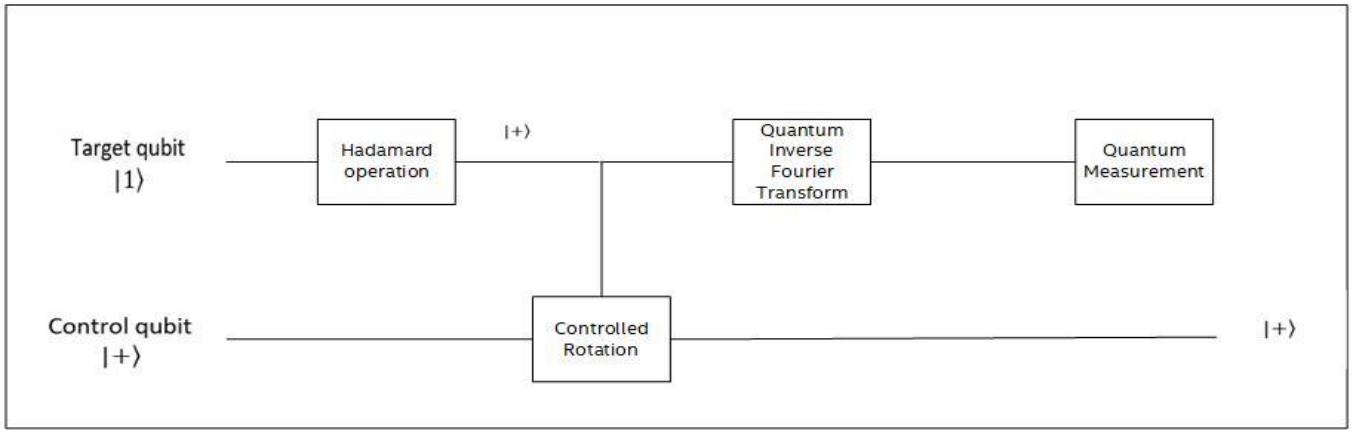

Fig 2: Quantum Phase Estimation stages

important to note that the phase estimation subroutine is used to solve fascinating problems like the order-finding problem and the factoring problem which can help in developing efficient algorithms having the potential to break even the RSA public key cryptosystem.

IV. Simulators

Simulation in engineering perspective is often referred as imitation of a system's functioning using computer technologies and programs that would evaluate the risks and benefits in a virtual environment. The actual behavior of a system cannot be perfectly same as the simulated one, but simulation gives us a better insight about various dimensions of that operation and the data collected can be used for further analysis and enhancement. Simulators in general has a very rich history in various fields like medicine, aviation etc. even in the late 18th and early 19th century. The best example would be the Link Trainer Flight Simulator which was built in the late 1920's and found great application during the World War II using which more than five lakhs of U.S fighter pilots got trained. After the invention of computers, simulations have evolved a lot and today's simulators have been employed in various fields like medicine, automotive, aviation, software, defense, aeronautics etc. Using simulators, we can reduce the cost of a particular experiment since hit and trail method will be too costly in few disciplines. Simulators helps us to handle unpredictable behaviors and phenomena and to avoid delay and failures. The input sources and output devices to a simulator can vary widely according to the nature of the simulator. Simulations allows us to handle larger amounts of data which may not be feasible or may be costly with physical systems. Simulations also lets us to view the design at various abstraction levels which aids us to have a better understanding on the architecture or design. In earlier days, simulations are carried out only by experts as they were highly complicated, and the results were also ambiguous. But thanks to the technological advancements, today simulations are pretty much simpler and can provide nearly accurate results. As technology grew rapidly, experts have also found a way to simulate a quantum operation using available classical computing resources. This was a breakthrough in quantum computing research as researchers and students who cannot afford to use the real quantum hardware can still try to have a better understanding of quantum concepts and can contribute to the research. With the present day's classical computing resources available we can simulate a maximum of up to 40 qubits which would be sufficient to perform most of the basic quantum operations. It is significant to note that simulation costs are just less than 1% compared to redesigning an implementation. This massive difference in cost suggests that one should always perform simulations before performing the actual production to escape risks. But there are few limitations and challenges which needs to be mitigated in building and using a simulator. The simulator should mimic the actual system to an extent and the reliability of the results generated should be validated. An engineer requires a basic level understanding of the physical system before using or building a simulator and few types of simulators require programming knowledge and expertise. We would be taking the full advantage of simulators to achieve a better understanding of quantum algorithms and quantum circuits and to develop new efficient quantum algorithms.

V. Validation

Validation in general refers to checking the validity or accuracy of a system and declaring it officially acceptable. In engineering perspective validation maybe related to the process of testing whether a particular system or design delivers the desired output in a safe and consistent manner. System on chip validation involves checking the functional correctness, robustness, meeting the power and timing constraints, reliability of the chip in the lab setup. System on chip validation involves three stages namely the pre-silicon phase, post-silicon phase and the in-field debugging. System on chip design is very rigid in producing flawless chips as redesign or correcting a defect in the chip design is too costly, hence more than three fourths of the overall efforts, time and resources are spent in validating a chip which makes it a major bottleneck in the system on chip design.[8] Pre-silicon validation refers to all the checks done before the fabrication process begins. Post-silicon validation refers to the efforts carried out on validating manufactured chips after the fabrication process in the real physical environment as a trial run before the mass production begins. The limited controllability and observability during the post silicon validation posts a huge challenge to the engineers. It is the responsibility of the post silicon validation engineer to ensure that no bugs escape to mass production as post silicon validation is the final level of defense against bugs to check the proper functioning and integrity of the design before mass production. The post silicon validation involves multiple stages like test plan generation, building of debug infrastructures, generation of tests, localization of error or defect etc. One of the hardest challenges with post silicon validation is that it will be performed usually under a very less time-to-market. In most cases, the simulation results play a major role in analyzing the system and making changes in the design or process. Hence, seamless operation and reliability of simulators is highly desirable. Validating a simulator refers to the process of examining the extent to which a simulation model and its related data can be are a precise representation of the physical world from the view of the intended usage of the model. If the validation results are not satisfactory, changes are made to the simulation model in such a way that there are very little differences with the observed behavior from the desired behavior. There are lot of challenges involved in validating a simulator than validating the real system due to the limited controllability and observability as simulators are just mathematical and computerized models of the real physical systems. Extra memory units called the trace buffers which stores the values of few specific registers during runtime can be used to increase the silicon observability during post silicon validation. But this too has a limitation that due to the small depth and width of the trace buffers very few registers can be traced. Moreover, reproducing the same data or bug in a simulator is very hard. Simulators can be validated by application of a set of test inputs, measuring the output and comparing them with the benchmark output which has been predefined based on few standards by the developer. The qubit simulator proposed in this paper is also validated using the same procedure. Validating a simulator helps

in avoiding unwanted ambiguities around the simulated results and helps with seamless flow in understanding the behavior and working of the actual system.

VI. Implementation

Quantum computers cannot be used as handheld devices as they require a nearly zero temperature environment which is feasible to be maintained only in laboratories and datacenters where the quantum hardware is enclosed inside a cryogenic fridge. Utilizing the advantages of internet and cloud services we can remotely access the quantum computer and make use of the quantum advantage from any part of the world through our classical computing resources. This accessibility is done through a classical program written in any of the programming languages like python or .NET languages like C# or F# which acts as a host program to the quantum program which will be applied to the real quantum hardware. The quantum program is written in any of the quantum programming languages like Q#, Qiskit, QCL, etc. The quantum program allocates required number of quantum bits, initializes them, perform quantum operations on them and then measures the results. Here we have proposed a qubit simulator in which everything will be carried out within the classical computer itself. We will still use the host program to send the instructions to the quantum program which in turn invokes the qubit simulator, perform operations, and measures the result as in figure 3.[1] We have used python qsharp package to write programs for the host program and for the quantum program we have used the advantage of Microsoft's open-source quantum programming language Q#. The interoperability between the quantum development kit and the Q# is achieved by qsharp python package which makes it easier to simulate functions and operations from within python. We have taken the help of anaconda scientific distribution for managing python and other scientific tools locally. A separate conda environment was created for the seamless operation of qubit simulator which comprises of the packages required. The proposed qubit simulator model can be used to simulate and validate various quantum operations in the locally available anaconda environment. Quantum operations can be understood better by decomposing them into their eigen states and the phases applied to each eigen state. It should be noted that global phases of states cannot be measured whereas local phases can be measured. The idea is to perform controlled versions of an operation so that we can turn the global phase into a local one which can be measured. This can be achieved by a common programming technique called phase kickback which states that the operation applied to the target qubit changes depending upon the control qubit. By controlling a given instruction on a qubit in the superposition state, we will be able to learn what would have been a global phase without control. All unitary operations i.e., a quantum operation that doesn't allocate, deallocate, or measure qubits can be controlled. Some of the controlled operations in Q# include Controlled-NOT, Toffoli, Controlled-SWAP, etc. [2] The controlled rotations on a target qubit kicks back the phase onto the control qubit which enables us to reuse the same target qubit over and over. When we apply the controlled rotation to a target qubit which is in the $|1\rangle$ state, the target qubit stays in the same state, but the control qubit changes. Reusing the same target register increases efficiency as preparation of right input state involves calling of lot quantum operations. We have picked the well-known phase estimation algorithm to be simulated and validated using two qubits in the qubit simulator as an example. The detailed explanation of phase estimation algorithm is provided in the section III. As in figure 4, the first step is to allocate an angle in a particular range which is then multiplied with a random value in the range 0 to 1 to be termed as hidden angle which will be estimated. The hidden angle along with the input scale values are passed on to the RunPhaseEstimation UsingControlledRotations function of the quantum program by the classical host program's run_phase_estimation_at_scales function. The hidden angle is sent to the HiddenRotation function which uses the Q# hidden argument and user defined types to our advantage to implement the simulator. This function which wraps up the hidden angle with the other two hidden arguments – the input scale value and the target qubit into a scalable operation which is a user defined type. The next step is to allocate a target qubit and prepare it in the $1\rangle$ state by flipping it using the quantum NOT gate. Now the returned user defined type from the HiddenRotation function is passed as an argument to the EstimateProbabilityAtScale function which takes in a hidden argument – the input scale value and the number of measurements carried for a scale value along with the hiddenrotation which is a Q# user defined type. It should be noted that the EstimateProbabilityAtScale function is applied to all the scale values in the input scale array. Each scale value will

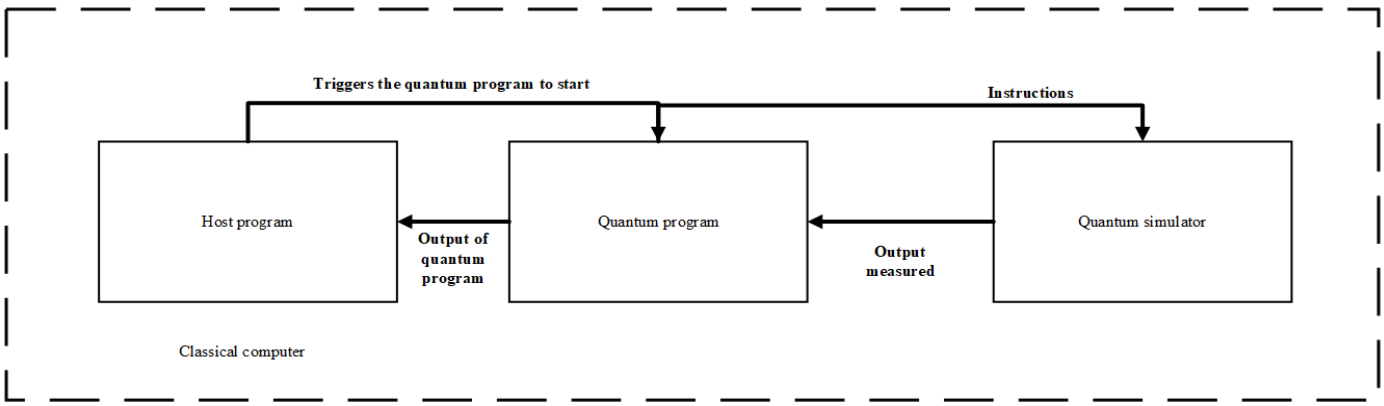

Fig 3: Implementation of qubit simulator

be multiplied in the ApplyScaledRotation function with the hidden angle which gives the rotation angle with which the controlled rotation operation will be performed on the allocated qubit. EstimateProbabilityAtScale involves allocating a control qubit for the number of times the measurements will be made on the target qubit for each scale value. The control qubit will be prepared in the superposition state using the Hadamard operation. It is important to note that the controlled rotations will be applied on the target qubit, but the measurements will be made on the control qubit which follows the phase kickback technique discussed above. R1 operation which is like the Rz operation is used to rotate the qubits in a controlled manner which supports the adjoint and the controlled functor. Once a finite number of measurements are made on a given control qubit for a given scale value, the estimated probability will be returned to the RunPhaseEstimationUsingControlledRotations function which in turn sends the estimated probability data respective to the input scale value in the form of an array to the classical host program's run_phase_estimation_at_scales function. The probability of measuring a one is calculated by incrementing the total number of ones after each measurement is made and dividing that value by the total number of measurements made for that each scale value. After each measurement is done, only the control qubit is prepared again but the target qubit is reused. Before handing over the control to the host program, the target qubit is unprepared to the |0> state. The host program makes use of the various python packages like numpy, typing, qsharp and SciPy. The numpy library is used to generate the random hidden angle and the input scale values in each range. The hidden angle which was picked can be estimated by applying the curve_fit function of scipy.optimize python library on the calculated probability data. The curve_fit function also helps us to understand the measurements better by representing the relationship between the input scale values and the probabilities of measuring a one in a graphical manner. The working of the simulator can be validated by providing different test inputs i.e., different random values of hidden angles and comparing them with the estimated rotation angle.

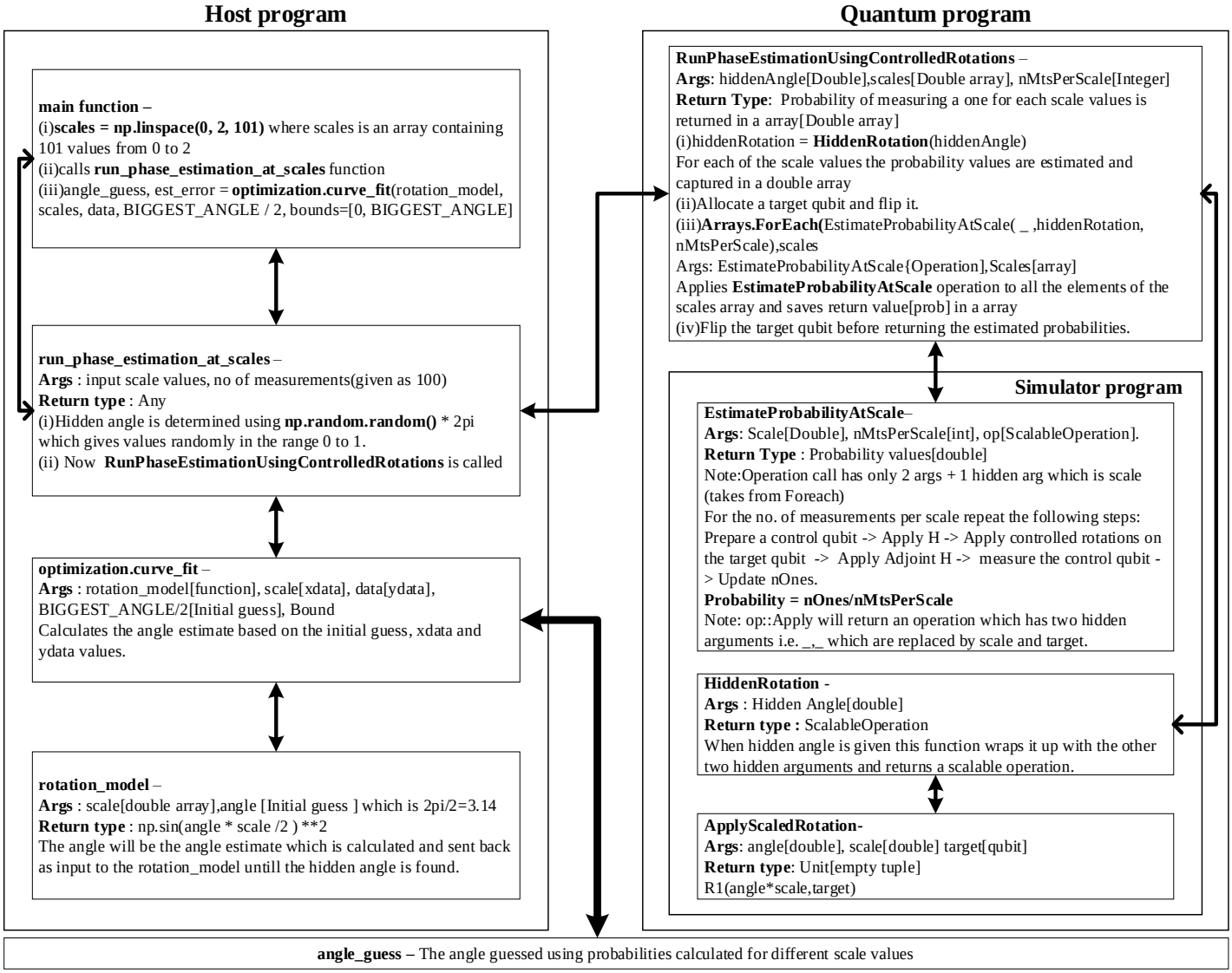

Fig 4: Code flow transfer between the classical and quantum program

VII. Results

The phase estimation algorithm was taken as an example and was simulated using the qubit simulator. The hidden angle values to estimation algorithm were provided in the range 0 to 2π. The input scale values were taken in the form of an array having 101 elements in the range of 0 to 2 separated by a difference of 0.02. The number of measurements made for each scale value on the control qubit was assumed to be 100. Considering all the input values and the probabilities measured using quantum program, the hidden angle was estimated making use of the python scipy library. To validate the functional correctness of the simulator, we have taken different test inputs and compared the estimated angles with them. When the hidden angle was 3.113 the angle estimated was 3.1113 and the relationship between scale values and estimated probabilities are plotted as in figure 5 for input scale values in the range 0 to 2.

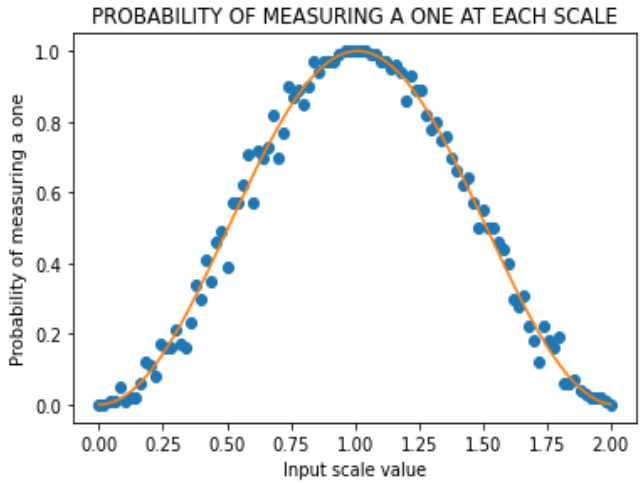

Fig 5: Relation between input scale and probabilities when hidden angle is 3.113

When the hidden angle was 4.7333 the angle estimated was 4.7386 and the relationship between the input scale values and the probabilities is plotted in the figure 6. Likewise, few other inputs can also be applied, and their respective estimated angles can be compared and validated.

VIII. Conclusion and Future Work

A quantum bit simulator based on the quantum computing concepts was presented. Here, a novel approach of expressing the quantum computations by producing graphs of the probability of measuring a qubit's state was experimented. This idea of qubits simulation can be extended to simulating different quantum algorithms like Shor's algorithm, Grover's algorithm etc. in the qubit simulator modifying only the simulator part of the quantum program in the figure 4. More test input patterns can be provided by modifying the python host program to enhance the validation capabilities. By comparing their execution time and bit error rates, enhancements can be made in the qubit simulator

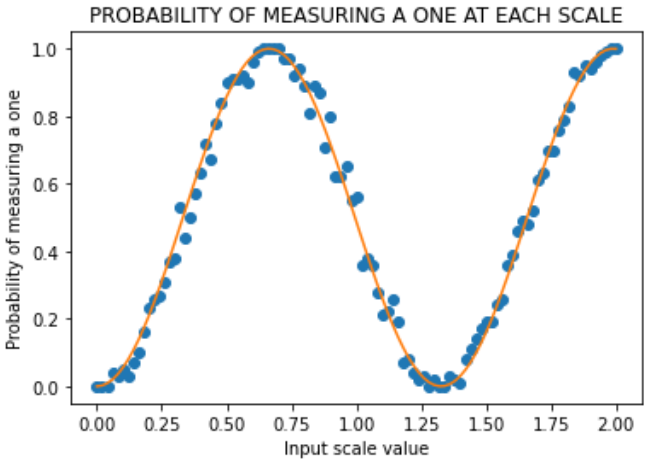

Fig 6: Relation between input scale and probabilities when hidden angle is 4.7333

ultimately leading to reliable and confident simulations and comparisons help in figuring out efficiency of the quantum algorithms and to develop new efficient quantum algorithms.

References

[1] Elanchezhiyan T and Dheerendra Singh Tomar, "Simulation and validation of qubits based on Phase estimation algorithm,"in *International journal for scientific and research publications*, Volume 12, Issue 1, January 2022 Edition. ISSN 2250-3153.

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

[3] Luca Pezze and Augusto Smerzi, "Quantum theory of phase estimation", *QSTAR, INO-CNR and LENS - Largo Enrico Fermi 6*, Firenze, IT .

[4] Peter.W.Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring", 0272-5428/94 $04.00 *0* 1994 IEEE.

[5] Paul Benioff, "The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines", 0022-4715180/0500-0563503.00/09 1980,Plenum Publishing Corporation.

[6] Kenji Sugisaki, Chikako Sakai, Kazuo Toyota, Kazunobu Sato, Daisuke Shiomi and Takeji Takui, "Bayesian phase difference estimation: a general quantum algorithm for the direct calculation of energy gaps", in *Royal Society of Chemistry*, DOI: 10.1039/d1cp03156b.

[7] Michael A. Nielsen & Isaac L. Chuang, "Quantum Computation and Quantum Information", ISBN 978-1-107-00217-3 Hardback.

[8] Subhasish Mitra,Sanjit A. Seshia and Nicola Nicolici, "Post-Silicon Validation Opportunities, Challenges and Recent Advances", DAC'10,June 13-18,2010,Anaheim, California, USA. ACM 978-1-4503-0002-5/10/06.

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

Correspondence Author – Elanchezhiyan T, MTech (VLSI Design), VIT University, Vellore, Tamil Nadu, India.
elanchezhiyan.t2020@vitstudent.ac.in