Approximate Quantum Array Multiplier

Aden Crimmins*, Sonia Lopez Alarcon†
KGCOE Department of Computer Engineering, Rochester Institute of Technology
Rochester, New York
Email: *abc8255@rit.edu, †slaeec@rit.edu

***Abstract*—Multiplication is a frequent computation in many algorithms, classical and quantum. This paper targets the implementation of quantum integer multiplication. Quantum array multipliers take inspiration from classical array multipliers, with the result of reduced circuit depth. They take advantage of the quantum phase domain, through rotations controlled by the multiplier's qubits. This work further explores this implementation by applying approximate rotations. Although this approach can have an impact on the accuracy of the result, the reduction in depth can result in better outcomes when noise is involved.**

***Index Terms*—Qubit, Quantum Entanglement, Multiplication, Fourier Transform, Approximation, Control Gates, Phase Domain, Basis, O-Time Complexity, Array Multiplication.**

I. Introduction

The design of quantum circuits faces a race against time to beat decoherence and the impact of noise. The depth of a quantum circuit, characterized by the number of computational steps or gate layers, is the key design metric that is proportional to execution time. Therefore, finding designs that minimize depth is critical to reaching meaningful results that are as protected from noise, decoherence, and, in sum, quantum error as possible.

Much like classical computing, quantum circuits can make use of basic circuit designs that act as building blocks for large-scale algorithms. This work focuses on one of these basic operations, the multiplication of integers implemented as quantum information. Improvements made to a quantum multiplication algorithm could prove useful for different applications, such as modular multiplication in Shor's algorithm [1], or could play a role in the implementation of oracles for Grover's search algorithm [2]. In previous work, quantum array multipliers showed a significant reduction in depth against other naïve implementations [3]. Reducing depth is based on the efficiency of the computations as well as on how parallelism is exploited through them. One other advantage of the quantum array multiplier is that its implementation is valid for any integer multiplication value, acting as an insertable module that encodes the multiplicand and multiplier in two corresponding quantum registers. This is opposed to other designs [4] that vary the implementation dependent on intermediate measurements.

In this paper we will show one additional advantage of the quantum array multiplier, proposing the *approximate quantum array multiplier*. In the quantum array multiplier, the multiplication is performed through rotations controlled by the qubits of the multiplier and multiplicand. The magnitude of these rotations decreases with the significance of the control qubits. Just like with any other computation, there is a choice to be made on the precision of the computation. It is possible to implement the design discarding the smaller rotations for being insignificant in the greater scheme of the results. This work looks at this approach to find that the depth of the quantum array multiplier particularly benefits from this simplified design, with little impact on the accuracy of the resulting product. Further, when noise is considered, the reduction in depth results in more accurate results in the approximate case than in the exact case.

This material is based upon work supported by the National Science Foundation under Award No. 2300476

II. Related Work

Quantum arithmetic has been developing for a few decades, with new algorithms and techniques being proposed for efficient and fast operations on qubits. The arithmetic operation focused on within this paper is multiplication, as it is a fundamental building block for a wide range of applications in many different algorithms and calculations.

Adders are relevant to this discussion on multiplication, as in most cases, multipliers involve additions in one way or another. The implementation of quantum addition using a full adder is given precisely by Sohel et al. [5]. A basic full adder can be described with CNOT and Toffoli quantum gates. The n-qubit design requires an n quantum register and an n classical register to hold the values for input and output in the quantum domain and the measured value in the classical domain. The paper demonstrated the complete implementation of the full adder truth table, with multiple intermediate steps in which the qubits are used to store the intermediate values that are not required at the end. The ripple carry adder circuit proposed by Vedral et al. [6] demonstrates the addition of two n-bit numbers as input and outputs a single bit, the most significant bit of the sum. These circuits require $n-1$ ancillae qubits, with $4n+1$ CNOT gates and $4n+1$ Toffoli (doubly controlled-NOT) gates, resulting in limited parallelism. The circuit was improved and was reduced to one ancilla qubit proposed by Cuccaro et al. [7] with $2n+1$ Toffoli gates, $5n+O(1)$ CNOT gates, and $2n+1$ negations; the estimated depth is $2n+1$. A different approach was taken by Draper [8] in which the quantum fourier transform is applied to perform addition using phase rotations in the quantum phase domain. This approach has an estimated depth of $n+1$ in the presence of parallel computation, as well as no ancillary qubits required to calculate the result.

One technique used in classical computing is to limit the accuracy of the calculation in order to reduce power

consumption, area, and delay of hardware incurred. While this makes for faster circuits that consume less power, the downside of this is that the results in the end are not guaranteed to be exact, rather, they are within a certain range of error defined by the amount of approximation used. This type of computation is often used in image filtering, data mining, and pattern recognition as they often do not require exact results, but "good enough" results, due to the error resilience built into those applications. Research that took this approach on quantum computers was performed by Sajadimanesh et al. [9]. They propose four different approximate circuits combined to perform multiplication, showing the different trade-offs of each design in terms of accuracy and circuit complexity. This approach was based on binary implementation of the integer multiplication using Toffoli gates, and its approximate version. Binary implementations are however highly inefficient and approximate Toffoli gates in this case incur into high error on the overall calculation's result, since the solution is only guaranteed to be within a certain range of the actual solution and not exact at any point due to dropped terms.

Taking a different approach to approximation, we focus on the controlled application of phase shifts within the phase domain within this work. This technique was introduced by Barenco et. al. [10] and focused specifically on the application of the quantum fourier transform. It was shown that by removing rotations below a certain threshold, an approximate quantum fourier transform was able to achieve better performance than its non-approximate counterpart in the presence of decoherence. This is due to the fact that, for especially small phase rotation applications, the accuracy added to the circuit is outweighed by the decoherence accumulated in the application of the rotation. By removing these extremely small rotations, it is possible to increase the final accuracy when decoherence is present in the system. This idea was taken a step further by Draper [8], in which he proposed an addition circuit that took advantage of both the quantum phase domain and the same approximation technique utilized in the approximate QFT. By setting a limit to the phase shifts taking place in the addition circuit, one can significantly reduce the depth of the addition circuit.

The focus of this paper is on the application of this phase approximation technique to perform efficient multiplication of two integers in a quantum environment. In order to apply this technique, multipliers that operate within the quantum phase domain are considered.

A. Quantum Array Multiplier, Repeated Addition and Quantum Fourier Multiplier

The quantum array multiplier (QAM) was proposed in previous work [3], and it is inspired by classical array multipliers. In the classical design, each full adder takes in both the result of the current row and the previous row. In the quantum case, the phase shift is directly passed on to the product naturally taking care of carry dependencies. The design is summarized in Figure 1. The quantum array multiplier goes through four main steps: (i) The multiplicand and multipliers are initialized to the corresponding binary value (via X gates) (ii) the product register is placed in the phase domain (via Quantum Fourier Transform block) (iii) double-controlled rotations of decreasing magnitude are applied, and (iv) the product is converted back to the computational basis (via Inverse Quantum Fourier Transform).

A more naïve quantum multiplier design in the phase basis [4] is based on the repeated addition (RA) of the multiplicand in an accumulator, as many times as the multiplier indicates. This repeated addition also takes place in the phase domain. The basic functionality of this multiplier is as follows: The initial state of the accumulator is encoded in the phase domain through a Quantum Fourier Transform (QFT), so the multiplicand can control rotations of the qubits' phases depending on the significance of the qubit upon which the rotation is acting. The multiplier is then decremented by 1 and measured. This cycle is repeated until the multiplier is measured to be 0. Last, the accumulator is returned to encoding in the computational basis through an inverse QFT to extract the result of the multiplication. An example of this design is shown in Figure 2. Previous work [3] demonstrated the better overall depth performance of QAM over RA.

A Quantum Fourier Multiplier (QFM) has been incorporated into the Qiskit toolset [11][12]. This multiplier works similarly to QAM but has two drawbacks when compared to the first one in this section: on one hand, it does include $2n\pi$ rotations, which is pointless since they bring the phase to the same practical point, and on the other, only allows for multiplications in which multiplicand and multiplier have the same width, which results in reduced versatility as an insertable component when compared to QAM. This implementation was not compared against in previous work [3], but as it will be shown in this work, it is less efficient depth-wise while following similar trends.

These three multipliers have one main feature in common, which is that the multiplication is performed in the phase domain. As the multiplier, multiplicand, and product registers grow in size, there are increasingly small rotations to perform. This work discusses the benefits and drawbacks of discarding some of these rotations in the same manner as the previous approximation technique discussed. The method for applying this approximation and how it will impact depth will be shown next.

III. Approximate Quantum Array Multiplier

When the multiplication is performed in the phase domain, the minimum phase shift gets increasingly small as the input operands' size increases. Due to technology limitations, one may not be able to perform rotations of a magnitude below a certain threshold. In addition, since every gate inherently adds some amount of noise to the circuit, applying such rotations may contribute to noise more than to the accuracy of the calculation. There is a threshold under which the rotation is not worth applying. Barenco, et. al. showed that in the case of the QFT an approximate version, the Approximate QFT (AQFT), could be more accurate than the full QFT [10]. The implications are that the QFT and IQFT stages of the multiplier are able to be reduced by limiting the phase shifts

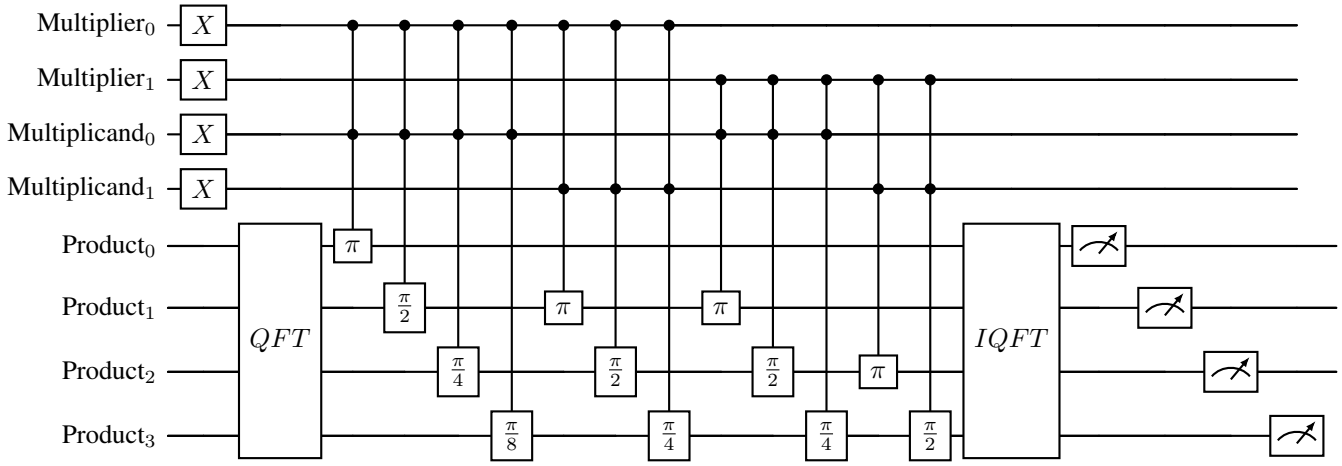


Fig. 1: Quantum Array Multiplier(QAM) with each stage included starting with the Initialization of the inputs, followed by the QFT stage, each of the row additions, and finally the IQFT.

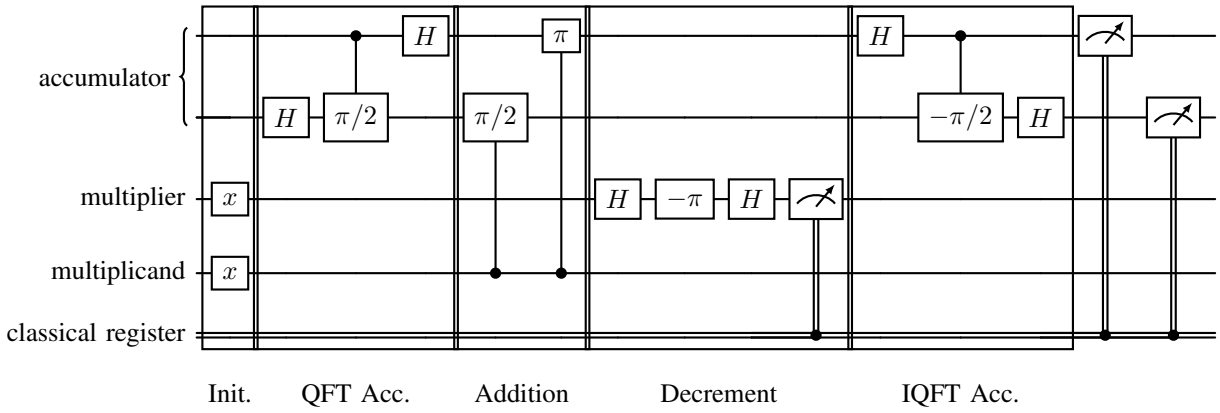


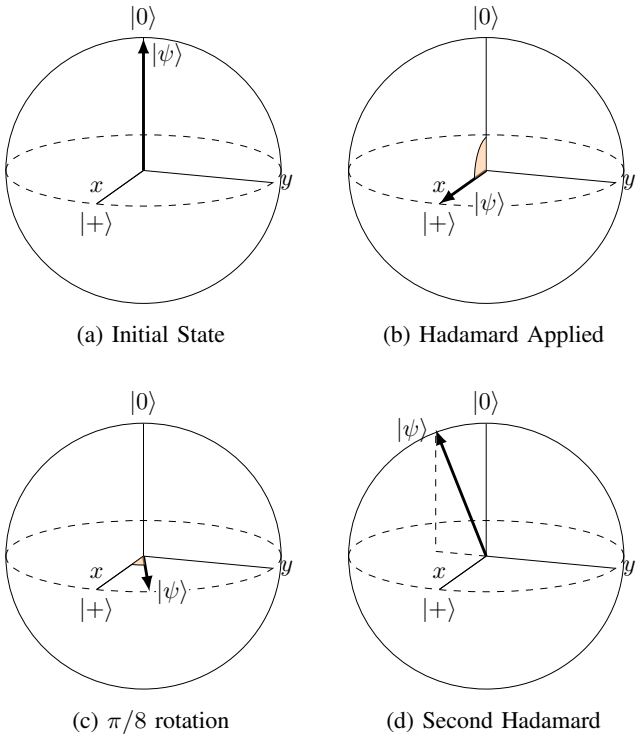
Fig. 2: Full 1x1 repeated addition multiplication circuit post improvements. Each stage outlined and labelled underneath the circuit.

to roughly $N = \lceil Log_2(Product\_width)+2 \rceil$ where the phase shift applied is represented by $R = \frac{\pi}{2^N}$. This approximation is applied to the QFT, multiplication, and IQFT blocks of the implementation. For example, in the case of a five-qubit input multiplicand and multiplier, the product register is ten qubits wide. In that case, $N = \lceil Log_2(Product\_width) + 2 \rceil = \lceil (Log_2(10) + 2) \rceil = 6$. The minimum rotation that can be applied then would be $R = \frac{\pi}{2^6} = \frac{\pi}{64}$. Any rotations below this value will not be implemented in this approximate circuit example.

Approximation in the phase domain does not affect to output like it would when precision is dropped in the classical setting. To better illustrate the effects of approximation, a one-qubit example is shown in Figure 3. Through this example, we intend to show how small rotations in the phase domain (whether added or removed) have a small impact on the measured outcome of the qubits, when the outcome is *basis-encoded*, as it is in the binary representation of the integer multiplication. In this case, the rotation is added rather than removed. As shown in Figure 3, the qubit is initially in state $|0\rangle$. After applying a Hadamard gate, it will evolve to the $|+\rangle$ state, now placed in the phase domain. To simulate the impact of the approximate phase shifting technique a ("small") phase of $\pi/8$ is applied, taking it out of the $|+\rangle$ state. After the small rotation, a Hadamard gate is once again applied to return the qubit to the computational basis. If there were no approximation, then the Hadamard would take it directly back to the $|0\rangle$ state, but with the small additional rotation, the final state is not exactly set to $|0\rangle$. Still, a measurement on the computational basis will project this state to the $|0\rangle$ state "most of the time" ($p_c = 0.96$) this case with probability and only sometimes it will wrongly measure the $|1\rangle$ ($p_w = 0.04$). This is the reason why even when rotations are removed, it is still possible to measure the correct output with high probability.

Two things must be noted in the case of quantum array multipliers, —or more generally, phase multipliers or adders.— First, removing multiple small rotations can potentially add up enough to take the final state "far" (phase-wise) from the computational basis, resulting in other basis states being measured with significant probability, and consequently lowering the probability of the correct solution. And second, the qubits that contribute with smaller rotations, and hence are more

likely to benefit from approximation are the most significant qubits of the product quantum register. If we were faced with a significant enough transformation of the probability distribution to result in other states having similar probability to the correct output, the new value(s) would not be in the vicinity of the correct value.



(a) Initial State (b) Hadamard Applied

(c) $\pi/8$ rotation (d) Second Hadamard

Fig. 3: Example Bloch diagram transformations showing the impact of approximation in the quantum phase basis. Starting in (a) with the initial state of $|0\rangle$ a Hadamard is applied transitioning to $|+\rangle$ as shown in (b). Once in the phase basis, a sample rotation of $\pi/8$ is applied, (c), followed by a Hadamard gate application returning the state *almost* to $|0\rangle$ as shown in (d).

The idea of the Approximate Quantum Fourier Transform (AQFT) was then advanced by Draper [8], in which he proposes using the same technique as the approximation of the QFT, but this time applied to the phase rotations of the quantum addition being performed. By eliminating the phase additions that fall below a certain threshold a reduced number of gates can be added, and using the same logic as the AQFT, in the presence of noise or decoherence, there is a potential increase in the accuracy of the circuit. Additionally highlighted by Draper was a method for parallelizing the execution of the addition operations in which each addition may be performed in $log_2 n$ time slices.

Both of these techniques were utilized within this work in the design of the AQAM. This design follows the same structure as the original QAM but includes a phase limit determined using the qubit size of the product register. Rotations below a threshold defined by $\lceil Log_2(Product\_width)+2 \rceil$ are dropped out of the calculation. Since the RA and QFM algorithms are also based on the phase domain, the same approximation technique applies to those two algorithms as well.

IV. Results

In this result section, the following algorithms are compared:

- Quantum Array Multiplier (QAM)
- Approximate Quantum Array Multiplier (AQAM)
- Repeated Addition (RA)
- Approximate Repeated Addition (ARA)
- Quantum Fourier Multiplier (QFM)

Each algorithm which computed in the phase domain and their approximate variations were compared, except QFM, which implements almost the same algorithm as QAM, just slightly less efficiently. Due to time constraints, the approximate behavior of this approach was estimated based on the behavior of AQAM.

The Qiskit [11] toolset was used to simulate and obtain metrics for these implementations. The main metrics used were the depth of each of the circuits as provided by Qiskit after undergoing decomposition and optimization as well as accuracy. The accuracy was assessed by providing the number of shots that resulted in the correct product.

A. Depth

The depth of these implementations depends on the numbers being multiplied specifically for the repeated addition implementations since it repeats the addition as many times as the multiplier requires. For that reason, two extreme cases were tested: $n \times n$ square multiplication and identity multiplication nx1.

1) Square Multiplication: multiplies each number n with bit widths from 1 to 20 having every bit initialized to 1. The resulting depths of each of the different multiplication algorithms can be seen in Figure 4. AQFM is estimated based on the behavior of the QAM since the same gates are removed in both cases. RA has the worst behavior in terms of depth. The exponential increase in depth comes from the number of repeated additions doubling each time the bit length of the inputs increases by 1. In addition, it is interesting to see that the approximation results in no savings in depth in the RA case. This is due to the structure of RA. Looking back at Figure 2, the addition step is followed by a decrement step in the multiplier. In the figure, this is only for a 1x1 case, but in a larger case, these two are interleaved. In addition, both blocks can be performed in parallel or partially in parallel. The addition also gets partially collapsed, to a lower depth than the decrement most of the time, since a fair number of rotations can be performed in parallel, which is not the case with the decrement step. For that reason, the decrement dominates the depth of the circuit in each pair addition-decrement, and therefore, dropping rotations in the addition portion saves no overall depth. For the QAM, AQAM shows a reduction in depth, as expected. It will be shown that AQAM's accuracy does not suffer significantly from this simplification. QFM performs worse than QAM and AQAM in terms of depth and the predicted behavior is that AQFM would still have higher depth than QAM for all the tested cases. The QFM

algorithm utilizes phase shifts of (2π, 4π, 8π, etc.) when performing the addition, which both does not fundamentally change the resulting values on the qubit being operated on and adds unnecessary gate layers.

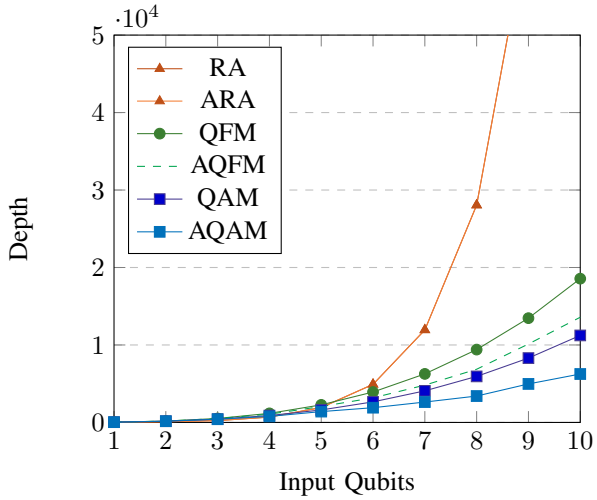


Fig. 4: Depth results for the square multiplication $n \times n$ of each of the algorithms being compared. AQFM is estimated based on the behavior of AQAM since it cancels the same gates in both cases.

2) Identity Multiplication: The second case for comparison between the different implementations is identity multiplication shown in Figure 5. In this test set the QFM was not included as, per limitations of its implementation in Qiskit, it cannot be used in cases where the bit widths of the two operands are different.

When the addition does not have to be repeated but once, RA excels in depth. In this simplest case, the dropped terms do save some gate layers, starting in the 8-qubit size experiment. This is due to the fact that, in this trivial multiplication case, the subtraction stage is smaller than the additions being performed. The difference, however, is so small that it is almost negligible. AQAM does save depth significantly when compared to the QAM, both of them being higher than RA. AQAM takes on a stepping form, not following a smooth curve. The rise in depth between the 16 and 17 qubit case comes from the logarithmic nature of the rotation terms dropped. When calculating the phase the minimum is $\pi/2^N$ where N is calculated with $\lceil Log_2(Product\_width) + 2 \rceil$. In the 16 input qubit case the product register has 32 qubits resulting in $\lceil \log_2(32) + 2 \rceil = 7$, whereas in the 17 input qubit case it is calculated to be $\lceil \log_2(34) + 2 \rceil = 8$, thus allowing for a larger number of phase shifts to be included in the circuit.

3) Multiplier value sweep: In order to better visualize how the depth changes between the trivial and square multiplication an additional set of tests was run. The third set of tests simulated how the depth of each circuit grows depending on varied input sizes. In all of the simulations, the multiplicand used as an input was 12 bits long, each of which was initialized to '1' corresponding to a decimal value of 4095. The multiplier

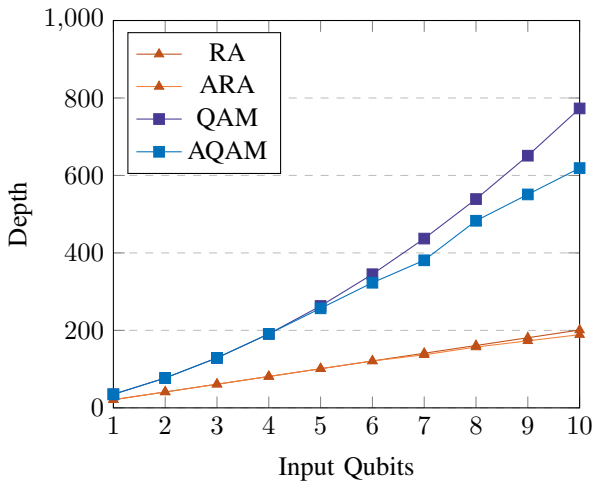


Fig. 5: Depth results for the identity multiplication $n \times 1$. Identity multiplication is not implementable in the QFM case, and therefore, it is not included in this plot.

was then varied from 1 to 4095 by left shifting the input and inserting a 1 into the least significant bit. The results of which are shown in Figure 6. Due to their structure not being influenced by the decimal values of the operands the QFM, QAM, and AQAM have a constant depth.

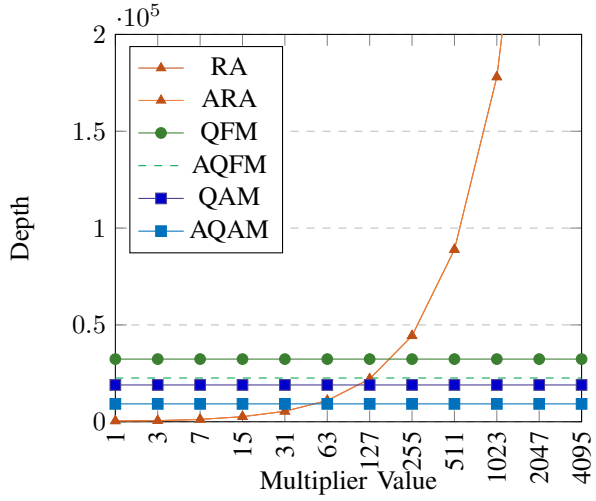


Fig. 6: Depth of each algorithm given a multiplicand value of 4095 and a multiplier value scaling from 1 to 4095. RA roughly doubles in depth at each interval as the number of times it needs to repeat has been doubled.

In the cases where the value of the multiplier is guaranteed or known to be small relative to the multiplicand, the RA algorithm has better depth. At an input value of 63, 127, and 255 it begins to perform depth-wise worse than the AQAM, QAM, and QFM respectively. This shows that the repeated addition is only the desired choice for 1.5% (63/4095) of the

possible input combinations when compared to AQAM, or 3.1% (127/4095) and 6.2% (255/4095) for QAM and QFM. When the input value is not known or is guaranteed to be fairly large, the ideal algorithms would be either the QAM or AQAM. As will be shown next, noise also has an impact on this decision.

B. Simulated Accuracy

Noiseless simulations as well as two different noisy simulations were tested in these comparisons. The noisy simulations include simulated noise from each of the primary sources as part of the Qiskit Aer simulation package. The specific model used to simulate noise in this work came from the GenericBackendV2. This was selected because it allowed for a variable number of input qubits rather than being limited to a small sample of qubits like those based on real devices. The downside of this is that it has a high default noise model, so noise accrues rather quickly in each of the simulations. The noise values are 10^{-5} and $5 \cdot 10^{-4}$ for single qubit and two qubit gates respectively. A second model was created to show potential future performance on systems with lower noise levels. Specifically the noise values for gate applications were reduced to 10^{-6} and 10^{-5} for single qubit and two qubit gates respectively. All simulations were performed with at least 1024 shots for the experiments that measure accuracy. The implementation is tested again in the two extreme cases, $n \times n$ and $n \times 1$.

The accuracy in this work comes from the probability distribution of the final state. Out of each simulation, the correct output is obtained with probability p_c. Design approximation and noise contribute to all the other outcomes with probability $1 - p_c$ across all other states. In these results, p_c is also used as the accuracy metric of the computation. These results are limited to seven input qubits due to the hardware-intensive nature of executing each simulation.Approximate algorithm results are shown only for the comparison of QAM and AQAM since RA had hardly any benefit from approximation, and QFM will follow the same trends as QAM but with slightly higher depth numbers.

1) Noiseless Simulation: The first set of tests used the ideal (noiseless) simulation to assess the impact of removed rotations in the final outcome of the multiplication algorithm. Since RA and QFM exact implementations have 100% accuracy, and their approximate implementations show no relevant information in this case as explained above, they are not included in this discussion. The expected result is that AQAM will display reduced accuracy as the size of the operands increases due to circuit simplifications. The results of these simulations for square multiplication can be seen in Figure 7. Under ideal conditions, the QAM remains 100% accurate regardless of bit width while the accuracy of the AQAM begins to decline as more qubits are added. When the inputs reach a qubit size of seven, the accuracy is 74%. This accuracy loss will be less pronounced in cases where not every bit in the input is high, so this case can be seen as the worst case accuracy for the given AQAM implementation. Still, it should be noted that this means the correct output had a 74% probability in the histogram, being the most probable outcome by far when compared to all other outputs —all of them adding up to the remainder of 26% for all the other outputs that are not correct ($2^{14}-1$ possible outputs in the 7×7)—. If the reduction in ideal accuracy becomes too large, the approximation scheme can be adjusted to allow smaller phase shifts to occur. This will bring the AQAM closer to the performance of the QAM.

A comparison of the accuracy of QAM and AQAM implementations in the trivial $n \times 1$ multiplication case is shown in Figure8. While there is still a reduced accuracy in the AQAM, it has a noticeably lower loss than the square multiplication. This can be attributed to the fact that the identity cases have lower depth, and to the lower number of phase shift activations that are performed in the trivial multiplication.

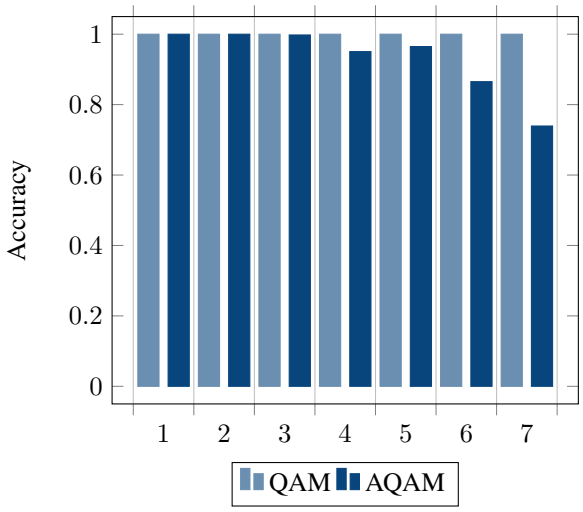


Fig. 7: Noiseless simulation of square ($n \times n$) multiplication with QAM and AQAM. As with all of the other algorithms, when in a noiseless environment, QAM has 100% accuracy for any given input width. AQAM sees a reduction in accuracy as the input size increases since it is removing an increasingly large number of phase shifts from being executed. AQAM's lowest accuracy is 74%.

2) Noisy Simulation: Each of the different multiplication algorithms were run under noisy simulated environments to observe changes in accuracy with increasing input width. The first set of simulations for the square ($n \times n$) case, performed with the default GenericBackendV2 noise model, is shown in Figure 9. The loss of accuracy is dramatic in this case. It can be seen that, while RA implementation starts with the most accuracy, it is the first to reach near 0% accuracy at four qubits. The trend aligns with the depth graph in Figure 4, starting better than QAM and QFM, but quickly deteriorating. QFM does eventually perform better than RA, but it remains worse than both QAM and AQAM implementations. Finally, comparing the performance of QAM and AQAM, they are almost identical in accuracy, with AQAM having slightly better accuracy once the input's width surpasses three qubits. Despite the loss in precision due to the design simplifications in the approximate case, the accuracy of AQAM still surpasses

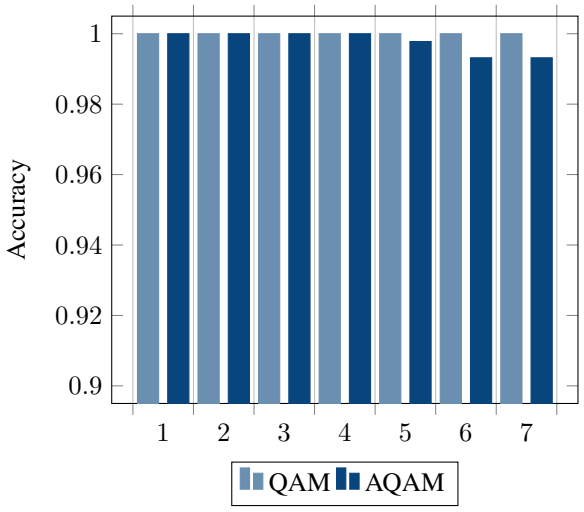


Fig. 8: Trivial $n \times 1$ QAM vs AQAM accuracy simulation results in a noiseless environment. As in the square multiplication case, the QAM remains 100% accurate while the AQAM begins to fall, but in this case, the accuracy reduction of the AQAM is less pronounced due to its lower count of gates removed.

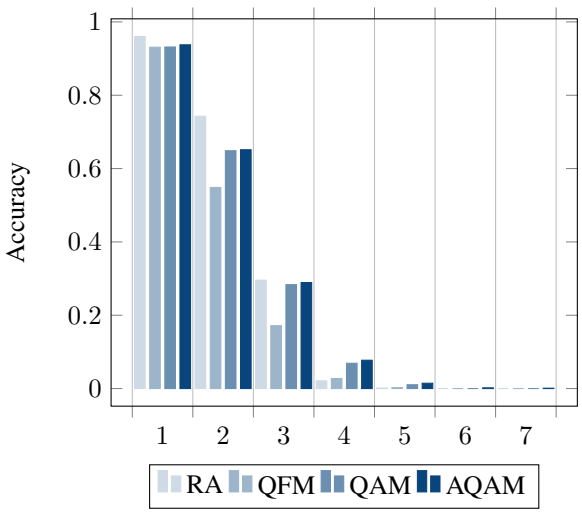


Fig. 9: Square ($n \times n$) noisy simulation showing the accuracy of each algorithm given an increasingly large input size. While initially the most accurate for an input width of one, the RA quickly falls below the others. In the larger input cases, it follows the sequence QFM, QAM, and AQAM from less to most accurate. In any case, the loss of accuracy is dramatic and the results are inconclusive for four or more input qubits.

QAM thanks to the reduced depth and consequent reduced overall noise. This confirms that in the presence of noise, there can be a benefit in sacrificing algorithmic precision for the sake of depth and overall accuracy. The overall accuracy is, in any case, extremely poor. For example, measuring the correct multiplier outcome for the four qubit input case results in only 0.077 probability for the best case (AQAM) and 0.021 for the worst (RA), and 0.69 QAM. AQAM is 7$\times$ better than RA. Despite approximations, AQAM is 1.1 $\times$ better than QAM. It should be noted that the individual probabilities under uniform superposition of an output with $4 + 4 = 8$ qubits are $\frac{1}{2^8} \approx 0.016$, which makes these probabilities still higher than uniform superposition values, but inconclusive in the real setting. As it will be shown in Figure 11, when noise is reduced, this accuracy can scale to show better results.

Using the same noise model, the trivial $n \times 1$ case was tested for accuracy, in order to get the other side of the spectrum, where a lower number of small rotations are removed from the design. Figure 10 depicts these results. RA has the best accuracy. Since only one addition is required, the positive impact in depth reduces the impact of noise on the final state, as expected. QAM and AQAM are both extremely similar as they have nearly the same depth for small input sizes, with AQAM having higher accuracy once it reaches an input width of five qubits. This is concurrent with the depth results shown previously in Figure 5. Again, the removed rotations pay off, thanks to the better depth behavior. For seven qubits, the probabilities of the correct outcome being measured are 0.46 (RA), 0.30 (AQAM) and 0.24 (QAM).

To demonstrate the benefits of approximate implementations as devices evolve towards lower noise levels, Figure 11 displays the behavior of higher-quality simulated devices with reduced noise levels. It is easier to see the potential benefit of the approximation technique in these results as the AQAM performs noticeably better than the original implementation of QAM. Both the QAM and AQAM outperform the other two algorithms when more than four qubits are utilized in the input qubit size. For the largest input (7 qubits), RA

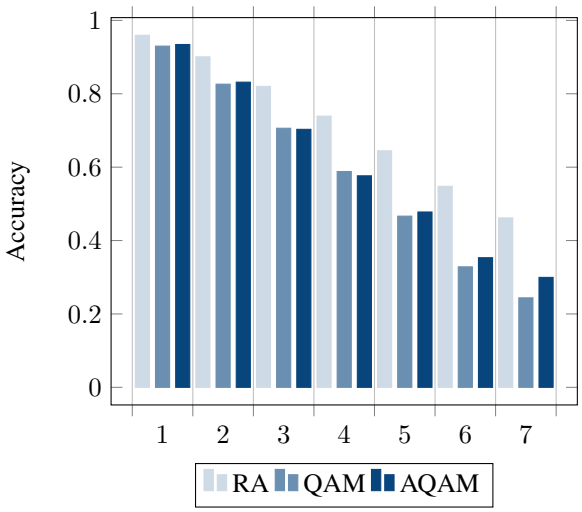


Fig. 10: Trivial ($n \times 1$) multiplication accuracy on a noisy system for RA, QAM, and AQAM. RA outperforms both the QAM and AQAM in all cases. (QFM cannot be tested when the two inputs are not equal in size.)

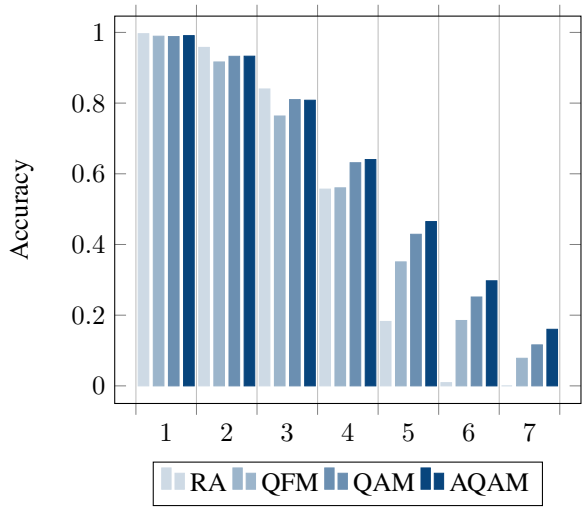


Fig. 11: Square ($n \times n$) reduced noise simulation which uses lower noise than the default backend. In this case, the distinction between the algorithms is more apparent, with the RA still falling at a faster rate than the others even though it starts at the highest accuracy. Of the other algorithms, in terms of worst-to-best accuracy, they appear as QFM, QAM, and AQAM.

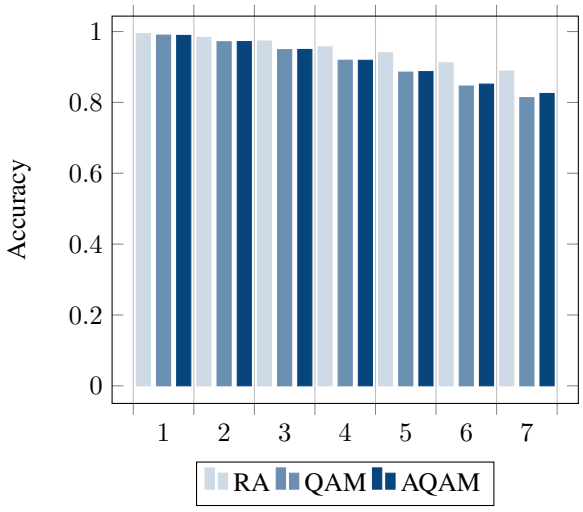


Fig. 12: Trivial ($n \times 1$) multiplication accuracy in the reduced noise setting for RA, QAM and AQAM. Similar to the noisy simulation, RA remains the best-case algorithm for trivial multiplication taking advantage of having only one iteration required for the multiplication. (QFM cannot be tested when the two inputs are not equal in size.)

did not produce the correct output (0 probability), while the others showed probabilities 0.07 (QFM), 0.11 (QAM), and 0.16 (AQAM). One thing to note is that the relative accuracy when comparing QAM and AQAM grows from one size to the next: 1.01, 1.09. 1.2 and 1.5 for four, five, six and seven qubit inputs respectively. The square noisy test case shown in Figure 9 showed a similar trend but with much lower accuracy. Approximation shows higher benefit as the problem size grows, from the higher benefit in depth reduction (Figure 4).

In this same setting, the trivial ($n \times 1$) multiplication was tested. Figure 12 reflects these results. The accuracy of these simulations is much higher than all the other noisy cases, due to the reduced depth of small-size multiplications and reduced noise levels. In accordance with the depth numbers depicted in Figure 5, RA results in more accurate outcomes for all input cases. Again, the benefits of AQAM begin to show at an input size of five, where QAM is just slightly less accurate than AQAM. At no point does the accuracy cross the 80% boundary for any of these final sets of experiments: for the seven qubit case, results were 0.88, 0.81 and 0.83 for RA, QAM, and AQAM respectively.

V. Conclusion

In this paper, we discuss the implementation of the Approximate Quantum Array Multiplier (AQAM) and compare it against other quantum array multipliers. Efficient quantum multipliers take advantage of the ability to encode information in the phase domain and operate with it through rotations. The approximate implementation relies on the fact that the outcome expressed as a probability distribution has tolerance to removing small rotations, under a certain threshold. This was mathematically justified in the previous work by Barenco *et al.* [10]. The end goal of this approach is to reduce the depth of the design. This paper applied this approach to Repeated Addition (RA) and Quantum Array Multiplier (QAM). The Quantum Fourier Multiplier (QFM), implemented within the Qiskit toolset, is also checked for reference, but not tested under approximation. One interesting takeaway is exposed by RA and its poor behavior under approximation. From a depth perspective, ARA does not show any advantage. It is important to notice that if the removed rotation steps occur in parallel with other non-reducible steps, then no overall reduction in gate-layers is observed, as it is the case for ARA. AQAM on the other hand shows a reduction in the number of gate-layers significant enough to improve the accuracy of the computation under noisy conditions, despite the approximation. As the depth of the problem increases, so does the benefit in accuracy compared to QAM.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997. [Online]. Available: http://dx.doi.org/10.1137/S0097539795293172

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Annual ACM Symposium on Theory of Computing*. ACM, 1996, pp. 212–219.

[3] A. Crimmins, S. L. Alarcon, M. Klein, M. Krebs, and S. Kate, "Quantum array multiplier," in *2023 IEEE International Conference on Rebooting Computing (ICRC)*, 2023, pp. 1–9. [Online]. Available: https://ieeexplore.ieee.org/document/10386449

[4] S. Anagolum. Arithmetic on quantum computers: Multiplication. [Online]. Available: https://medium.com/@sashwat.anagolum/arithmetic-on-quantum-computers-multiplication-4482cdc2d83b

[5] M. Sohel, N. Zia, M. Ali, and N. Zia, "Quantum computing based implementation of full adder," 11 2020, pp. 1–4.
[6] V. Vedral, A. Barenco, and A. Ekert, "Quantum networks for elementary arithmetic operations," *Phys. Rev. A*, vol. 54, pp. 147–153, Jul 1996. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.54.147
[7] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," 2004. [Online]. Available: https://arxiv.org/abs/quant-ph/0410184
[8] T. G. Draper, "Addition on a quantum computer." [Online]. Available: http://arxiv.org/abs/quant-ph/0008033
[9] S. Sajadimanesh, J. P. L. Faye, and E. Atoofian, "Practical approximate quantum multipliers for NISQ devices," in *Proceedings of the 19th ACM International Conference on Computing Frontiers*, ser. CF '22. Association for Computing Machinery, pp. 121–130. [Online]. Available: https://dl.acm.org/doi/10.1145/3528416.3530244
[10] A. Barenco, A. Ekert, K.-A. Suominen, and P. Törmä, "Approximate quantum fourier transform and decoherence," *Physical Review A*, vol. 54, no. 1, pp. 139–146, 1996. [Online]. Available: http://arxiv.org/abs/quant-ph/9601018
[11] "Qiskit: An open-source framework for quantum computing," 2021. [Online]. Available: https://qiskit.org/
[12] L. Ruiz-Perez and J. C. Garcia-Escartin, "Quantum arithmetic with the Quantum Fourier Transform," vol. 16, no. 6, p. 152. [Online]. Available: http://arxiv.org/abs/1411.5949