Investigating the Complexity of Various Quantum Incrementer Circuits

Presented by: Carlos Manuel Torres Jr.
Mentor: Dr. Selman Hershfield

Department of Physics
University of Florida
Gainesville, FL

Abstract:

Implementing Shor's Algorithm with quantum circuits involves many complex operations such as addition and modular exponentiation. The variety and complexity of the quantum gates involved are tremendous. The quantum incrementer is the simplest nontrivial quantum circuit which exhibits basic arithmetic operations such as addition, carry propagation, and the reset of carry, or auxiliary, bits. These operations are ubiquitous in quantum computing; thus, an investigation of this simplest case, the quantum incrementer, offers fundamental insight which can be used to build more complicated circuits.

An analysis of various 4-bit incrementer circuits is presented, and is generalized to the n-bit case. Three incrementer circuit topologies are derived and compared based on their total number of gates, the number of carry or auxiliary bits utilized, and the complexity of the types of gates used. A general method is derived to decompose complicated circuits into simpler circuits which are easier to manage and physically implement. Due to the cancellation of intermediate unitary gates, it is shown that adding auxiliary bits slightly increases the complexity of a given circuit by the order of 3n, which pales in comparison to the complexity of the original circuit of the order n^2.

I. Introduction:

The fundamental logic unit in classical computation theory is the bit, which has two distinct states represented by 0 or 1. The counterpart of the bit for a quantum computer is the qubit, which can be in a quantum state of $|0\rangle$, $|1\rangle$, or a complex linear superposition of these two orthonormal computational basis states: $\alpha|0\rangle + \beta|1\rangle$, where $|\alpha|^2+|\beta|^2 = 1$. Qubits can be prepared in superposition states allowing for massive parallel quantum information processing. Quantum computing reduces the complexity classes of various known algorithms from computer science. Harnessing the power of superposition introduces a significant speedup advantage over even the best classical supercomputers, thus prompting a threat to world-wide security cryptosystems.

Quantum Information Theory indicates that information is physical, thus motivating various experimental and theoretical physicists and engineers to develop a coherent two-state quantum system which would model the qubit. Perhaps the greatest limiting factor to realizing a quantum computer today is the issue of decoherence. Decoherence is any interaction between a two-state quantum system and its ambient environment which would diminish the coherence, or purity, of these states and ultimately destroy the qubit's superposition property. Once the superposition of a qubit is jeopardized, we result with nothing more than a classical bit.

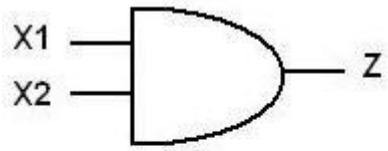
Qubits are prepared in some input register and then passed through a quantum circuit. A quantum circuit consists of various quantum logic gates which are essentially unitary transformations operating on these qubits [1]. Afterwards, the qubits can be read at the output register. Two basic quantum gates are the controlled-NOT, or C-NOT, and the controlled-controlled-NOT, or C^2-NOT, otherwise known as the Toffoli gate. A wire denotes the path a qubit travels throughout the circuit. Note that this is not a physical wire, but actually represents the time evolution of a specific qubit throughout the system.

Digital logic relies on classical gates which are irreversible and non-unitary such as the AND, OR, and several variations of these. This presents a problem when considering quantum computing

because we cannot simply map classical circuits to construct a quantum processor. AND, OR, and similar irreversible gates do not exist in quantum computing because quantum mechanics is by nature unitary. Probability is conserved in quantum computing whereas it is not in classical computing.

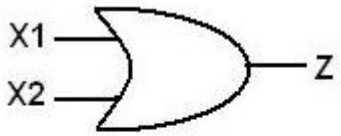
Figure 1 shows a 2-bit AND gate and its truth table. The AND operation has a true outcome only when both of the input bits are true; every other permutation of input bits results in a false outcome. Figure 2 shows a 2-bit OR gate and its truth table. The OR operation exhibits three possible true outcomes and only one false outcome. If both of the input bits are false, then the outcome is false; otherwise, the outcome is true.

Since there exists multiple input permutations resulting in the same outcome, we cannot distinguish which input permutation caused the specific outcome. This lack of one-to-one correspondence mapping between domain (input) and range (output), that is, one input for one output, illustrates the non-unitary and irreversible nature of classical computing. This can be easily remedied in quantum computing by adapting auxiliary "work" bits which present extra inputs or outputs to classical gates, thus making them unitary by having the same number of input bits as output bits.



X1	X2	Z
0	0	0
0	1	0
1	0	0
1	1	1

Figure 1. The AND gate and its corresponding truth table.



X1	X2	Z
0	0	0
0	1	1
1	0	1
1	1	1

Figure 2. The OR gate and its corresponding truth table.

The following reversible gates are widely used in quantum computation. Note that they are reversible and unitary because they have equal number of inputs as outputs. C-NOT is a conditional operation. It performs a NOT operation on the target qubit, x_2, if and only if the controlled qubit, x_1, is in state 1; otherwise, it acts as the identity operator on the target qubit [2].

$$\text{C-NOT:} \qquad |x_1\rangle|x_2\rangle \rightarrow |x_1\rangle|x_1 \text{ XOR } x_2\rangle$$

The symbol for the C-NOT gate is shown in Figure 3:

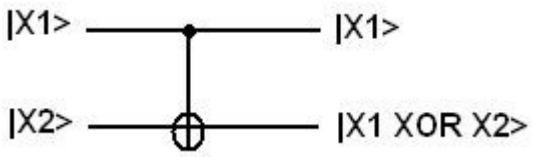


Figure 3. C-NOT gate; invert target qubit only if control qubit is active.

C-C-NOT is an extension of the C-NOT gate. It performs a NOT operation on the target qubit, x_3, if and only if the both control qubits, x_1 and x_2, are in state 1; otherwise, it acts as the identity operator on the target qubit [3].

$$\text{C-C-NOT:} \qquad |x_1\rangle|x_2\rangle|x_3\rangle \rightarrow |x_1\rangle|x_2\rangle|x_3 \text{ XOR } (x_1 \text{ AND } x_2)\rangle$$

The symbol for the Toffoli or C-C-NOT gate is shown in Figure 4:

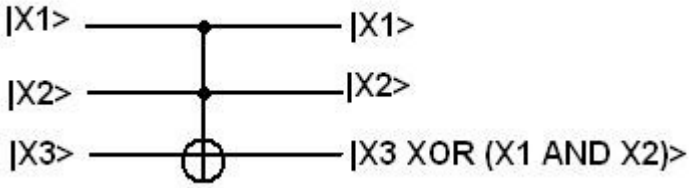


Figure 4. Toffoli gate; invert target qubit only if both control qubits are activate.

Figure 5 shows how a circuit can be reduced by deleting certain auxiliary bit lines [4].

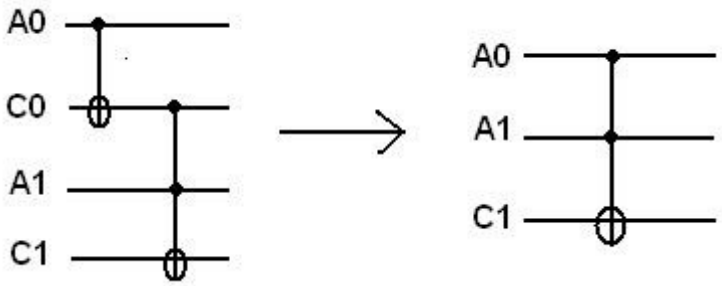


Figure 5. Two equivalent circuits. The C_0 carry bit is deleted since it mimics the A_0 bit. C_0 and A_1 must be in state 1 in order to operate on C_1. A_0 is implied to be in state 1 since C_0 must be in state 1 for C_1 to be determined. We refer to this as a "mimicking effect."

In order to operate on the C_1 carry bit, C_0 and A_1 must be in state 1. If C_0 is in state 1, this implies that A_0 must also be in state 1. Thus the intermediate C_0 carry bit mimics the behavior of its preceding A_0 bit and can be deleted without altering the overall operation of the circuit. The circuit on the left, consisting of a C-NOT and Toffoli gate is reduced to the circuit on the right, consisting of only a Toffoli gate.

The notation used in this paper to distinguish the various circuits follows the format:

(N-bit number:M-carry bits:Reset Carry bits En/Disabled)

where the particular circuits analyzed are the (4:0), (4:3:RE), (4:3:RD), and the (4:1:RE) circuits. The (4:0) circuit is a 4-bit incrementer with no carry bits. The (4:3:RE) circuit is a 4-bit incrementer with three carry bits which are reset upon reaching the next state. The (4:3:RD) circuit is a 4-bit incrementer with three carry bits which are not reset upon reaching the next state. Finally, the (4:1:RE) circuit is a 4-bit incrementer with one carry bit which is reset upon reaching the next state.

II. Discussion:

"4-Bit Incrementer, No Carry Bits (4:0)" Circuit:

The first circuit is a simple 4-bit incrementer without any carry bits. It is depicted in Figure 6. We represent the current state of the 4-bit number by: $A_3A_2A_1A_0$ and its corresponding next state by: $A_3^+A_2^+A_1^+A_0^+$. The current state is the 4-bit number located at the input register before incrementation occurs and the next state is the resulting number located at the output register after incrementation is

performed.

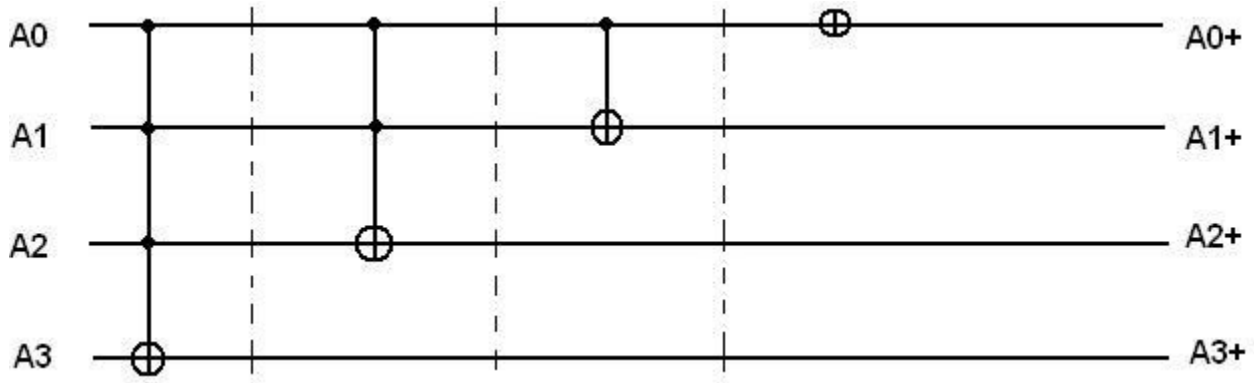


Figure 6. Circuit layout for the 4-bit incrementer without any carry bits (4:0). Note that the dashed vertical lines show how many stages are implemented. Each stage consists of various operations which set the current most-significant bit to its next state value.

Time evolution progresses from left to right across a quantum circuit. The current state of the 4-bit number resides in the input register. During the first stage, we encounter a C^3-NOT gate. This gate performs a NOT operation on the A_3 input bit if and only if input bits A_2, A_1, and A_0 are all in state 1; otherwise, it passes the input bit, A_3, towards the output register. This resulting value of A_3, located at the output register, is the next state value, A_3^+.

During the second stage, we encounter a Toffoli gate. This gate performs a NOT operation on the A_2 input bit if and only if bits A_1 and A_0 are both currently in state 1; otherwise, it passes the input bit, A_2, towards the output register, resulting in the next state value, A_2^+. The third stage consists of a C-NOT gate. This gate performs a NOT operation on the A_1 input bit if and only if the A_0 bit is currently in state 1; otherwise, it allows the input bit, A_1, to head towards the output register, resulting in the next state value, A_1^+. The fourth and final stage concerns a NOT gate. This gate simply inverts the A_0 input bit unconditionally, thus setting the next state value, A_0^+ at the output register.

The next state logic equations for the four stages of this "4-bit, no carry" circuit are:

$$A_3^+ = A_3 + A_2*A_1*A_0$$
$$A_2^+ = A_2 + A_1*A_0$$
$$A_1^+ = A_1 + A_0$$
$$A_0^+ = /A_0$$

Consider the following example utilizing the above logic equations for this circuit:

Let $A_3A_2A_1A_0 = 0111$ be the current state residing in the input register. The various gate operations are performed on this current state value as follows:

$$A_3^+ = A_3 + A_2*A_1*A_0 = 0 + 1*1*1 = 0 + 1 = 1$$
$$A_2^+ = A_2 + A_1*A_0 = 1 + 1*1 = 1 + 1 = 0$$
$$A_1^+ = A_1 + A_0 = 1 + 1 = 0$$
$$A_0^+ = /A_0 = /1 = 0$$

The result $A_3^+A_2^+A_1^+A_0^+ = 1000$ is the next state which resides in the output register. Note that the value at the input register was incremented by 1 and sent to the output register.

"4-Bit Incrementer, 3 Carry Bits, Reset Enabled (4:3:RE)" Circuit:

The second circuit is a simple 4-bit incrementer utilizing three carry bits. It is depicted in Figure 7. This circuit implements auxiliary or dummy bits which temporarily store carry values from previous lower-bit addition stages. These carries propagate throughout the various addition stages and eventually set the current-stage most-significant bit to its next state value at the output register.

The current state of the 4-bit number, $A_3A_2A_1A_0$, is set at the input register. Notice that the carry bits, $C_2C_1C_0$, are always initialized to zero, and in this particular circuit they will be reset to zero at the output register as the next state: $C_2^+C_1^+C_0^+$. Thus, the input register stores the current state: $A_3C_2A_2C_1A_1C_0A_0$, while the output register stores the next state: $A_3^+C_2^+A_2^+C_1^+A_1^+C_0^+A_0^+$.

During the first stage, we encounter a C-NOT gate. This gate will invert the initial value of C_0 if and only if A_0 is set to 1; otherwise, it passes through the initial value of C_0. The next gate is a Toffoli gate, which inverts the initial value of C_1 if and only if the current states of C_0 and A_1 are both 1; otherwise, it passes through the initial value of C_1. Another Toffoli gate follows which inverts the initial value of C_2 if and only if the current states of C_1 and A_2 are both 1; otherwise, it passes through the initial value of C_2. The next gate is a C-NOT gate, which inverts the initial value of A_3 if and only

if the current state of C_2 is set to 1; otherwise, it passes through the initial value of A_3 to the output register and sets it as the next state value, A_3^+.

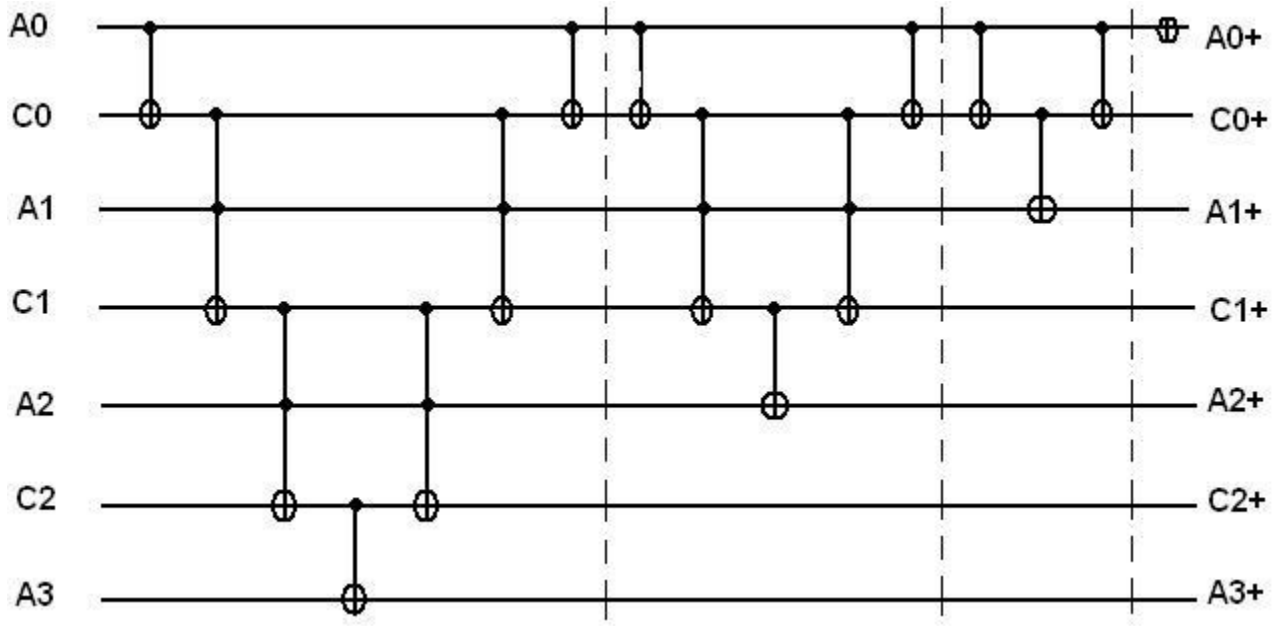


Figure 7. Circuit layout for the 4-bit incrementer with three carry bits and reset enabled (4:3:RE). Note that the dashed vertical lines show how many stages are implemented. Each stage consists of various operations which set the current most-significant bit to its next state value.

This particular circuit requires that the carry bits be reset upon their next state. Since the last operation was to determine the most-significant bit, A_3, we can now undo the carry propagations that led up to determining the next state of A_3 and thus reset the carry bits by performing reverse operations with each corresponding gate. This is possible because all of these quantum gates are unitary in nature. Unitary transformations satisfy the condition:

$$UU^{-1} = U^{-1}U = 1 \qquad (1)$$

Thus, after having determined the most-significant bit value which is sent to the output register, we reset the carry propagations by proceeding in reverse order and mirror-imaging each previous gate. Recall that C_1, A_2, and C_2 all had to be in state 1 for A_3 to be determined. Working backwards from A_3 and implementing a reverse-Toffoli gate yields:

$$C_2^+ = C_2 + A_2 * C_1 = 1 + 1 * 1 = 1 + 1 = 0$$

which resets the C_2 carry bit before it passes towards the next stage. Similarly, C_0 and A_1 must have both been in state 1 for C_1 to have propagated earlier. Operating with another reverse-Toffoli gate yields:

$$C_1^+ = C_1 + A_1 * C_0 = 1 + 1 * 1 = 1 + 1 = 0$$

which resets the C_1 carry bit before it enters the next stage. Finally, A_0 must have been in state 1 for C_0 to have propagated earlier. Operating with a reverse-C-NOT gate yields:

$$C_0^+ = C_0 + A_0 = 1 + 1 = 0$$

which resets the C_0 carry bit before it enters the next stage. All of these operations constitute the first stage of the incrementer associated with determining A_3, the most-significant bit.

Now we proceed to the second stage, in which a similar procedure is executed in order to determine the next most-significant bit, A_2, and send its next state value to the output register. The first gate we encounter is a C-NOT gate. This gate will invert the current state of C_0 (coming from the first stage) if and only if A_0 is set to 1; otherwise, it passes through the current state of C_0. The next gate is a Toffoli gate, which inverts the current state of C_1 (coming from the first stage) if and only if the current states of C_0 and A_1 are both 1; otherwise, it allows the current state of C_1 to pass through. The following gate is a C-NOT gate, which inverts the current state of A_2 if and only if the current state of C_1 is set to 1; otherwise, it passes through the current state of A_2 to the output register. Similar to the first stage, we operate with reverse-Toffoli and reverse-C-NOT gates:

$$C_1^+ = C_1 + A_1 * C_0 = 1 + 1 * 1 = 1 + 1 = 0$$
$$C_0^+ = C_0 + A_0 = 1 + 1 = 0$$

which reset the carry bits, C_1 and C_0, to zero before proceeding to the next stage.

We arrive at the third stage with the objective of determining the next most-significant bit, A_1. The first gate we encounter is a C-NOT gate. This gate will invert the current state of C_0 (coming from the second stage) if and only if A_0 is set to 1; otherwise, it passes through the current state of C0. The

next gate is another C-NOT gate. This gate will invert the current state of A_1 (coming from the second stage) if and only if C_0 is set to 1; otherwise, it passes through the current state of A_1 to the output register. We operate with a reverse-CNOT gate:

$$C_0^+ = C_0 + A_0 = 1 + 1 = 0$$

to reset the carry bit C_0 to zero before proceeding to the next stage. Finally, the fourth and final stage consists of a NOT gate, which simply inverts the current state of A_0 (coming from the third stage) and sends its next state value to the output register.

Figure 8 shows the (4:3:RE) circuit with the specific unitary gates to be annihilated. It is possible to annihilate the adjacent mirror-image unitary gates:

$$\text{C-NOT}*\text{C-NOT}^{-1} = 1$$
$$\text{C-NOT}*\text{C-NOT}^{-1} = 1$$
$$\text{TOFFOLI}*\text{TOFFOLI}^{-1} = 1$$

because of their unitary property (see Eq (1)). This reduced circuit (see Figure 7) exhibits fewer gates and results in a simpler and possibly faster circuit. Another simplification to this (4:3:RE) circuit after annihilation of mirror-image gates is that we could delete the two reverse-Toffoli gates if we are not concerned with resetting the carry bits after incrementation. This (4:3:RD) circuit is shown in Figure 8.

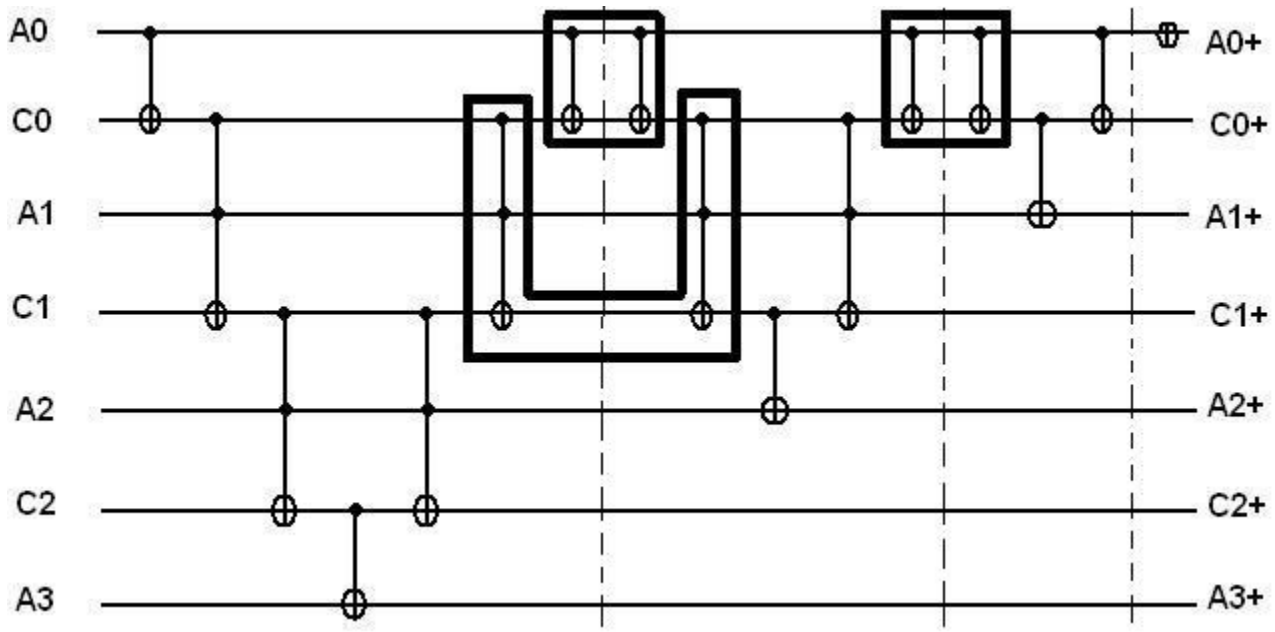


Figure 8. Circuit layout for the (4:3:RE) highlighting the unitary gates to be annihilated. Note that the dashed vertical lines show how many stages are implemented. Each stage consists of various operations which set the current most-significant bit to its next state value.

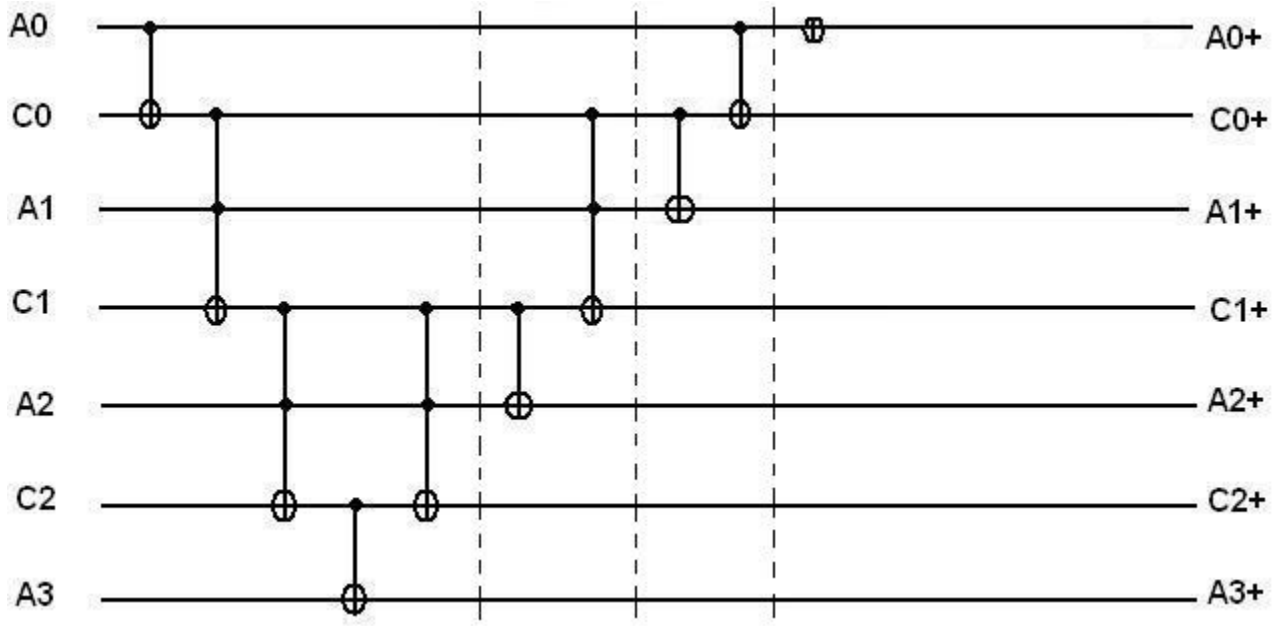


Figure 9. Circuit layout for the (4:3:RE) after annihilation of three specified sets of unitary gates. Note that the dashed vertical lines show how many stages are implemented. Each stage consists of various operations which set the current most-significant bit to its next state value.

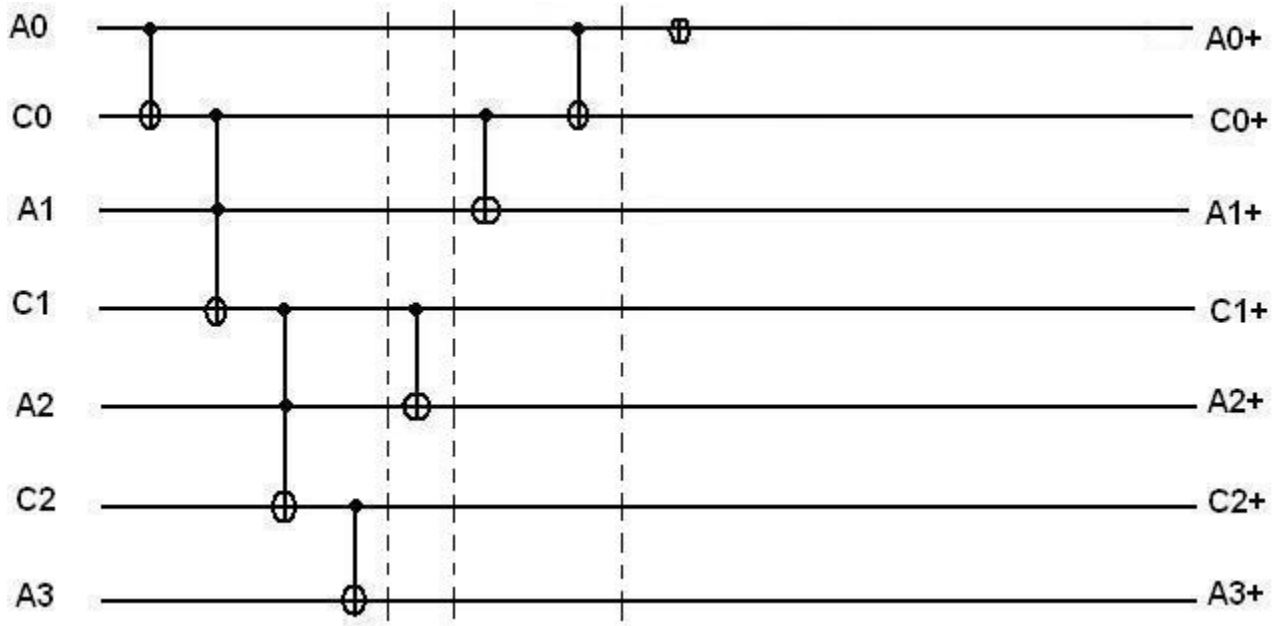


Figure 10. Circuit layout for the (4:3:RD) when the carry bits are not reset after incrementation. Note that the dashed vertical lines show how many stages are implemented. Each stage consists of various operations which set the current most-significant bit to its next state value.

"4-Bit Incrementer, 1 carry bit, Reset Enabled (4:1:RE)" Circuit:

The final circuit is a 4-bit incrementer with only one auxiliary carry bit. This one carry bit is reset upon reaching its next state at the output register. This circuit is depicted in Figure 11. It is essentially the (4:3:RE) circuit mentioned in Figure 9 with the C_2/C_2^+ and C_0/C_0^+ auxiliary lines deleted (refer to Figure 5 for the "mimicking effect").

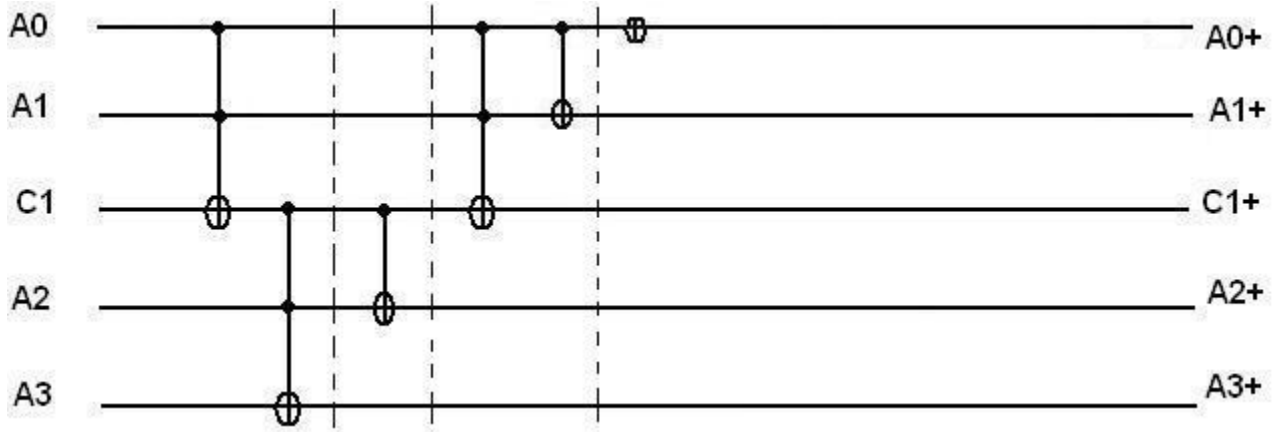


Figure 11. Circuit layout for the (4:1:RD) when two of the three auxiliary carry bits are excluded.

We have derived three main circuits which perform the same incrementation operation. These circuits are the (4:0) circuit in Figure 6, the (4:3:RE) circuit in Figure 9, and the (4:1:RE) circuit in

Figure 11.

Table I. Comparing the three main 4-bit incrementer circuit topologies.

Circuit	(4:0) [Figure 4]	(4:3:RE) [Figure 7]	(4:1:RE) [Figure 9]
Number of Gates	4	10	6
Number of Carry Bits	0	3	1
Types of Gates	(1) C^3-NOT (1) TOFFOLI (1) C-NOT (1) NOT	(4) TOFFOLI (5) C-NOT (1) NOT	(3) TOFFOLI (2) C-NOT (1) NOT

Table I compares these circuits in terms of their number of gates, whether carry bits are used, and their types of gates. The (4:3:RE) circuit qualifies as having the most number of gates, the (4:0) circuit has zero carries, and both the (4:3:RE) and (4:1:RE) circuits have the simplest types of gates: C-NOT, Toffoli, and NOT.

Generalization to the n-bit case (Proof and further elaboration in upcoming revision):

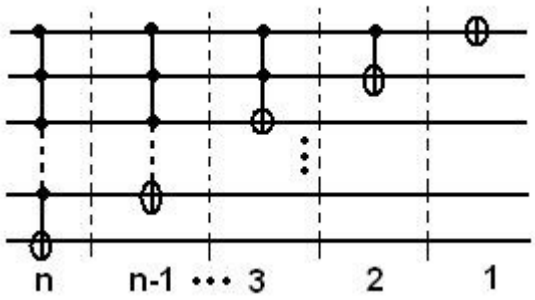


Figure 12. Generalization of n-bit/n-gate circuit.

Figure 12 shows the generalization to n-bits of the 4-bit incrementer circuit aforementioned in Figure 6. There are (n-1)-2 = n-3 auxiliary bits, where the (n-1) bit lines are sandwiched between the top and bottom bit lines. We don't include the top or bottom lines because they are never auxiliary bits but rather the least and most significant bits in a given circumstance. The auxiliary bits are found in between each of the (n-1) sandwiched bit lines. Thus, the number of auxiliary bits is the difference between the (n-1) sandwiched lines and the top and bottom bit lines.

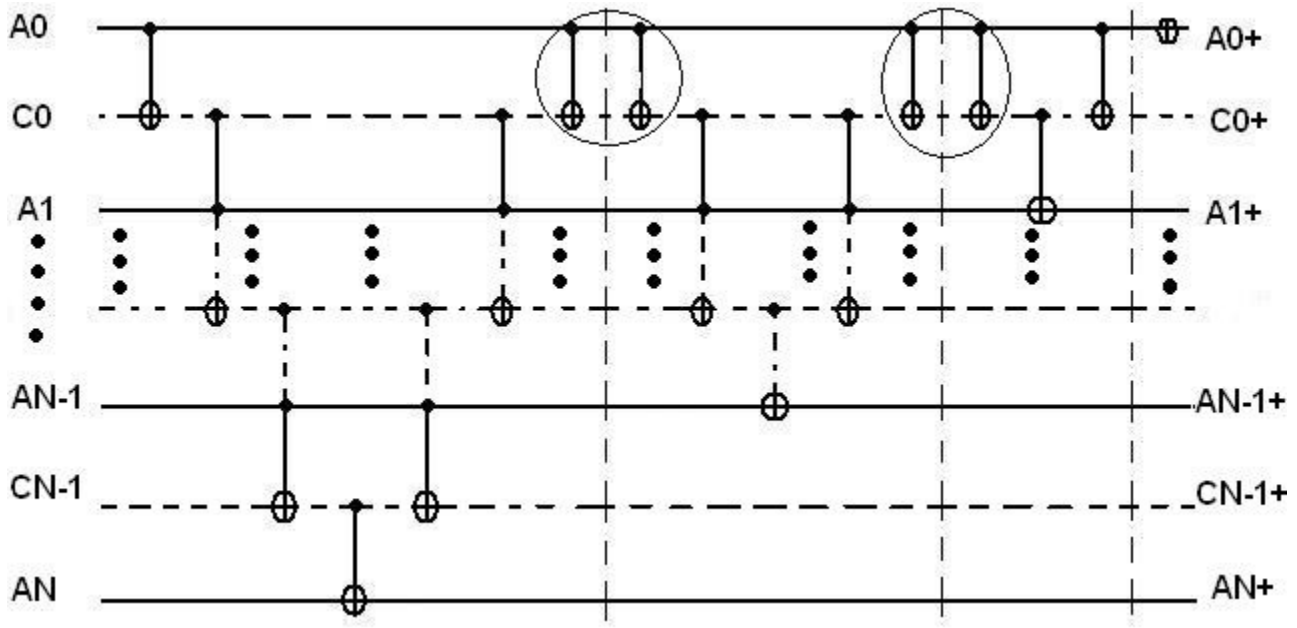


Figure 13. Generalization of n-bit incrementer with carry bits.

Figure 13 shows the generalized n-bit gate in terms of Toffoli, C-NOT, and NOT gates. It also outlines a few implicit cancellations of C-NOT gates. Recall that there are (n-3) auxiliary bits as we proceed towards determining the most-significant bit, A_N. As we reset the carry bits by proceeding backwards, we pass another (n-3) auxiliary bits.

At the end of the process, n bits are determined (ie. A_N, A_{N-1}, ..., A_1, A_0). Thus, the total number of gates added by using auxiliary bits sums to: $2(n-3) + n = 3n-6$. It is evident that using auxiliary bits increases the complexity of the initial circuit shown in Figure 13 by an order of 3n. The increase in complexity by using carry bits is not too costly. Recall that the original circuit has complexity on the order of n2.

Conclusion:

Three main quantum incrementer circuits were derived and analyzed: the (4:0) circuit, the (4:3:RE) circuit, and the (4:1:RE) circuit. These circuits differ in their number of quantum gates, whether carry bits were utilized, and in their types of gates. We learned that any one of these circuits can be mapped into three different topologies depending on one's particular design deliverables. These three main topologies were presented in Table I.

We commenced with the (4:0) incrementer circuit, which consists of high-order gates. These high-order gates involve many interacting qubits, which presents a formidable challenge to physically implement. One can reduce these high-order gates to simpler Toffoli and C-NOT gates, which are easier to manage and realize.

These 4-bit incrementers and 3-bit carries were generalized to n-bit incrementers and (n-3) bit carries, respectively. If the mimicking effect shown in Figure 3 is employed, further reductions in carry bits can be made.

Acknowledgments:

I am very grateful to Dr. Selman Hershfield for his patience and guidance throughout our interactive lectures and discussions this summer. I appreciate Dr. Kevin Ingersent and Kristin Nichola for having made this REU program a successful, informative, and enjoyable experience. I also thank the NSF for funding this REU program at the University of Florida. And above all, I thank God to whom I owe my love, my existence, and my motivation in this journey of a life which He continues to provide me.

References:

[1] N. D. Mermin, "Notes for physicists on the theory of quantum computation", informal notes for three lectures at LASSP Autumn School of Quantum Computation, Cornell, September 20, 22, and 24, 1999, www.lassp.cornell.edu/lassp_data/NMermin.html.

[2] A. Ekert, P. Hayden, and H. Inamori, “Basic Concepts in Quantum Computation”, arXiv:quant-ph/0011013v1

[3] A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J.A. Smolin, and H. Weinfurter, “Elementary Gates for Quantum Computation”, Phys. Rev. A, **52**, 3457-67, (1995).

[4] M. Mc Gettrick and B. Murphy, “Simulation of the CCC-Not Quantum Gate”, *Technical Report*

NUIG-IT-061002, Department of Information Technology, NUI, Galway

http://www.it.nuigalway.ie/publications/TR/Tr-sem.asp, 2002.