Resource Efficient Design of Quantum Circuits for Quantum Algorithms

Himanshu Thapliyal

Department of Electrical and Computer Engineering

University of Kentucky, Lexington, KY

hthapliyal@uky.edu

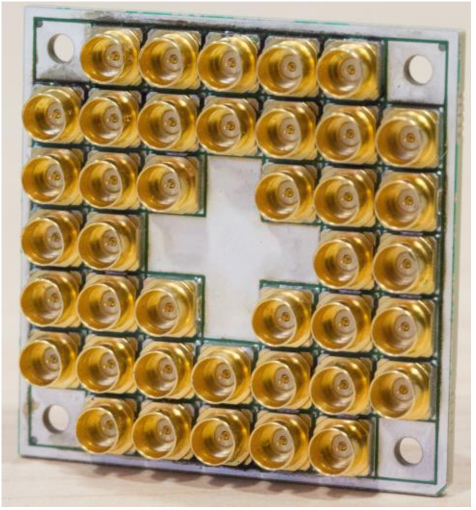
Quantum Circuits for Quantum Algorithms

- Shor's factoring algorithm and solving discrete log problem.
- Cryptanalysis which makes encryption and digital signature schemes such as RSA and Elliptic Curve Cryptography (ECC) vulnerable to quantum attacks.
- Quantum algorithms such as class number and triangle finding algorithms, and scientific algorithms in quantum chemistry.

Quantum computing offers big performance gains in number theory, encryption, search and scientific computation

Progress on Quantum Computing Processor

- **In April 2017, IBM 5 qubit quantum processor.**
- **In April 2017, Google 9 qubit computing chip.**
- **In May 2017, IBM 17 qubits quantum computing processor.**
- **In Oct 2017, Intel 17-qubit quantum computing test chip.**



Intel’s 17-qubit quantum computing chip
(Credit: Intel Corporation Source: newsroom.intel.com)

Motivation

- Quantum circuits for arithmetic operations are required to implement quantum algorithms
- Practical quantum circuits must be based on fault tolerant gates such as Clifford + T gates
- **Existing quantum computers have few qubits**

Challenges

- Must use fault tolerant quantum gates
- Reversibility introduces more circuit overhead
- Must minimize use of the quantum T gate
  - The T gate is costly to implement

Quantum gates used in this work

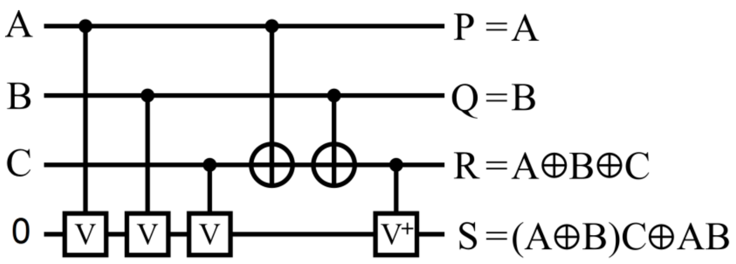
Type of Gate	Symbol	Matrix
Not gate	N	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Hadamard gate	H	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
T gate	T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i.\frac{\pi}{4}} \end{bmatrix}$
T gate Hermitian transpose	$T^\dagger$	$\begin{bmatrix} 1 & 0 \\ 0 & e^{-i.\frac{\pi}{4}} \end{bmatrix}$
Phase gate	S	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
Phase gate Hermitian transpose	$S^\dagger$	$\begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}$
Feynman gate	C	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

Quantum Circuit Optimization: Different from Classical

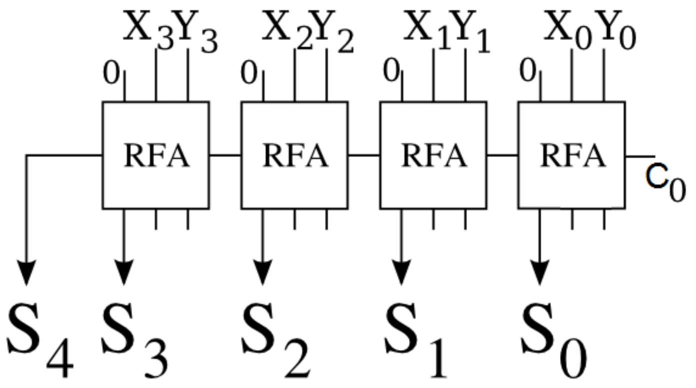
Full adder having inputs A,B,C where C is carry input.

$$Sum = A \oplus B \oplus C$$

$$Cout = AB \oplus ((A \oplus B).\ C)$$



1 bit Reversible Full Adder



4 bit Reversible Full Adder

Total Bits= 8 Input Bits+ 1Carry Bit+ 4 Ancilla Inputs

=13 bits

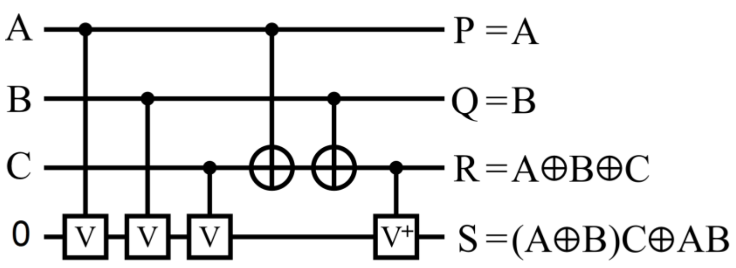
Suppose the best realizable quantum computer due to technology limitations had only 9 qubits.

New Design Methodology For Quantum Circuits

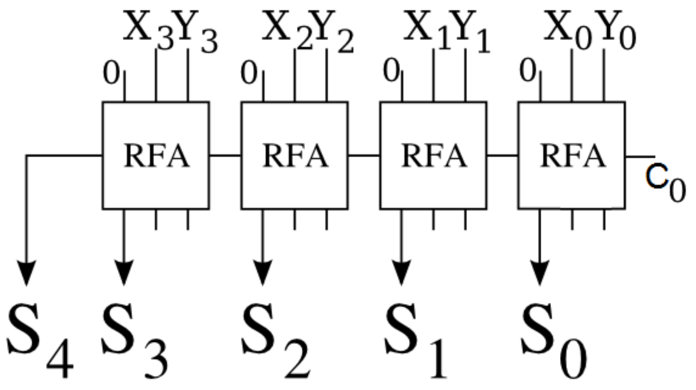
Full adder having inputs A,B,C where C is carry input.

Sum= A $\oplus$B $\oplus$C

Cout= AB $\oplus$ ((A $\oplus$ B). C)



1 bit Reversible Full Adder



4 bit Reversible Full Adder

4 bit adder has 4 Ancilla Inputs

The design of a n bit reversible adder based on the conventional ripple carry approach of cascading will need n ancilla inputs

Existing quantum hardware is limited in terms of number of available qubits . Thus, qubits needs to be kept to a minimum.

Design Methodology of Reversible Binary Adder

H. Thapliyal and N. Ranganathan, “Design of Efficient Reversible Logic Based Binary and BCD Adder Circuits”, ACM Journal of Emerging Technologies in Computing Systems, Vol.9, No.3, pp. 17:1–17:31, Sep 2013.

Reversible Binary Adder (Cont'..d)

Consider the addition of two n bit numbers **a_i and b_i** stored at memory locations **A_i and B_i**, respectively, where $0 \leq i \leq n-1$.

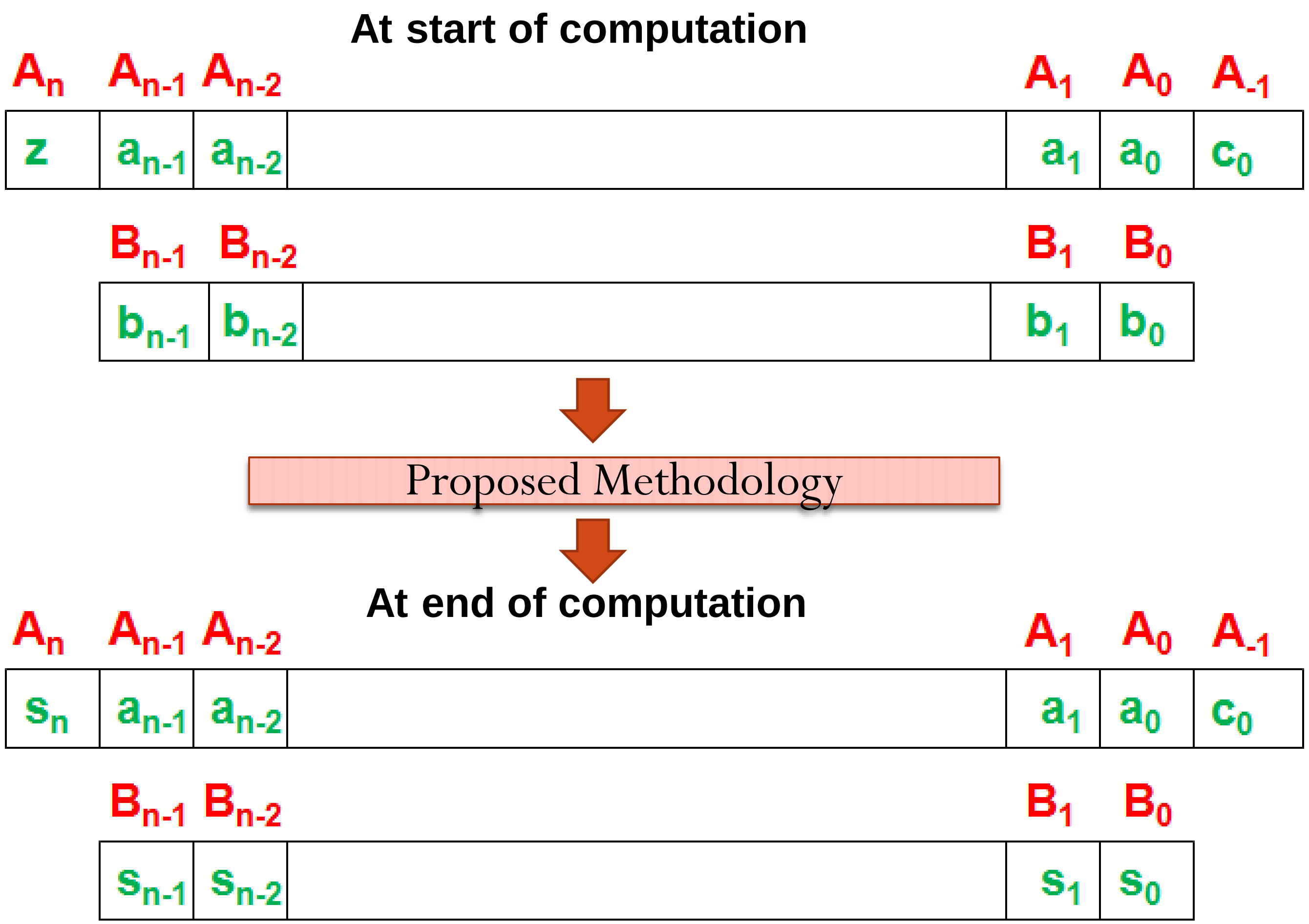
$$s_i = \begin{cases} a_i \oplus b_i \oplus c_i & \text{if } 0 \leq i \leq n-1 \\ c_n & \text{if } i = n \end{cases}$$

where c_i is the carry bit and is defined as:

$$c_i = \begin{cases} c_0 & \text{if } i = 0 \\ a_{i-1}b_{i-1} \oplus b_{i-1}c_{i-1} \oplus c_{i-1}a_{i-1} & \text{if } 1 \leq i \leq n \end{cases}$$

The **input carry c_0** is stored at memory location **A_{-1}.** Further, consider that memory location **An is initialized with $z \in \{0, 1\}$.**

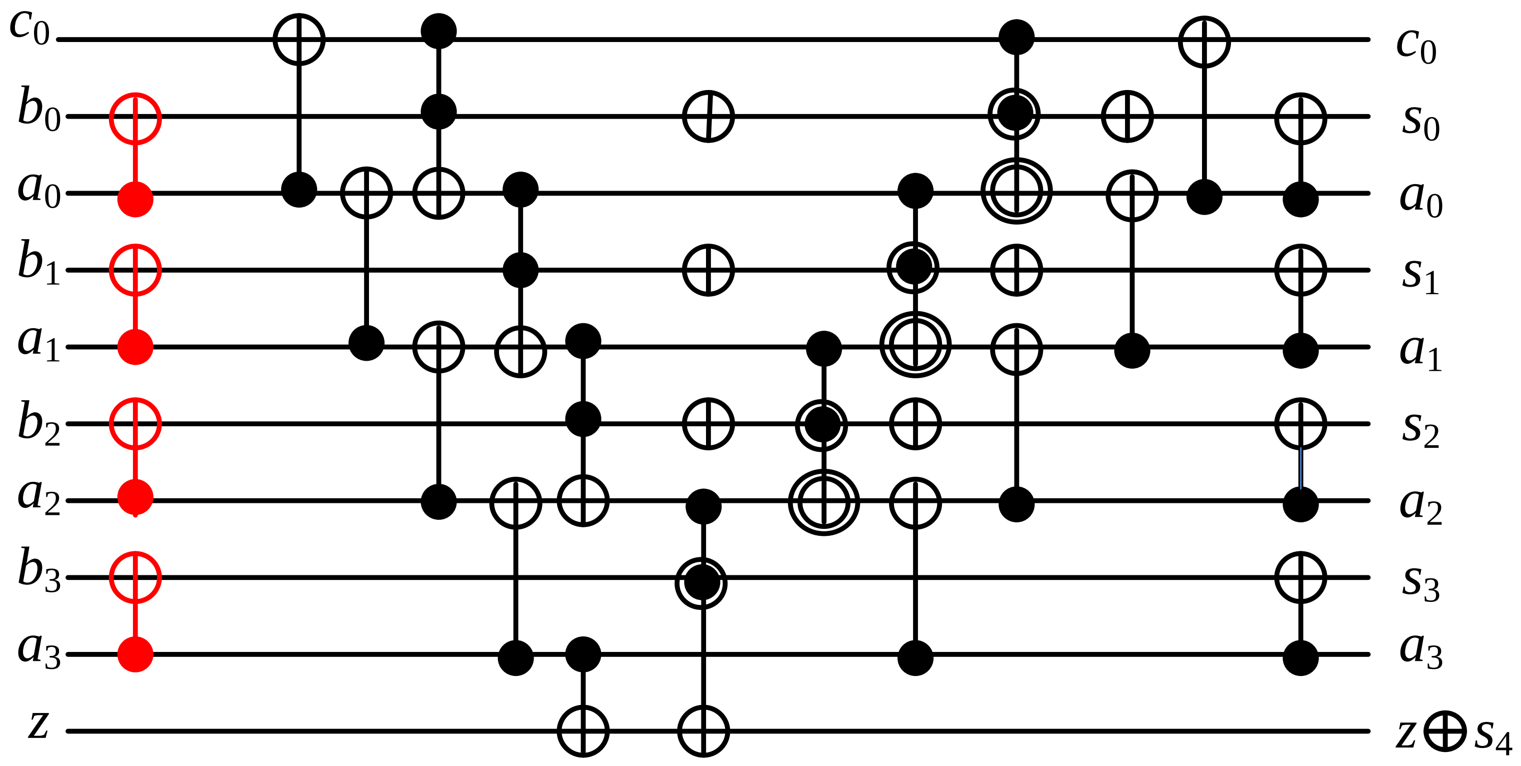
Reversible Binary Adder (Cont'..d)



Steps of Proposed Methodology

Step 1:

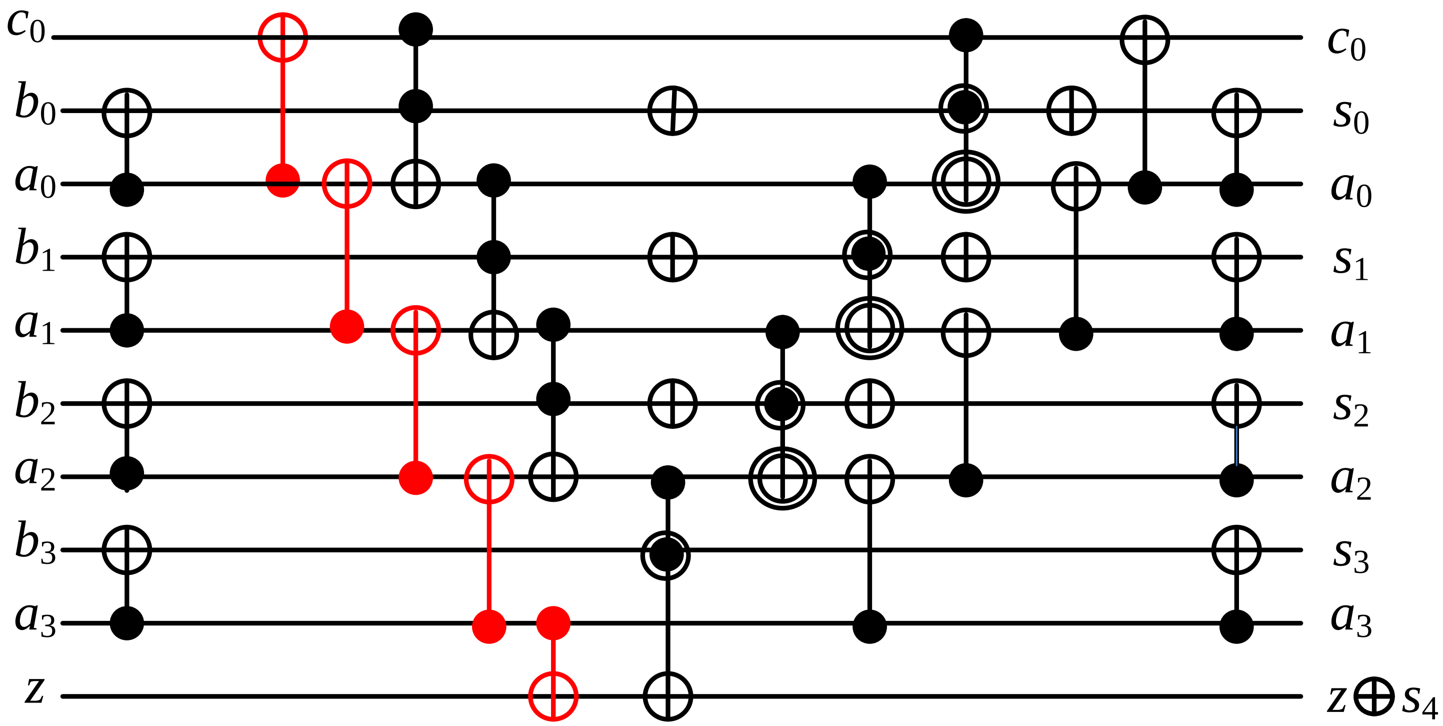
For i=0 to n-1: At pair of locations, A_i and B_i apply the CNOT gate.



Step 2:

→For i= -1 to n-2: At pair of locations A_{i+1} and A_i apply the CNOT gate.

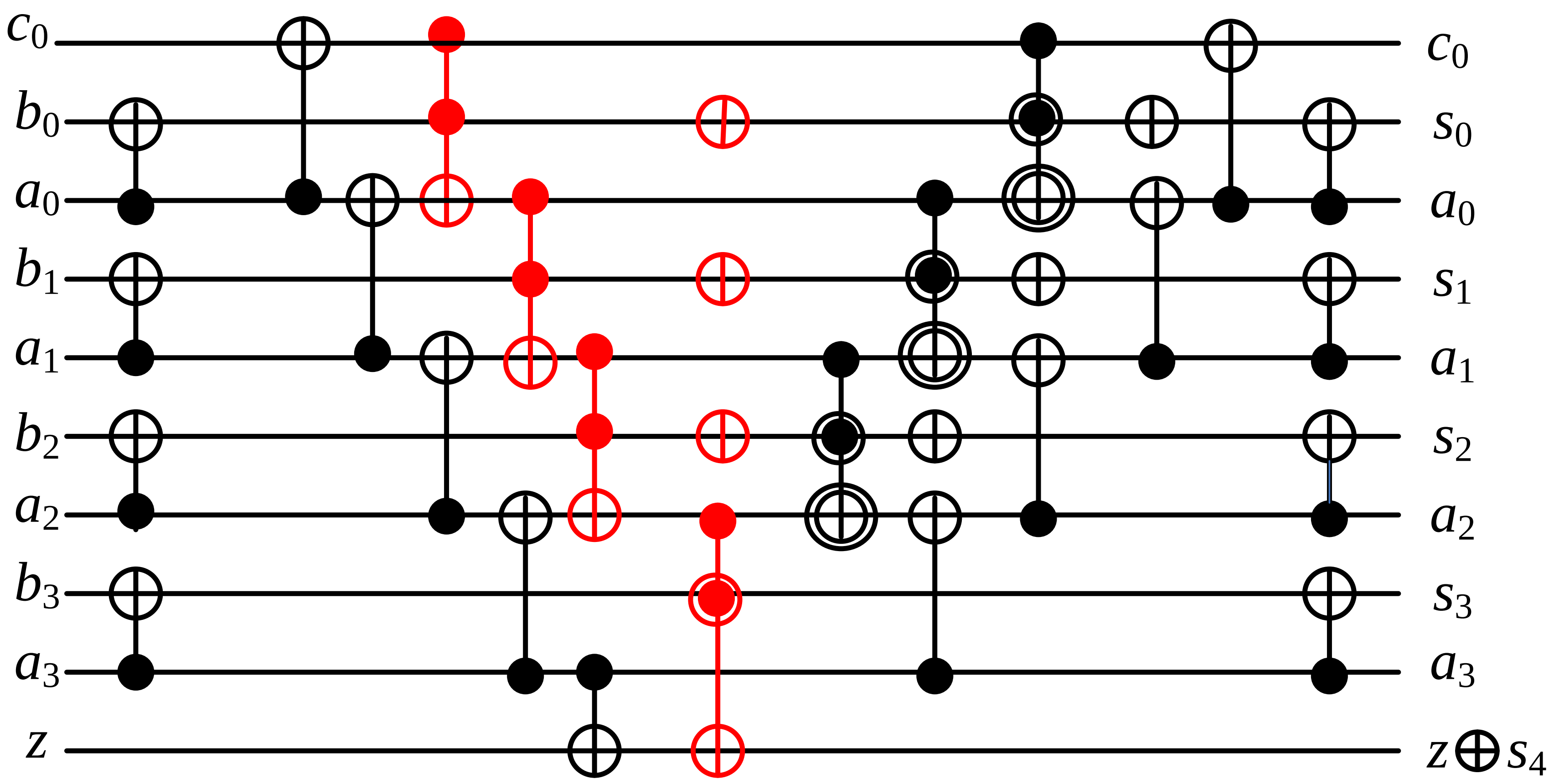
→Further, apply a CNOT gate at pair of locations A_{n-1} and A_n.



Step 3:

→ For i=0 to n-2: At locations A_{i-1}, B_i and A_i apply the Toffoli gate. Apply a Peres gate at location A_{n-2}, B_{n-1} and A_n,.

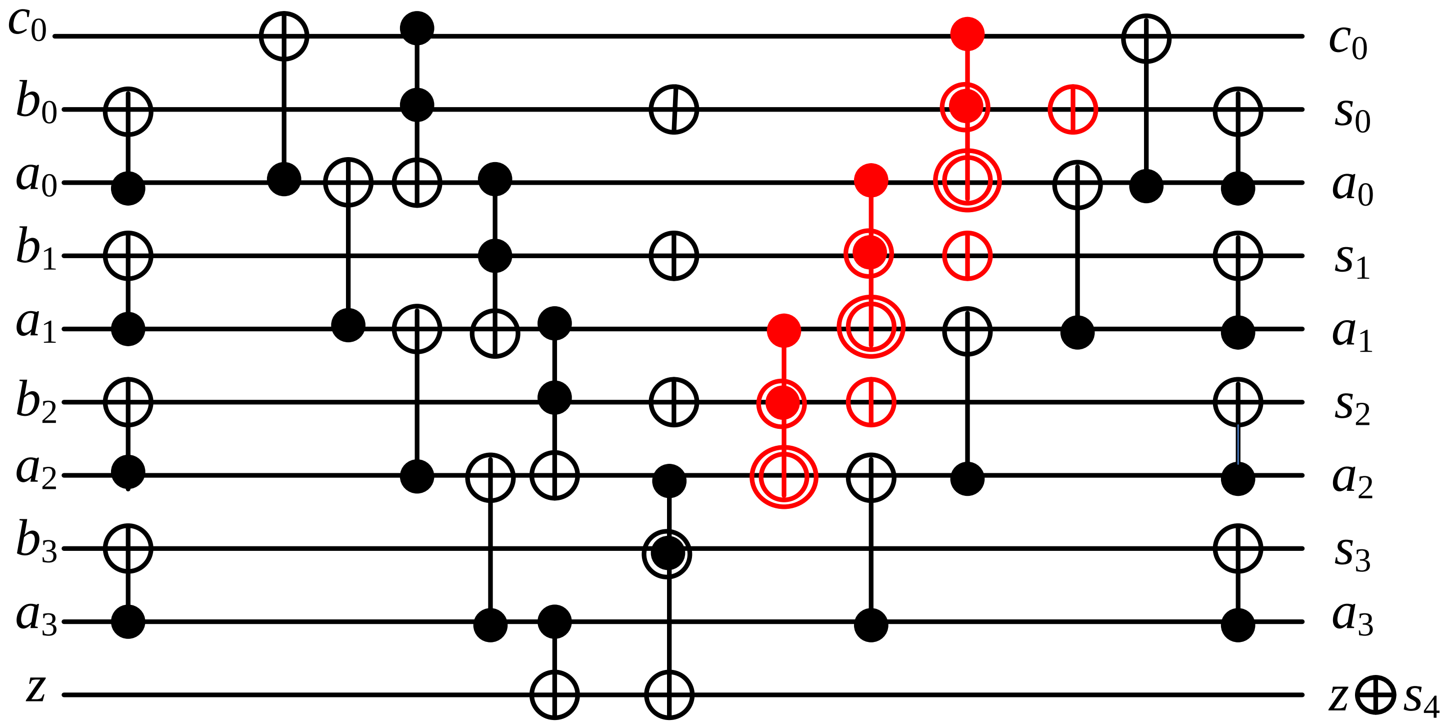
→ Further, for i=0 to n-2: Apply a NOT gate at location B_i.



Step 4:

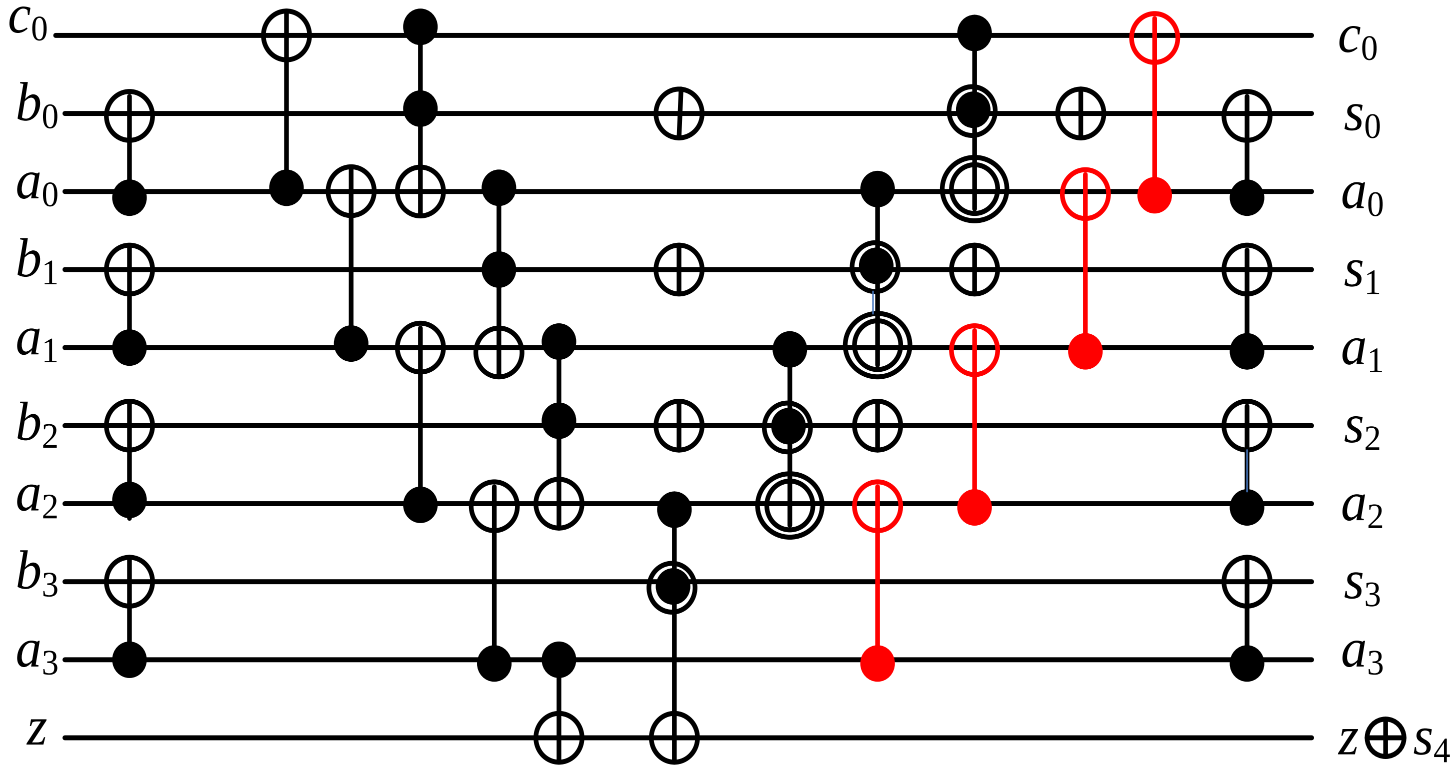
→For i=n-2 to 0: At locations A_{i-1}, B_i and A_i apply the TR gate.

→Further, For i=0 to n-2: Apply a NOT gate at location $B_{i.}$



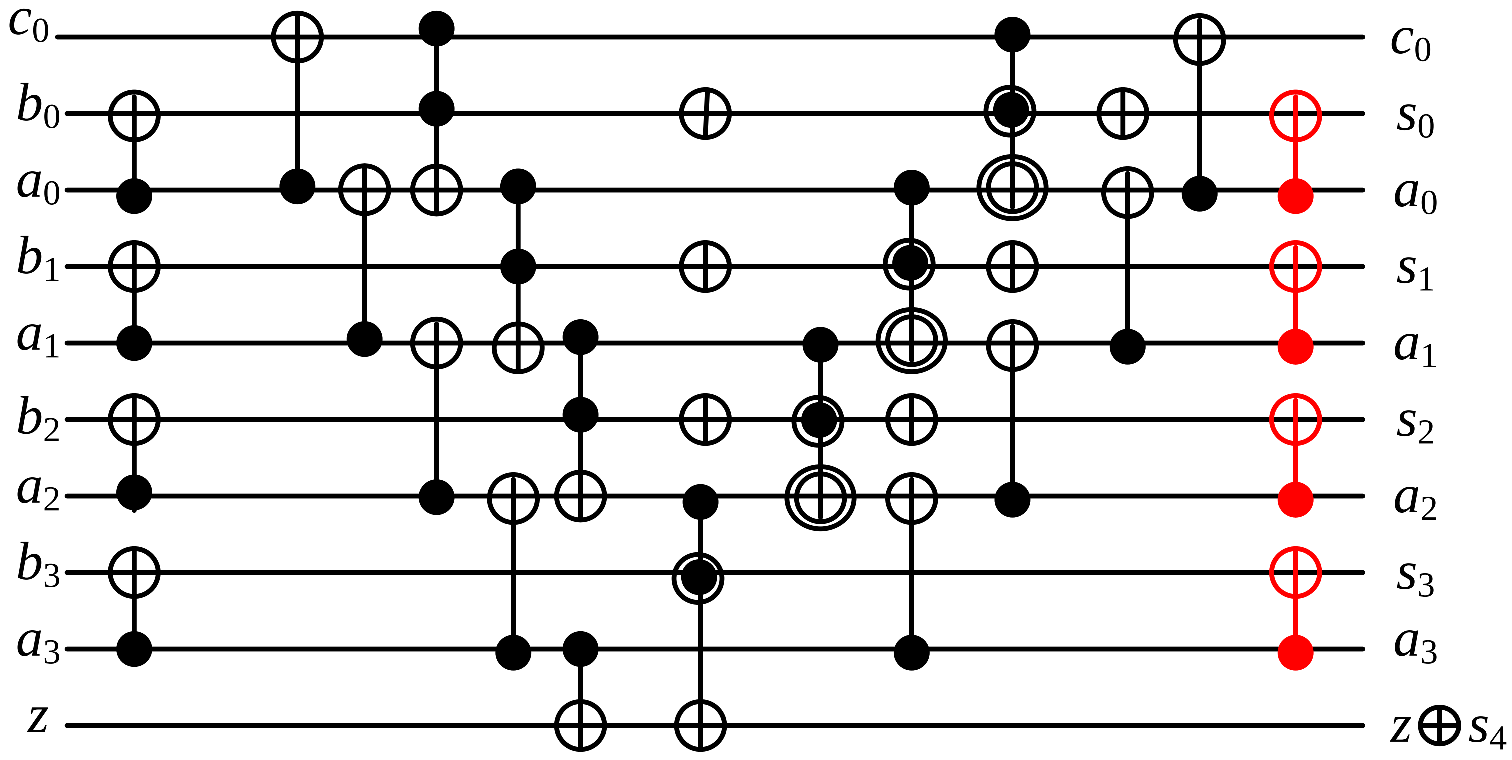
Step 5:

For i=n-1 to 0: At pair of locations A_i and $A_{i\text{-}1}$ apply the CNOT gate.



Step 6:

For i=0 to n-1: At pair of locations A_i and B_i apply the CNOT gate.



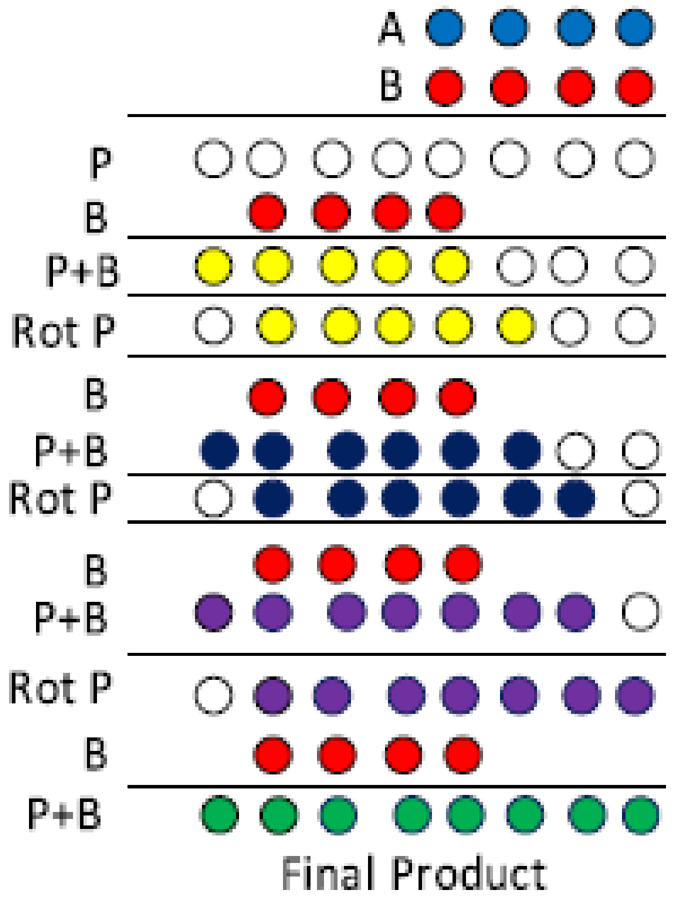
Proposed Multiplier Algorithm:

Add and Rotate

Algorithm 1 Add and Rotate method to model nxn Multiplier

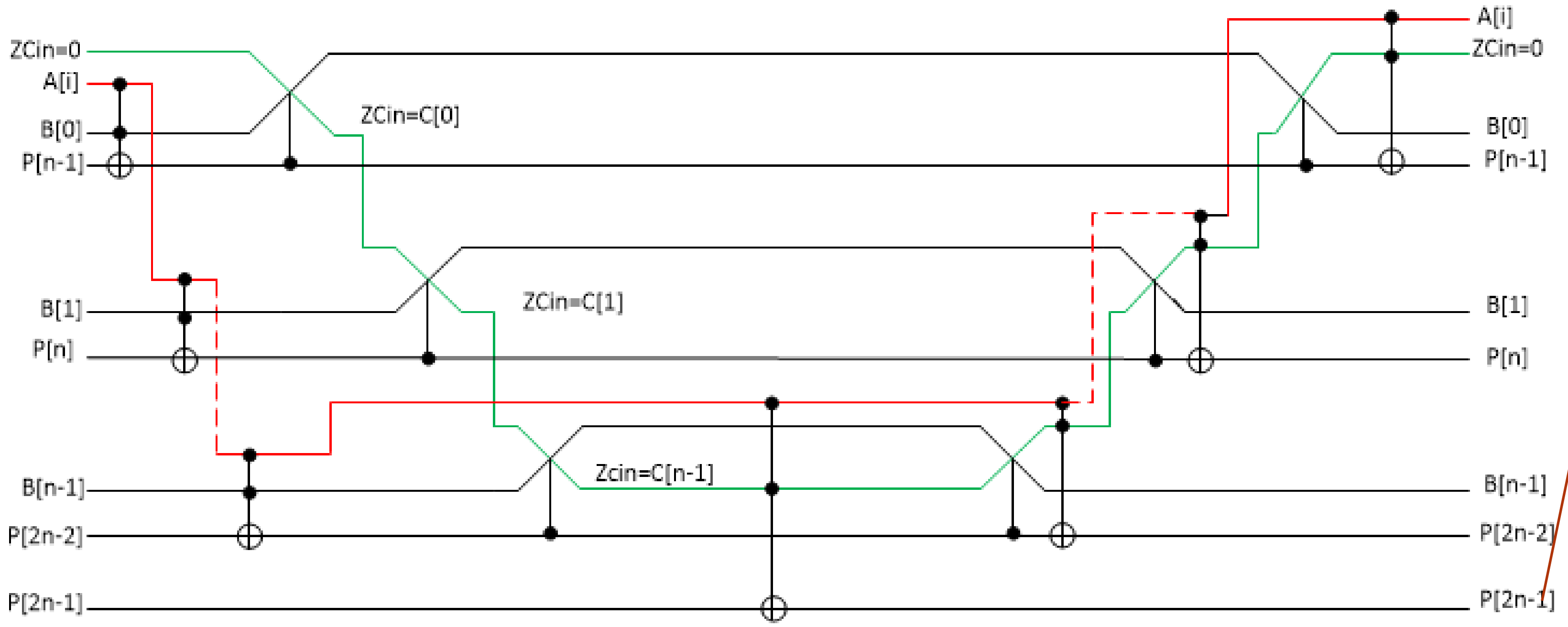
function MULTIPLIER($|A_n\rangle$, $|B_n\rangle$, $|P_n\rangle = |0_{2n}\rangle$)
  for $i = 0$ **to** $n - 2$ **do**
    if $|A_{[i]}\rangle = |1\rangle$ **then**
      $|P_{[2n-1:n-1]}\rangle = |P_{[2n-1:n-1]}\rangle + |B\rangle$;
    end if
    $|P_{[2n-1:0]}\rangle =$ ROTATERIGHT($|P_{[2n-1:0]}\rangle$);
  end for
  if $|A_{[n-1]}\rangle = |1\rangle$ **then**
    $|P_{[2n-1:n-1]}\rangle = |P_{[2n-1:n-1]}\rangle + |B\rangle$;
  end if
  return P;
end function

A-Multiplier-n qubit register
B-Multiplicand- n qubit register
P- 2n qubit Product register
i- classical parameter



H. V. Jayashree, H. Thapliyal, H. R. Arabnia, and V. K. Agrawal, "Ancilla-input and garbage-output optimized design of a reversible quantum integer multiplier," The Journal of Supercomputing, vol. 72, no. 4, pp. 1477–1493, Apr. 2016

Proposed ADD or NOP Quantum Circuit



Here P[2n-1] location holds final cout value if control pin(ie A[i]) is 1, otherwise it will retain its old cout value.

i	A[i]	P[j]	B[i]	ab xor P	C[i-1]	Sum	C[I] or Zcin	
0	1	0	0	0	0	0	0	B[i]
1		0	0	0	1	1	0	
2		0	1	1	0	1	0	C[i-1]
3		0	1	1	1	0	1	
4		1	0	1	0	1	0	C[i-1]
5		1	0	1	1	0	1	
6		1	1	0	0	0	1	B[i]
7		1	1	0	1	1	1	

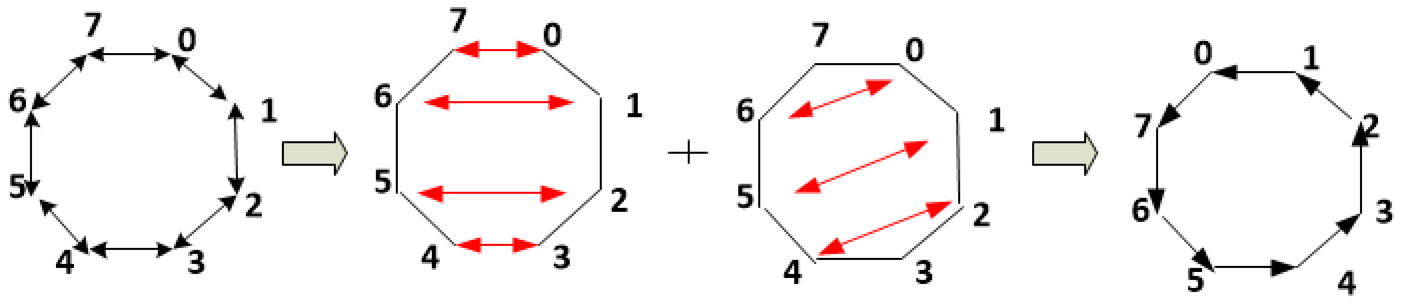
i	A[i]	P[j]	B[i]	AB xor P	C[i-1]	SUM	C[i] or Zcin	
0	0	0	0	0	0	0	0	B[i]
1		0	0	0	1	1	0	
2		0	1	0	0	1	0	C[i-1]
3		0	1	0	1	0	1	
4		1	0	1	0	1	0	C[i-1]
5		1	0	1	1	0	1	
6		1	1	1	0	0	1	B[j]
7		1	1	1	1	1	1	

- Zcin is the carry line which propagates the previous stage carry to next stage.
- If P xor (A. B) is =0 then Zcin is same as B[i] value else it is same as C[i-1] value.
- Here j is given by $i+n-1 \le j \le 2n-1$.

Thomsen, Glück and Axelsen, Journal of Physics, 2010

Rotate Right Operation: Example 8 bits

P[7]	P[6]	P[5]	P[4]	P[3]	P[2]	P[1]	P[0]



I stage

II stage

P[7]	P[6]	P[5]	P[4]	P[3]	P[2]	P[1]	P[0]
1	0	1	1	1	0	1	0

P[7]	P[6]	P[5]	P[4]	P[3]	P[2]	P[1]	P[0]
0	1	0	1	1	1	0	1

After stage I

P[7]	P[6]	P[5]	P[4]	P[3]	P[2]	P[1]	P[0]
0	1	0	1	1	1	0	1

After stage II

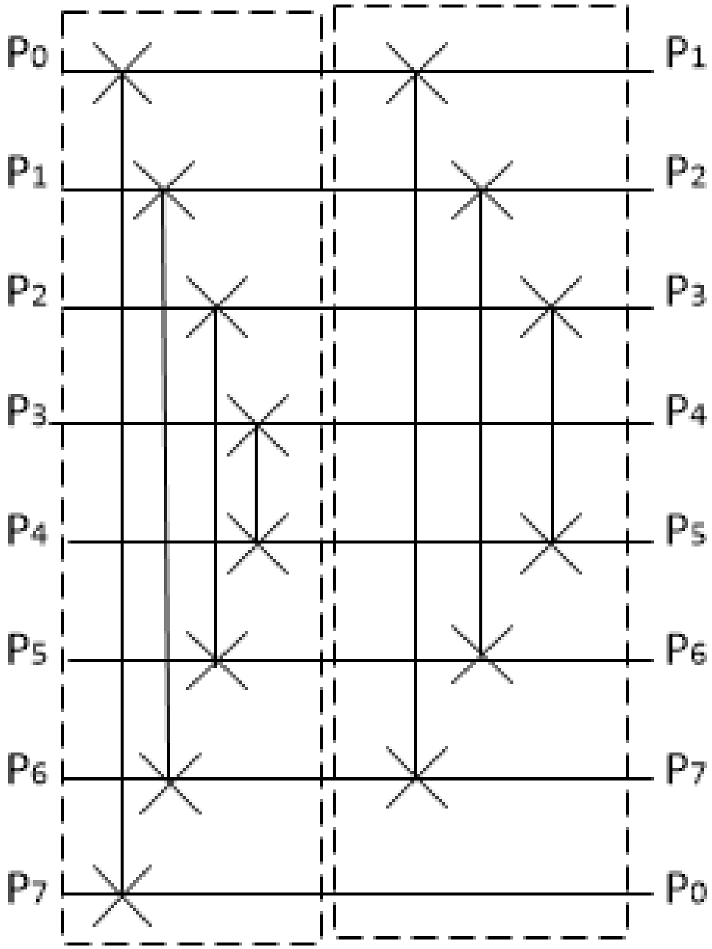
Moore, Cristopher, and Nilsson, *SIAM Journal on Computing* 31.3 (2001): 799-815.

Rotate Right-Reversible Circuit using Swap Gates

Algorithm 2 Pseudocode for rotate right operation

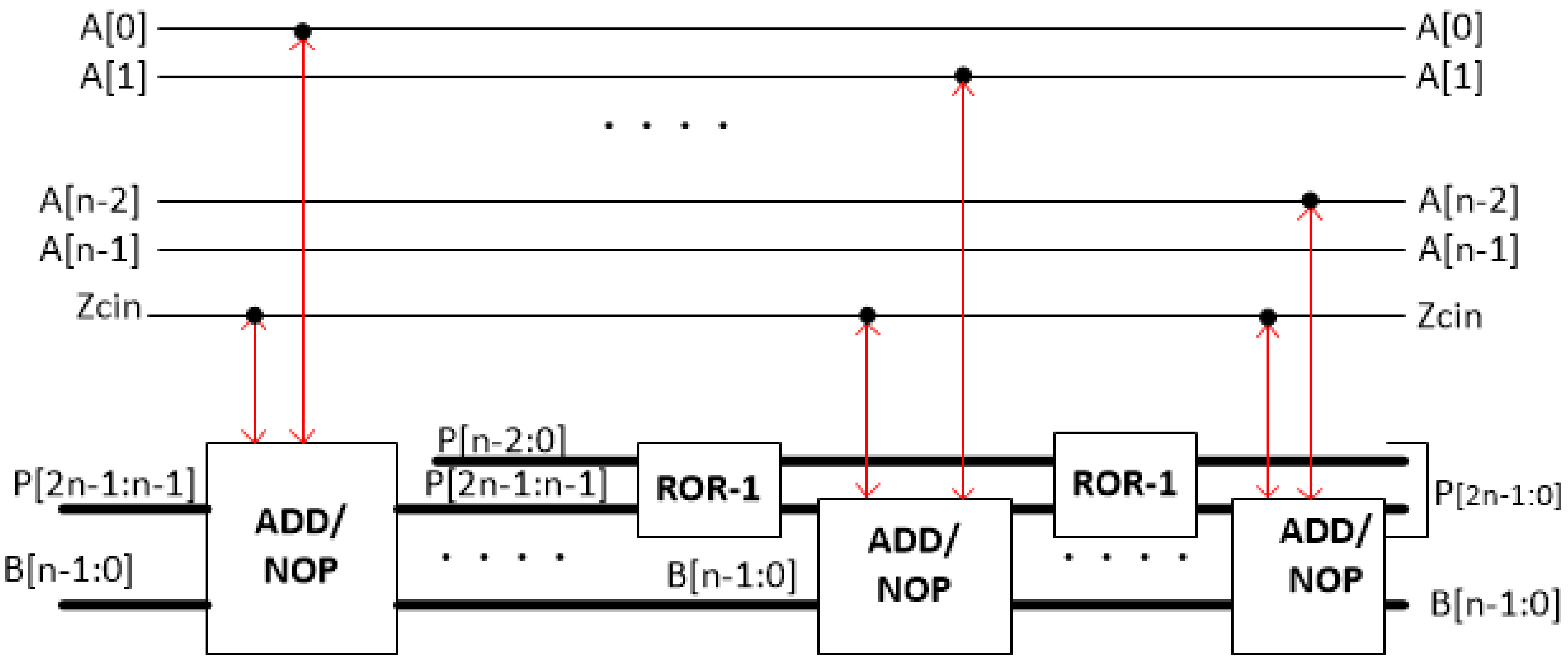
```
ROTATERIGHT(|P[2n−1:0]⟩)
k=SIZEOF(|P[2n−1:0]⟩);
if k mod 2 == 0 then                              ▷ For even number of bits
    i = 0; j=k − 1;
    while i < k/2 && j >= k/2  do                 ▷ First Stage
        SWAP(|P[i]⟩,|P[j]⟩);
        i = i + 1; j = j − 1;
    end while
    i = 0; j = k − 2;
    while  i < k/2 − 1 && j >= k/2  do            ▷ Second stage
        SWAP(|P[i]⟩|P[j]⟩);
        i = i + 1; j = j − 1;
    end while
else                                              ▷ For odd number of bits
    i = 0; j = k − 1;
    while i < k/2 && j >= k/2 + 1 do              ▷ First Stage
        SWAP(|P[i]⟩,|P[j]⟩)
        i = i + 1; j = j − 1
    end while
    i = 0; j = k − 2;
    while i < k/2 && j >= k/2 do                  ▷ Second Stage
        SWAP(|P[i]⟩,|P[j]⟩)
        i = i + 1; j = j − 1;
    end while
end if
return P;
```

Rotate Right-Reversible Circuit using Swap Gates



- Shifter takes 2n-1 swap gates.(for nxn multiplier)
- Gates inside dash box works in parallel.
- Depth will be 2 Swap Gates
- Gives constant delay of 2x3=6 irrespective of n value.

Complete Reversible Circuit for nxn Multiplier



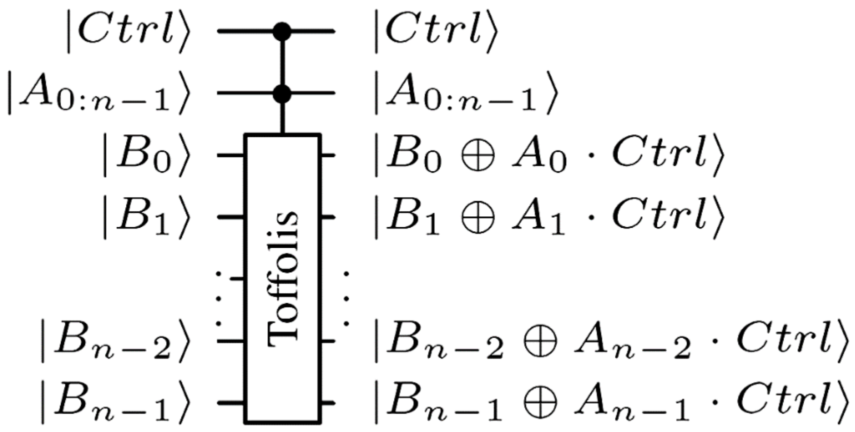
T-Count Optimized Quantum Circuits for Multiplication

- Implements: $A \cdot B$ via shift and add algorithm
- $A \cdot B = A \wedge B_0 + (A \wedge B_1) \cdot 2 + \cdots + (A \wedge B_i) \cdot 2^i + \cdots$
- Three step algorithm
- Multiplication circuit is based on specific components

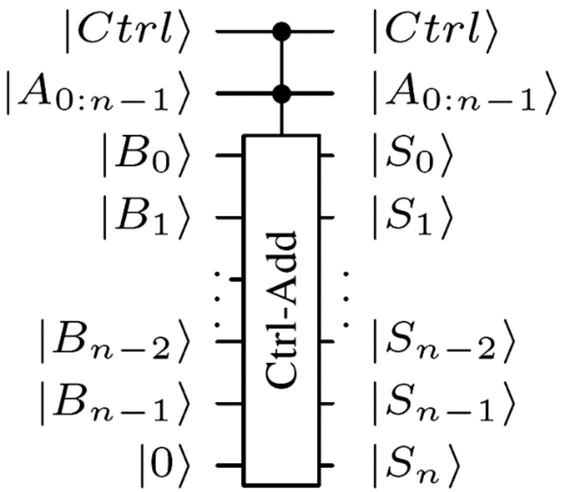
Will demonstrate quantum circuit design algorithm with two 4 bit numbers A and B

T Count Optimized Quantum Circuits for Multiplication

Components used in multiplication circuit



(a) Toffoli Gate Array

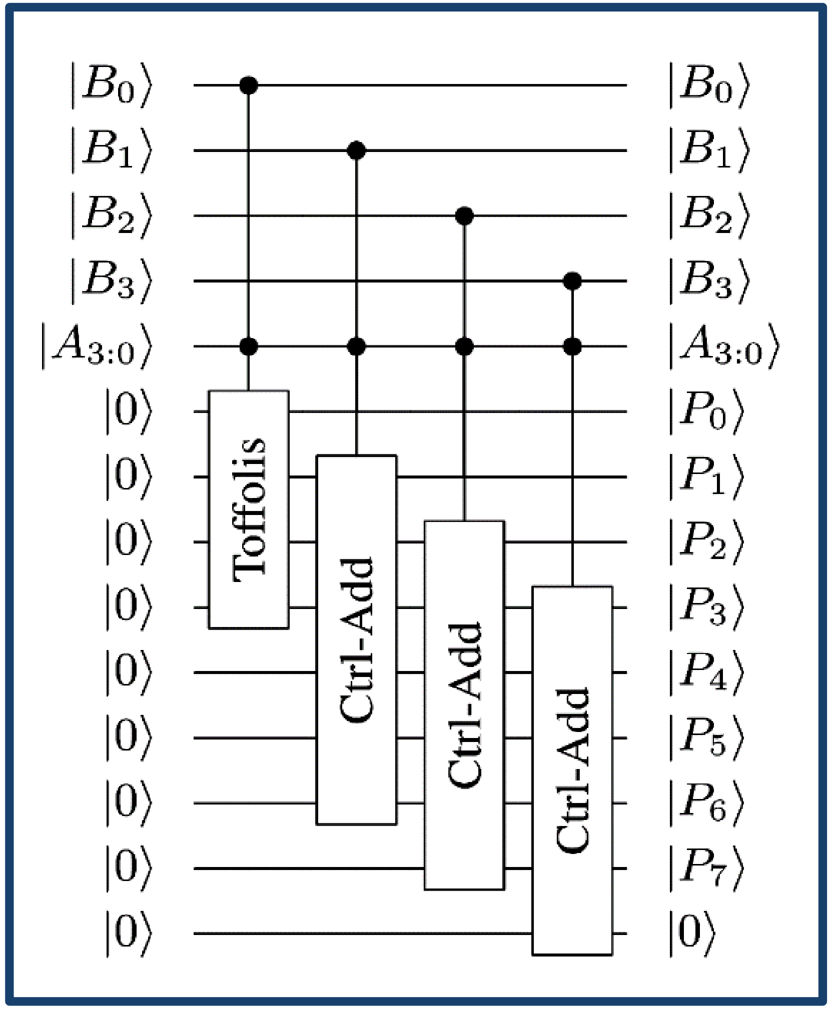


(b) Proposed quantum *Ctrl-Add* circuit with no input carry

Complete Multiplication circuit (T-Count Optimized)

- $|A\rangle$ and $|B\rangle$ are the inputs to be multiplied
- $|P\rangle$ is the product
- Circuit produces product by calculating:

$$\sum_{i=0}^{3} A \cdot B_i \cdot 2^i$$



Comparison of multiplication circuits

Comparison of quantum integer multiplication circuits

	1	2	3	Proposed
T-count	$56 \cdot n^2$	$28 \cdot n^2 + 7 \cdot n$	$42 \cdot n^2 - 42 \cdot n + 48$	$21 \cdot n^2 - 14$
qubits	$5 \cdot n + 1$	$4 \cdot n + 1$	NA	$4 \cdot n + 1$
ancillae	$3 \cdot n + 1$	$2 \cdot n + 1$	NA	$2 \cdot n + 1$

1 is the design by Lin et. al. [1]
2 is the design by Jayashree et. al. [2]
3 is the design by Babu [3] modified to remove garbage output.
Table entries are marked NA where a closed-form expression is not available for the design by Babu [3].

- [1] C.-C. Lin, A. Chakrabarti, and N. K. Jha, "Qlib: Quantum module library," J. Emerg. Technol. Comput. Syst., vol. 11, no. 1, pp. 7:1–7:20, Oct. 2014. [Online]. Available: http://doi.acm.org/10.1145/2629430
- [2] H. V. Jayashree, H. Thapliyal, H. R. Arabnia, and V. K. Agrawal, "Ancilla input and garbage-output optimized design of a reversible quantum integer multiplier," The Journal of Supercomputing, vol. 72, no. 4, pp.1477–1493, 2016. [Online]. Available: http://dx.doi.org/10.1007/s11227-016-1676-0
- [3] H. M. H. Babu, "Cost-efficient design of a quantum multiplier–accumulator unit," Quantum Information Processing, vol. 16, no. 1,p. 30, 2016. [Online]. Available: http://dx.doi.org/10.1007/s11128-016-1455-0

Conclusion

- Resource efficient design of quantum circuits are vital to the design of quantum algorithms in hardware
- New methodologies to design efficient quantum circuits for arithmetic operation need special attention in quantum computing.