J Supercomput (2016) 72:1477–1493
DOI 10.1007/s11227-016-1676-0

Ancilla-input and garbage-output optimized design of a reversible quantum integer multiplier

H. V. Jayashree[1] · Himanshu Thapliyal[2] · Hamid R. Arabnia[3] · V. K. Agrawal[4]

Published online: 18 February 2016

Abstract A reversible logic has application in quantum computing. A reversible logic design needs resources such as ancilla and garbage qubits to reconfigure circuit functions or gate functions. The removal of garbage qubits and ancilla qubits are essential in designing an efficient quantum circuit. In the literature, there are multiple designs that have been proposed for a reversible multiplication operation. A multiplication hardware is essential for the circuit design of quantum algorithms, quantum cryptanalysis, and digital signal processing applications. The existing designs of reversible quantum integer multipliers suffer from redundant garbage qubits. In this work, we propose a reversible logic based, garbage-free and ancilla qubit optimized design of a quantum integer multiplier. The proposed quantum integer multiplier utilizes a novel add and rotate methodology that is specially suitable for a reversible computing paradigm. The proposed design methodology is the modified version of a conventional shift and add method. The proposed design of the quantum integer multiplier incorporates add or no operation based on multiplier qubits and followed by a rotate right operation. The proposed design of the quantum integer multiplier produces zero garbage qubits and shows an improvement ranging from 60 to 90 % in ancilla qubits count over the existing work on reversible quantum integer multipliers.

Keywords Reversible logic · Multiplier · Fredkin gate · Quantum arithmetic

✉ Himanshu Thapliyal
hthapliyal@uky.edu

[1] Department of ECE, PES Institute of Technology, Bangalore, KA, India

[2] Department of Electrical and Computer Engineering, University of Kentucky, Lexington, KY, USA

[3] Department of Computer Science, University of Georgia, Athens, GA, USA

[4] Department of ISE, PES Institute of Technology, Bangalore, KA, India

1 Introduction

A reversible logic has application in quantum computing. Reversible circuits are required to have an equal number of inputs and outputs. They are designed without any feedback and fanout. There are a few parameters or resource constraints used to measure the performance of reversible circuits namely quantum cost (QC), garbage outputs (GO), ancilla inputs (AI), gate count (GC), and delay ($\triangle$). The quantum cost of a reversible circuit is the number of 1×1 and 2×2 quantum gates that are used to construct the circuit. Garbage outputs are the ones that are neither the primary outputs nor the ones required for further computation. An ancilla or constant inputs are required to derive a certain function and to retain one-to-one mapping. A delay corresponds to the number of primitive quantum gates in the critical path of the circuit.

Multipliers are the major computational units that are used frequently in digital signal processing computations. Optimization is a major objective in designing a multiplier with design constraints. In a reversible circuit design, it is necessary to minimize the count of garbage outputs and ancilla inputs in order to reduce the total number of qubits; therefore, we present the design of a reversible multiplier which produces zero garbage outputs and minimizes the number of ancilla inputs compared to the existing multiplier designs in the literature.

In this work, we present a modified version of the add and shift method of multiplication. The basic components used in our design are add or NOP block and rotate right (ROR) block. To meet our design requirement, we also present a modified circuit of reversible ALU design presented in [1]. In addition, we present a generalized circuit design methodology that is supported by a generalized behavioral model to design a constant depth rotate right reversible circuit. This design is motivated by the design presented in [2]. The reversible multiplier design presented in this work outperforms existing multiplier designs in terms of its garbage outputs and ancilla inputs. We also give an estimate of the gate count, quantum cost, ancilla inputs, and delay for $N \times N$ qubit multiplier. In the optimization of performance parameters, there is always a trade-off; such as in optimizing one parameter, the other parameters get affected. Here, our objective is to optimize the ancilla inputs and garbage outputs, due to this the remaining parameters get affected. To indicate the trade-off, the estimation of all the performance parameters is given for each block used in designing a $N \times N$ reversible multiplier. The paper is organized into several sections namely: Sect. 1 provides an introduction to reversible logic gates; Sect. 2 elaborates on the background of reversible logic gates; Sects. 3–5 cover existing designs, behavioral model of the proposed design, and the proposed circuit design methodology, respectively; Sects. 6–8 cover performance parameters calculation, results comparison, and conclusion, respectively.

2 Background on reversible logic gates

This section covers the basics of reversible logic gates. Any N variable reversible system is built with $N \times N$ reversible circuits. A few 1×1 and 2×2 primitive quantum gates are used to construct large-sized reversible gates and circuits. The quantum cost of reversible gates used in this work can be found in [3]. The reversible gates used in this work are Fredkin, CNOT, Toffoli, and Swap gates [4–6].

2.1 CNOT gate

A CNOT gate is also known as a Feynman gate (FG). It is a 2×2 reversible gate. The inputs and outputs are denoted as (A, B) and (P, Q), respectively. Here, A is treated as a control qubit, while B is treated as a target qubit. The mapping of input to outputs are denoted as $P \leftrightarrow A$, $Q \leftrightarrow A \oplus B$. The block diagram and symbol of CNOT gate are shown in Fig. 1. QC of FG is 1.

2.2 Toffoli gate (TG)

This gate is also known as a C^2 NOT gate. The TG used in our work is a 3×3 gate with inputs (A, B, C) and outputs (P, Q, R), respectively. Here, A and B are the control qubits, while C is the target qubit. The mapping between inputs and outputs is given with the relation $P \leftrightarrow A$, $Q \leftrightarrow B$, $R \leftrightarrow (A \cdot B) \oplus C$. The block diagram and symbol of TG are presented in Fig. 2. QC of TG is 5.

2.3 Fredkin gate (FRG)

A Fredkin gate is commonly used as a controlled Swap gate. In this paper, we use a 3×3 Fredkin gate. A, B, and C are the inputs and P, Q, and R are the output qubits. The mapping of the input and outputs are given based on the value of A, which is the control qubit. When A is high, $Q \leftrightarrow C$ and $R \leftrightarrow B$. When A is low, $Q \leftrightarrow B$ and $R \leftrightarrow C$. Irrespective of A value, $P \leftrightarrow A$. The block diagram and symbol of FRG gate are shown in Fig. 3. QC of FRG is 5.

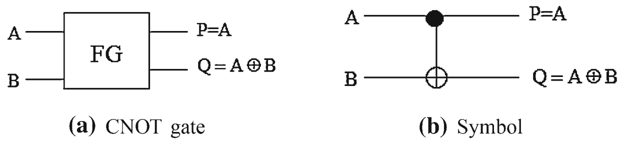


Fig. 1 CNOT gate and its symbol

Fig. 2 Toffoli gate and its symbol

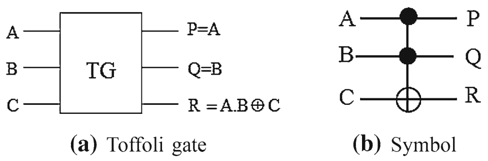
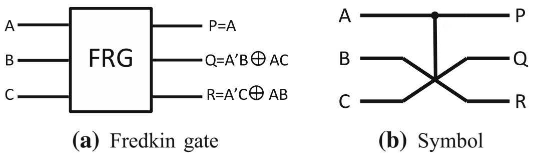


Fig. 3 Fredkin gate and its symbol



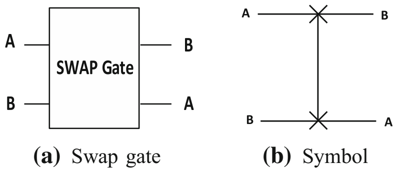


Fig. 4 Swap gate and its symbol

2.4 Swap gate (SG)

The Swap gate is a 2×2 gate. It swaps the input and output qubits unconditionally. The mapping is $A \leftrightarrow Q$ and $B \leftrightarrow P$. The block diagram and symbol are shown in Fig. 4. QC of SG is 3.

3 Existing work

The research on reversible logic is being explored in the domains of design, synthesis, and testing. Although there are many synthesis techniques available to realize reversible circuits, having dedicated designs of a reversible circuit component gives flexibility in choosing the designs based on the application requirement. Multiple ways of designing arithmetic circuits have been explored in conventional, reversible and quantum computing [7–21]. Several interesting contributions have been explored in the existing synthesis of reversible logic circuits [3,22–29]. In this section, we discuss the existing reversible multiplier designs that are important arithmetic circuits in processing digital signals.

There are several multiplier designs proposed by many authors in view of optimizing different performance parameters namely quantum cost, ancilla inputs, garbage outputs, logic depth, and a combination of these parameters. The multipliers proposed in [30–34] follow two phases in computing product terms. In the first phase, the partial products are computed. In the second phase, the summation of partial products are computed to get the final product terms. In all the designs mentioned above, the partial product generation and summation stages are improved either in terms of constant inputs, quantum cost, or garbage outputs. We found that the design in [34] gives better results when compared to other existing work in terms of ancilla inputs and gate count. Apart from the regular parallel or array multiplier designs, other techniques of multiplications like Booth, Wallace, and Vedic are proposed by researchers in [35–37]. All these designs are illustrated for smaller operand width. It is necessary for any designer to choose designs based on their performance over a wide range of operand width. We found that the recent publication on multiplier in [38] discussed NxN reversible multiplier design. We considered the designs proposed in [38,39] and compared it with our work, as the efforts in both of these papers were to reduce the number of constant inputs and garbage outputs.

4 Proposed reversible multiplier behavioral model

In this section, we present an algorithm for multiplying two n qubit numbers A and B. The result is stored in $2n$ qubit product register P. There are two conventional techniques of add and shift method of multiplying two numbers. Shift the multiplicand left and add it to the product register contents iteratively or add the multiplicand and shift the product register contents right iteratively. In the first technique, after the computation is completed, the multiplicand will not be in its original form and since one cannot recover the multiplicand, it will be considered as garbage qubits. We choose the second technique. The final result will be the contents of the product register and the multiplicand contents are unaltered, so that garbage outputs are not generated.

4.1 Behavioral model of $N \times N$ multiplier

The multiplier algorithm is best explained using an example of 4×4 multiplication with a dot diagram as shown in Fig. 5. Initially, the P register is loaded with ancilla 0 qubits. The multiplicand B is added to the P register contents. For the P register, it considers a least significant position (LSP) from $n-1$ and move up to $2n-1$ position. For the B register, LSP starts from 0 and moves up to $n-1$. A multiplicand is added to the P register only if the corresponding multiplier qubit is high; otherwise, only rotate operation is performed. The rotate right operation is performed irrespective of the value of a multiplier qubit. While adding a multiplicand to the P register, the $(n-1)$th position of the P register is aligned to the 0th position of the B register. Due to this alignment, one rotate operation is eliminated at the end of computation. The diagram shown in Fig. 5 is self-explanatory of the algorithm.

4.2 Behavioral model of rotate right operation

In this section, we present a rotate right operation for $2n$ qubit data width. In the multiplication technique presented in the Algorithm 1, there is a need to rotate the P register contents to the right; the size of the P register is $2n$ qubit width. The rotate right operation is performed by swapping the qubits in two stages. The circuit is designed to obtain constant logic depth. To give an illustrative example, we present a rotate right operation in Fig. 6 for data width of 8 qubits. The numbers 0–7 represent the qubit positions. The initial representation of qubits is shown in the left most part of Fig. 6. The rotate right operation is performed in two stages. In the first stage, qubits

Algorithm 1 Add and rotate method to model $N \times N$ multiplier

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

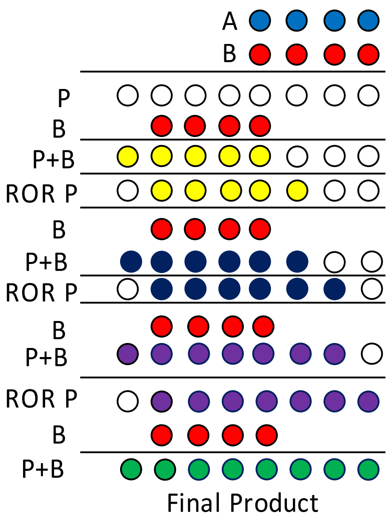


Fig. 5 4 × 4 qubit multiplication dot diagram

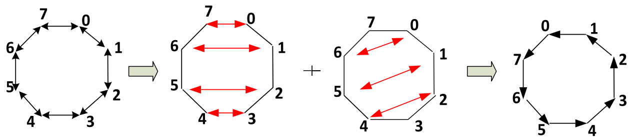


Fig. 6 Rotate right with two sets of disjoint transpositions

in the position pairs (0, 7), (1, 6), (5, 2), and (4, 3) are swapped [(0, 7) indicates 0 is swapped with 7]. In the second stage, (0, 6), (1, 5), and (2, 4) are swapped. It is visible from the diagram that the swapping of qubits in each stage is parallel which reduces the logic depth compared to the sequential shifting of qubits. The reversible circuit design of rotate right operation will be discussed in the latter section of this paper. We present a generalized pseudo-code for the method discussed in Fig. 6 to swap the qubits in two stages.

The pseudo-code presented in the Algorithm 2 performs rotate right operation by 1 qubit position. This code works for both even and odd data width. The Algorithm 2 will be useful in any application when data input width is variable (i.e., even or odd). For the multiplication technique proposed in this paper, the width of product register (P) is always even; hence, the second half of the pseudo-code is redundant for the proposed work.

5 Proposed garbageless reversible multiplier circuit design

From the behavior models Algorithms 1 and 2 presented in the Sect. 4, it is clear that to compute the product of two n qubit numbers, we need to design the following reversible circuits: (1) n qubit addition or no operation (ADD/NOP) circuit; (2) uncontrolled rotate right operation circuit. This section elaborates on the reversible circuit design methodology of ADD/NOP and rotate right (ROR) block.

Algorithm 2 Pseudo-code for rotate right operation

```
ROTATERIGHT($|P\rangle$)
$k$=SIZEOF($|P\rangle$);                                   ▷ $k$ is an integer
$k1$=FLOOR($k/2$);                                         ▷ $k1$ is an integer
if $k$ mod 2 == 0 then                                     ▷ For even number of qubits
    $i = 0$; $j = k - 1$;                                  ▷ $i$ and $j$ are integers
    while $i < k1$ && $j >= k1$ do                         ▷ First Stage
        SWAP($|P_{[i]}\rangle$,$|P_{[j]}\rangle$);
        $i = i + 1$; $j = j - 1$;
    end while
    $i = 0$; $j = k - 2$;
    while $i < k1 - 1$ && $j >= k1$ do                     ▷ Second stage
        SWAP($|P_{[i]}\rangle|P_{[j]}\rangle$);
        $i = i + 1$; $j = j - 1$;
    end while
else                                                       ▷ For odd number of qubits
    $i = 0$; $j = k - 1$;
    while $i < k1$ && $j >= k1 + 1$ do                     ▷ First Stage
        SWAP($|P_{[i]}\rangle$,$|P_{[j]}\rangle$)
        $i = i + 1$; $j = j - 1$
    end while
    $i = 0$; $j = k - 2$;
    while $i < k1$ && $j >= k1$ do                         ▷ Second Stage
        SWAP($|P_{[i]}\rangle$,$|P_{[j]}\rangle$)
        $i = i + 1$; $j = j - 1$;
    end while
end if
return $P$;
```

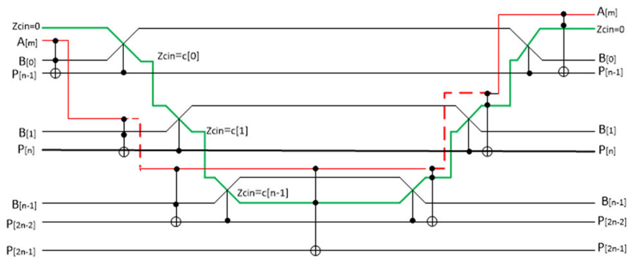


Fig. 7 Reversible ADD/NOP circuit

5.1 ADD or NOP circuit design

ADD or NOP block has evolved from the ALU design proposed in [1]. We have modified the original work to adapt to our garbageless multiplier design. The reversible circuit design of ADD/NOP block is shown in Fig. 7. Inputs to the ADD/NOP block are:

(a) n qubit product register $|P_{[2n-1:n-1]}\rangle$; (b) n qubit input operand $|B_{[n-1:0]}\rangle$; (c) 1 qubit input $Zcin$ initialized with ancilla 0; (d) 1 qubit input operand $A_{[m]}$, where m is the qubit position varying from 0 to $n-1$. Here, $A_{[m]}$ acts as the control qubit; if it is high, the P and B register contents are added. At the output, we have the B register contents unaltered, where the P register contents will have the sum of P and

B contents. If $A_{[m]}$ is low, then the B and P register contents are regenerated at the output without modification. The role of $Zcin$ is to propagate the carry generated from the previous qubit position. If the control qubit $A_{[m]}$ is high, $P_{[2n-1]}$ will have the final carry out generated; otherwise, it will retain its value.

The computation of ADD/NOP block is summarized below. Here, the index of B varies from 0 to $n-1$ and P varies from $n-1$ to $2n-1$, according to the requirement of the multiplier design. The index of A is chosen to be m which ranges from 0 to $n-1$, where n is the size of operands (multiplier and multiplicand). Here, j is used to indicate the qubit position of the product register P.

1. *Computation Phase 1*
   Initialize $Zcin$ with ancilla 0 qubit. In further stages, the same line will propagate the carry generated from the previous stage.
2. *Step 1* Apply 3×3 Toffoli gate at locations $A_{[m]}$, $B_{[0]}$, and $P_{[n-1]}$. After the computation, $A_{[m]}$ and $B_{[0]}$ will retain their value. $P_{[n-1]}$ will get transformed according to the equation given below.

$$|P_{[n-1]}\rangle = (|A_{[m]}\rangle \cdot |B_{[0]}\rangle) \oplus |P_{[n-1]}\rangle \tag{1}$$

3. *Step 2a* For $0 \le i \le n-1$ and $n-1 \le j \le 2n-2$, apply 3×3 Fredkin gate at locations $P_{[j]}$, $B_{[i]}$, and $Zcin$. Here, $P_{[j]}$ acts as a control line to FRG gate and it will not change after the computation. The remaining lines $B_{[i]}$ and $Zcin$ will get modified according to the equations shown below.

$$|Zcin\rangle = \begin{cases} |B_{[i]}\rangle & \text{if } |P_{[j]}\rangle = 1 \\ |Zcin\rangle & \text{if } |P_{[j]}\rangle = 0 \end{cases} \tag{2}$$

$$|B_{[i]}\rangle = \begin{cases} |Zcin\rangle & \text{if } |P_{[j]}\rangle = 1 \\ |B_{[i]}\rangle & \text{if } |P_{[j]}\rangle = 0 \end{cases} \tag{3}$$

4. *Step 2b* For $1 \le \mathrm{i} \le n-1$ and $n \le \mathrm{j} \le 2n-2$, apply a 3×3 Toffoli gate at locations $A_{[m]}$, $B_{[i]}$, and $P_{[j]}$. After the computation, $A_{[m]}$ and $B_{[i]}$ will retain their value. $P_{[j]}$ will get transformed according to the equations given below.

$$|P_{[j]}\rangle = \begin{cases} |P_{[j]}\rangle \oplus |B_{[i]}\rangle & \text{if } |A_{[m]}\rangle = 1 \\ |P_{[j]}\rangle & \text{if } |A_{[m]}\rangle = 0 \end{cases} \tag{4}$$

   Steps 2a and 2b execute concurrently.
5. *Step 3* Apply a 3×3 Toffoli gate at locations $A_{[m]}$, $B_{[n-1]}$, and $P_{[2n-1]}$. This step is required in order to store the final carry out after n qubit addition. After the computation, $P_{[2n-1]}$ stores final carry out.

$$|P_{[2n-1]}\rangle = \begin{cases} |B_{[n-1]}\rangle & \text{if } |A_{[m]}\rangle = 1 \\ |P_{[2n-1]}\rangle & \text{if } |A_{[m]}\rangle = 0 \end{cases} \tag{5}$$

In other words, $P_{[2n-1]}$ stores the final carry out only if the control line that is corresponding to the multiplier qubit $A_{[m]}$ is high; otherwise, it will restore the previous value present in it, that means, it will retain the previous carry value stored in $P_{[2n-1]}$ from the previous computation.

6. *Computation Phase 2*
   The steps in this phase contribute to the generation of the final product value P and the regeneration of the contents of multiplicand B .
7. *Step 4a* For $2n-2 \geq j \geq n-1$ and $n-1 \geq i \geq 0$, apply a 3×3 Fredkin gate at locations $P_{[j]}$, $B_{[i]}$, and $Zcin$. After the computation, the value at $P_{[j]}$ will be retained as it is whereas $Zcin$ and $B_{[i]}$ will get modified according to the equations shown in Computation Phase 1, Step 2a.
8. *Step 4b* For $2n-2 \geq j \geq n-1$ and $n-1 \geq i \geq 0$, apply a 3×3 Toffoli gate at locations $A_{[m]}$, $Zcin$, and $P_{[j]}$. At the end of computation, $A_{[m]}$ and $Zcin$ will retain its value; whereas, $P_{[j]}$ will get modified according to the equation mentioned below.

$$\left|P_{[j]}\right\rangle = \begin{cases} |Zcin\rangle \oplus \left|P_{[j]}\right\rangle & \text{if } \left|A_{[m]}\right\rangle = 1 \\ \left|P_{[j]}\right\rangle & \text{if } \left|A_{[m]}\right\rangle = 0 \end{cases} \tag{6}$$

Steps 4a and 4b execute sequentially.

5.2 Rotate right reversible circuit design

The reversible circuit for the rotate right operation is shown in Fig. 8. The circuit takes no ancilla and rotates the data to the right from MSB qubits to LSB qubits by 1 position (ROR). The reversible rotate circuit is designed using Swap gates and performs a rotate operation with constant delay. For clarity of understanding, we have shown eight qubit ROR circuit design. The product register qubits $P_{[0]}$ to $P_{[7]}$ are given as input. After one rotate operation, $P_{[0]}$ occupies $P_{[7]}$th qubit position and qubits from $P_{[7]}$ to $P_{[1]}$ shift to the right by one position. The quantum cost of Swap gate is 3. The delay in performing the rotation operation involves two Swap gates in series; therefore, the constant delay of 6 is obtained by considering each cycle individually and decomposing it into two sets of disjoint swaps. The gates shown in the dotted boxes are executed in parallel. The rotator design is motivated by the property proven in [2]. According to the authors, any permutation is the composition of two set of disjoint transpositions. This is illustrated in Fig. 6, which also shows that the cycle is a composition of two reflections. The authors have proven that any permutation of n qubits can be performed in 4 layers (levels or logic depth) of CNOT gates with n ancilla input qubits, or in six layers with no ancilla input qubits (delay of two Swap gates). If we had opted for first technique of multiplication in which the multiplicand (n bit operand) is shifted, it leaves the multiplicand altered, which in turn will yield garbage or garbage output.

The rotate circuit proposed performs an unconditional rotate operation in the sense that irrespective of multiplier qubit value, a rotation is performed. Another option that we explored was to use a conventional multiplication technique that needs con-

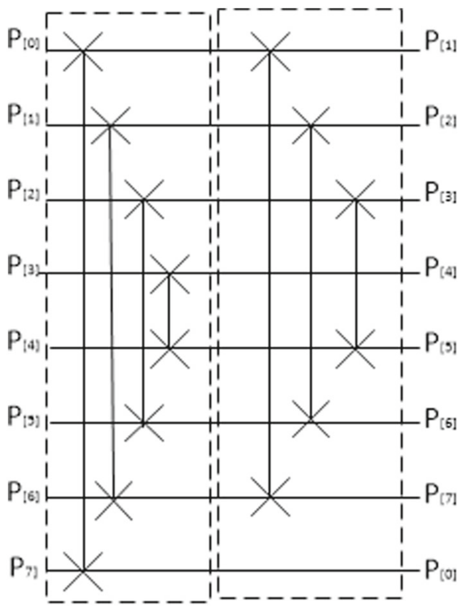


Fig. 8 Reversible rotate right circuit (ROR)

ditional shift or rotate circuit. A controlled Swap gate or Fredkin gate instead of Swap gate can be used to rotate the qubits. The rotate operation is controlled by $A_{[m]}$ qubit. The same control qubit is used by all the Fredkin gates in the rotate circuit as shown in Fig. 9. Here, the computation becomes sequential and the delay will increase with the size of the rotate circuit unlike our proposed design. Another reason for ignoring the design shown in Fig. 9 is that the quantum cost of Fredkin gate is more than the Swap gate. To optimize the delay and quantum cost, we omitted this option. If the delay has to be maintained constant, then one has to store the control lines for these Fredkin gates and use them in parallel. This will again increase the number of ancilla lines, thus violating our objective of minimizing the ancilla lines.

5.3 Reversible multiplier circuit design methodology

In this section, we illustrate the design steps of a reversible multiplier show in Fig. 10.

1. For $m = 0$ to $n - 2$ repeat Step-1 and Step-2.
2. Step 1: *ADD or NOP*
   Apply the data qubits $A_{[m]}$, $Zcin$, product register $P_{[2n-1:n-1]}$, and the multiplicand register contents $B_{[n-1:0]}$ to ADD/NOP block. After the computation, the contents of $A_{[m]}$, $Zcin$, and B are restored; whereas, the P register contents will get modified according to the computation equations mentioned in the Section V-A. ADD/NOP circuit will perform the addition on P and B register contents if $A_{[m]}$ =high; otherwise, the contents of those registers are retained.

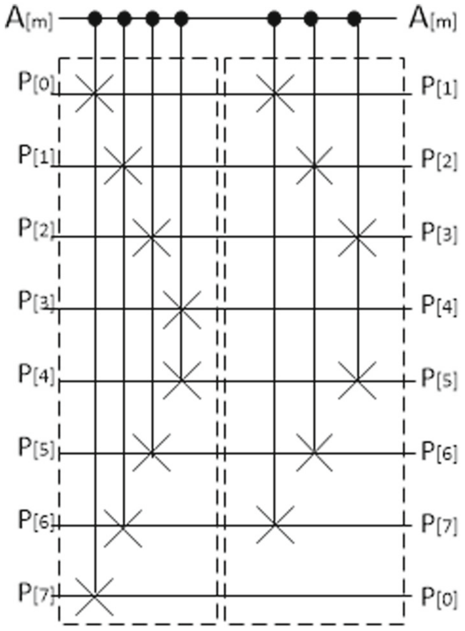


Fig. 9 Alternative reversible rotate circuit

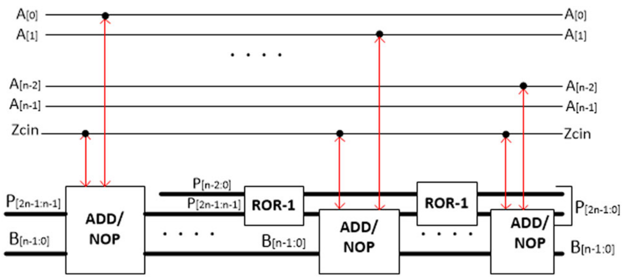


Fig. 10 Reversible multiplier circuit

3. Step 2: *rotate right (ROR)*
   Apply the P register contents to ROR (ROR-1 block indicates rotate right by 1 position) block, which performs a rotate right operation with a constant delay of 6. The computation is carried out as follows:

$$\left|P_{[2n-1:0]}\right\rangle \leftarrow \left|P_{[2n-1:0]}\right\rangle \circlearrowright 1.$$

4. Step 3: update $m = n - 1$, repeat Step-1.

6 Performance parameters calculation

In this section, we discuss the performance parameters and the calculation for each circuit used in the reversible multiplier design. As a final part of the calculation, we show the overall calculation of the reversible multiplier.

6.1 Performance parameters of ADD/NOP block

The equations shown below are with respect to the design mentioned in Fig. 7. The calculation of quantum cost (QC) is shown below.

$$\begin{aligned} \text{QC(ADD/NOP)} &= 5 \times \text{No. of TG} + 5 \times \text{No. of FRG} \\ &= 5 \times (2n + 1) + 5 \times (2n) \\ &= 20n + 5 \end{aligned} \tag{7}$$

The ancilla inputs include the product register qubits $(n + 1)$ which are initially set to ancilla 0 and $Zcin$ used for carry propagation initialized to ancilla 0. So the ancilla for n qubit ADD/NOP block is given below.

$$\text{AI(ADD/NOP)} = n + 2 \tag{8}$$

The delay of ADD/NOP block includes the critical path delay. To find the critical path, the design has been divided into stages where each stage is sequential in execution. This is illustrated in Fig. 11 in vertical lines. The computation stages are divided into Phase 1 and Phase 2. The Phase 1 consists of the computation of half sum and final carry out. In Phase 2, the full sum is computed and the content of the B register is regenerated. There are $3n+2$ stages. We have computed the number of stages involving Toffoli gates and Fredkin gates. Since the delay is proportional to the quantum gates present in each reversible gate, the total delay of ADD/NOP circuit can be found by using the equation shown below.

$$\begin{aligned} \triangle\text{(ADD/NOP)} &= \text{Delay of single stage} \times \text{No. of stages} \\ &= 5 \times (3n + 2) \\ &= 15n + 10 \end{aligned} \tag{9}$$

6.2 Performance parameter of ROR block

The performance parameters are estimated for the rotate right reversible circuit (ROR). It is discussed in the previous sections that to perform one-time rotation, two phases of swap operations are carried out. The swapping of qubits in each phase are computed parallely. The rotate circuit presented has no garbage outputs and no ancilla input qubits. Qcost (QC) and delay ($\triangle$) are calculated as follows:

$$\begin{aligned} \text{QC(ROR)} &= 3 \times \text{No. of SG} \\ &= 3 \times (n - 1) \\ &= 3n - 3 \end{aligned} \tag{10}$$

The above equation is for a generalized rotate circuit design. For our work, we feed the $2n$ qubits of the product register contents to the rotate block. The modified equation is shown below.

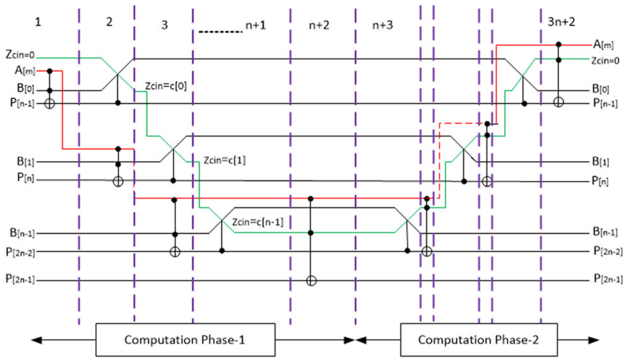


Fig. 11 Critical path computation

$$\begin{aligned}\text{QC(ROR)} &= 3 \times \text{No. of SG} \\ &= 3 \times (2n - 1) \\ &= 6n - 3 \qquad (11)\end{aligned}$$

$$\triangle(\text{ROR}) = 6 \qquad (12)$$

6.3 Performance parameters of $N \times N$ reversible multiplier block

The performance parameters calculation for $N \times N$ reversible multiplier block is generated by summing up the calculations of ADD/NOP and rotate right block (ROR) components.

$$\begin{aligned}\text{QC(Mul)} &= n \times \text{QC(ADD/NOP)} + (n - 1) \times \text{QC(ROR)} \\ &= n \times (20n + 5) + (n - 1) \times (6n - 3) \\ &= 26n^2 - 4n + 3 \qquad (13)\end{aligned}$$

Although the ancilla inputs of the rotate right circuit is nil, the input to the ADD/NOP block and ROR circuit is the P register contents, and all the $2n$ qubit locations of P register which are initialized with ancilla 0 qubits.

$$\begin{aligned}\text{AI(Mul)} &= \text{AI(ADD/NOP)} + \text{AI(ROR)} \\ &= 2n + 1 \qquad (14)\end{aligned}$$

The delay of the multiplier is the summation of delay of ADD/NOP block and rotate right circuit.

$$\begin{aligned}\triangle(\text{Mul}) &= n \times \triangle(\text{ADD/NOP}) + (n - 1) \times \triangle(\text{ROR}) \\ &= n \times (15n + 10) + (n - 1) \times 6 \\ &= 15n^2 + 16n - 6 \qquad (15)\end{aligned}$$

7 Comparison results

In the literature, there are more designs presented for a 4×4 reversible multiplier. It is necessary to design a circuit which is scalable to any size. Hence, we compared our design with other $N \times N$ reversible designs that are available in the literature. The designs proposed in [38,39] show the calculation for $N \times N$ reversible multiplier. In both of these papers, the authors have shown only the ancilla inputs and garbage outputs calculations; the comparisons shown in Tables 1 and 2 list only ancilla inputs and garbage outputs for these papers. It is clear from the result shown in Table 1 that as the operands size increases, the percentage of improvement also increases. Our proposed design showed a better improvement in terms of the ancilla inputs resulting in saving the chip area since the number of lines are reduced.

We have listed the garbage outputs of the designs proposed in [38,39]. Our design outperforms the existing designs because it is 100 % better in terms of garbage outputs

Table 1 Ancilla inputs comparison of $N \times N$ reversible multiplier

N	Ancilla inputs in proposed design	Ancilla inputs in Kotiyal et al. [38]	Ancilla inputs in Zhou et al. [39]	% Imp over [38]	% Imp over [39]
4	9	23	28	60.86	67.85
8	17	83	120	79.51	85.83
16	33	303	496	89.10	93.34
32	65	1135	2016	94.27	96.77
64	129	4351	8128	97.03	98.41
128	257	16,959	32,640	98.48	99.21
256	513	66,815	130,816	99.23	99.60
512	1025	264,959	523,776	99.61	99.80
1024	2049	1,054,719	2,096,128	99.80	99.90

Table 2 Garbage outputs comparison for $N \times N$ reversible multiplier

N	Garbage outputs in Kotiyal et al. [38]	Garbage outputs in Zhou et al. [39]	% Imp over [38,39]
4	22	36	
8	81	168	
16	300	720	
32	1131	2976	
64	4346	12,096	100
128	16,953	48,768	
256	66,808	195,840	
512	264,951	784,896	
1024	1,054,719	3,142,656	

Table 3 Comparison with Karatsuba recursive multiplier

Designs	Gate count	Ancilla inputs	Delay
$K(1)$	$O(n^{\log_2 3})$	$6n$	$O(n)$
$K(2)$	$O(n^{\log_2 6})$	$4n$	$O(n^{\log_2 6})$
$K(3)$	$O(n^{\log_2 3})$	$5n + n/2 + 1$	$O(n^{\log_2 3})$
$K(4)$	$O(n^{\log_2 6})$	$3n + n/2$	$O(n^{\log_2 6})$
Proposed	$O(n^2)$	$2n + 1$	$O(n^2)$

$K(1)$, $K(2)$, $K(3)$, and $K(4)$ indicates Karatsuba designs 1, 2, 3, and 4 proposed in [40]

since our design produces no garbage. We compared our design with another garbageless reversible multiplier design using the recursive scheme in [40]. Here, we compare our work with Karatsuba multiplier design presented in [40]. The comparison for gate count, ancilla inputs, and delay is shown in Table 3. The design of Karatsuba (1) shown in Table 3 follows Bennet's first scheme. An extra register is used to store the result and the circuit is run backward. The parallel recursive calls are made to reduce the time complexity. The design of Karatsuba (2) also uses parallel recursive call, but the design does not follow Bennet's first scheme, instead it follows a recursive garbage disposal scheme. The multiplication is computed parallel to garbage disposal. But the trade-off is that the number of gates increases due to the different design blocks adapted in the garbage disposal design process. The design of Karatsuba (3) follows Bennet's first scheme for garbage disposal. The only difference with respect to the design of Karatsuba (1) is that the recursive calls are sequential rather than parallel. Karatsuba (4) is designed using the recursive scheme similar to the one adapted in Karatsuba (2), but recursive calls are sequential. For the designs presented in [40], we considered the minimum bound on ancilla inputs calculation. It is observed from Table 3 that with the slight increase in the delay and gate count, the proposed design has improved the ancilla inputs compared to all the Karatsuba designs.

8 Conclusion

In this work, we have proposed ADD and rotate based on an $N \times N$ reversible multiplier design. We presented the general behavioral model of the design. The proposed multiplier is compared with the relevant existing reversible multiplier designs in the literature. We presented the generalized equations for the performance parameters of the proposed reversible multiplier. It is observed from comparison results that our work outperforms the other designs in terms of the ancilla inputs and zero garbage outputs. The proposed design can be integrated in a larger data path subsystem designs where the garbage outputs and ancilla inputs reductions are the major concerns.

References

1. Thomsen MK, Glück R, Axelsen HB (2010) Reversible arithmetic logic unit for quantum arithmetic. J Phys A Math Theor 43(38):382002
2. Margolus N (1990) Parallel quantum computation. Complex Entropy Phys Inf Santa Fe Inst Stud Sci Complex 8:273–287

3. Maslov D (2005) Reversible logic synthesis benchmarks page. http://www.cs.uvic.ca/~dmaslov. Accessed 31 Aug 2015
4. Fredkin E, Toffoli T (2002) Conservative logic. Springer, New York
5. Barenco A, Bennett CH, Cleve R, DiVincenzo DP, Margolus N, Shor P, Sleator T, Smolin JA, Weinfurter H (1995) Elementary gates for quantum computation. Phys Rev A 52(5):3457
6. Smolin JA, DiVincenzo DP (1996) Five two-bit quantum gates are sufficient to implement the quantum fredkin gate. Phys Rev A 53(4):2855–2856
7. Draper TG, Kutin SA, Rains EM, Svore KM (2006) A logarithmic-depth quantum carry-lookahead adder. Quantum Inf Comput 6(4):351–369
8. Takahashi Y, Kunihiro N (2005) A linear-size quantum circuit for addition with no ancillary qubits. Quantum Inf Comput 5(6):440–448
9. Takahashi Y (2009) Quantum arithmetic circuits: a survey. IEICE Trans Fundam Electron Commun Comput Sci 92(5):1276–1283
10. Takahashi Y, Tani S, Kunihiro N (2009) Quantum addition circuits and unbounded fan-out. arXiv:0910.2530
11. Choi B-S, Van Meter R (2011) On the effect of quantum interaction distance on quantum addition circuits. ACM J Emerg Technol Comput Syst (JETC) 7(3):11
12. Vedral V, Barenco A, Ekert A (1996) Quantum networks for elementary arithmetic operations. Phys Rev A 54(1):147
13. Thapliyal H, Ranganathan N (2013) Design of efficient reversible logic-based binary and bcd adder circuits. ACM J Emerg Technol Comput Syst (JETC) 9(3):17
14. Thapliyal H, Jayashree H, Nagamani A, Arabnia HR (2013) Progress in reversible processor design: a novel methodology for reversible carry look-ahead adder. In: Transactions on computational science XVII. Springer, New York, pp 73–97
15. Thapliyal H, Arabnia H, Vinod AP (2006) Combined integer and floating point multiplication architecture (CIFM) for FPGAs and its reversible logic implementation. In: 49th IEEE international midwest symposium on circuits and systems (MWSCAS'06), vol 2. IEEE, pp 438–442
16. Thapliyal H, Arabnia HR, Srinivas M (2006) Reduced area low power high throughput bcd adders for ieee 754r format. arXiv:cs/0609036
17. Thapliyal H, Arabnia HR (2006) Reversible programmable logic array (RPLA) using Fredkin & Feynman gates for industrial electronics and applications. arXiv:cs/0609029
18. Thapliyal H, Srinivas M, Arabnia HR (2005) Reversible logic synthesis of half, full and parallel subtractors. In: ESA, pp 165–181
19. Thapliyal H, Srinivas MB, Arabnia HR (2005) A need of quantum computing: reversible logic synthesis of parallel binary adder-subtractor. In: Embedded systems and applications, pp 60–68
20. Thapliyal H, Srinivas M, Arabnia HR (2005) A reversible version of 4 × 4 bit array multiplier with minimum gates and garbage outputs. In: Embedded systems and applications, pp 106–116
21. Thapliyal H, Arabnia HR, Srinivas M (2009) Efficient reversible logic design of BCD subtractors. In: Transactions on computational science III. Springer, New York, pp 99–121
22. Prasad AK, Shende V, Markov I, Hayes J, Patel KN (2006) Data structures and algorithms for simplifying reversible circuits. ACM JETC 2(4):277–293
23. Golubitsky O, Maslov D (2012) A study of optimal 4-bit reversible toffoli circuits and their synthesis. IEEE Trans Comput 61(9):1341–1353
24. Maslov D, Dueck GW (2004) Reversible cascades with minimal garbage. IEEE Trans Comput-Aided Des Integr Circuits Syst 23(11):1497–1509
25. Yang G, Song X, Hung WN, Perkowski MA (2008) Bi-directional synthesis of 4-bit reversible circuits. Comput J 51(2):207–215
26. Maslov D, Saeedi M (2011) Reversible circuit optimization via leaving the boolean domain. IEEE Trans Comput-Aided Des Integr Circuits Syst 30(6):806–816
27. Maslov D (2015) On the advantages of using relative phase Toffolis with an application to multiple control Toffoli optimization. arXiv:1508.03273
28. Saeedi M, Zamani MS, Sedighi M, Sasanian Z (2010) Reversible circuit synthesis using a cycle-based approach. J Emerg Technol Comput Syst 6:13:1–13:26
29. Hung WN, Song X, Yang G, Yang J, Perkowski M (2006) Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis. IEEE Trans Comput-Aided Des 25(9):1652–1663

30. Zomorodi Moghadam M, Navi K (2012) Ultra-area-efficient reversible multiplier. Microelectron J 43(6):377–385
31. Bhagyalakshmi H, Venkatesha M (2012) Optimized multiplier using reversible multi-control input Toffoli gates. Int J VLSI Des Commun Syst 3(6):27–40
32. Panchal VK, Nayak VH (2014) Analysis of multiplier circuit using reversible logic. Int J Sci Res Dev 1(6):279–284
33. Rangaraju H, Suresh AB, Muralidhara K (2013) Design of efficient reversible multiplier. In: Advances in computing and information technology. Springer, New York, pp 571–579
34. Mamataj S, Das B, Chandran S (2015) An approach for designing an optimized reversible parallel multiplier by reversible gates. In: Computational advancement in communication circuits and systems. Springer, New York, pp 345–355
35. Sultana J, Mitra SK, Chowdhury AR (2015) On the analysis of reversible booth's multiplier. In: 2015 28th international conference on VLSI design (VLSID). IEEE, pp 170–175
36. Thapliyal H, Srinivas M (2006) Novel reversible multiplier architecture using reversible TSG gate. arXiv:cs/0605004
37. Banerjee A, Das DK (2013) The design of reversible multiplier using ancient indian mathematics. In: 2013 international symposium on electronic system design (ISED). IEEE, pp 31–35
38. Kotiyal S, Thapliyal H, Ranganathan N (2015) Reversible logic based multiplication computing unit using binary tree data structure. J Supercomput 71(7):2668–2693
39. Zhou R, Shi Y, Wang H, Cao J (2011) Transistor realization of reversible ZS series gates and reversible array multiplier. Microelectron J 42(2):305–315
40. Portugal R, Figueiredo C (2006) Reversible Karatsubas algorithm. J Univers Comput Sci 12(5):499–511