Optimization and Implementation of Reversible BCD Adder in Terms of Number of Lines

P. Radhika Ramya, Y. Sudha Vani
Departement of Electronics and Communication Engineering, Vignan University

Article Info

Article history:

Received Dec 2, 2012
Received Jan 16, 2013
Accepted Feb 5, 2013

Keyword:

Reversible logic
Low-power CMOS
BCD adder
Quantum computing

ABSTRACT

The Reversible logic plays an important role to obtain the minimal energy dissipation. It is the main requirement in the low power digital design. To implement a reversible function some additional lines are required. Sometimes it is more than the primary inputs which leads to increased size of the circuit. Hence the reducing the number of circuit lines is one of the major criterion in reversible logic. The general idea is to merge the garbage output lines with appropriate constant input lines. In this work, Toffoli gate implementation of BCD adder and press gate implementation of BCD adder is taken and an optimization algorithm has been applied to reduce the number of lines in the circuit. The kit used to implement the design is Vertex5 in which both the area and delay are analyzed.

Corresponding Author:

Y.Sudha Vani,
Departement of Electronics and Communication Engineering,
Vignan University,
Guntur, Andhrapradesh, India
Email: vani.yamani@gmail.com

1. INTRODUCTION

The shrinking transistor sizes and power dissipation are the major barriers in the development of smaller and powerful circuits [1]. So the low power designs with high performance are given prime importance by researchers as power has become a first order design consideration. While efforts are being made to reduce power dissipation due to leakage currents, alternate circuit design considerations are also gaining importance. The reversible logic provides the power optimization with its application in nanotechnology, low power CMOS, and quantum computing [2-3].

R.Landauer's [4] demonstrated in the early 1960s, irreversible hardware computation results in energy dissipation due to the information loss, regardless of its realization technique. It is proved that the loss of each one bit of information dissipates at least KTln2 joules of energy in the form of heat, where $K=1.3806505x10^{-23}m^2kg^{-2}K^{-1}$ is the Boltzmann's constant and T is the absolute temperature at which operation is performed [5-6]. Reversible logic circuits have theoretically zero internal power dissipation because they do not lose information. Bennett [7] showed that in order to avoid KTln2 joules of energy dissipation in a circuit, it must be built using reversible logic gates.

If the function to be synthesized is reversible, the number of circuit lines must be equal to the number of the primary inputs and primary outputs. There is a significant difference in the synthesis of logic circuits using classical logic gates and reversible gates [8-10]. Synthesis of reversible logic circuits is significantly more complicated than traditional irreversible logic circuits because in a reversible logic circuit, fan-out and feedback are not allowed [11]. A reversible logic circuit should have the following features:

a) Use minimum number of reversible gates.
b) Use minimum number of garbage outputs.
c) Use minimum constant inputs.

A reversible implementation of BCD adder with reduced number of garbage outputs is proposed by James [8] and further some of the implementations have been done [10]. These are implemented using different reversible gates. In some of the reversible circuits often need a significant amount of additional circuit lines sometimes magnitude more than the primary inputs. So this is the problem if number of circuit lines are increasing it may decrease the reliability of the circuit. Thus keeping the number of circuit lines as small as possible is an important issue. Some of the reversible circuits often need a significant amount of additional circuit lines sometimes magnitude more than the primary inputs. If the numbers of lines in the circuit are increasing it may decrease the reliability of the circuit. Thus keeping the number of circuit lines as small as possible is an important issue. To overcome this problem the optimization algorithm is proposed [2].

In this work, a Toffoli gate implementation of BCD adder is considered and the optimized is applied to circuit to reduce the number of lines.

The rest of the paper is structured as follows. The section II describes reversible logic and reversible gates. The section III gives the reversible implementation using Toffoli Gates. The optimization algorithm and the implementation results and comparison with related work are shown in section IV and section V .The section VI concludes the work.

2. REVERSIBILE LOGIC AND REVERSIBLE GATES

Reversible logic- A multiple-output Boolean function $B^n \rightarrow B^n$ is reversible if it maps each input pattern to a unique output pattern and the number of inputs is equal to the number of outputs. A reversible function can be realized by a circuit $G=G_1, G_2:: G_k$ comprised of a cascade of reversible gates G_i with no fan-out or feedback (see eg [3][5]). Each reversible gate consists of a set of control lines and target lines. The gate operation is applied to the target lines if all the control lines meet the required control conditions. Control lines and unconnected lines always pass through the gate unaltered.

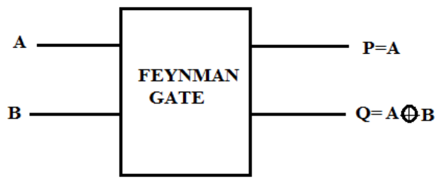

Figure 1. 2*2 Feynman gate

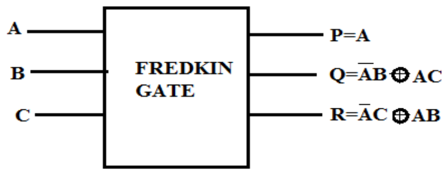

Figure 2. 3*3 Fredkin gate

Reversible gates-the important basic reversible logic gates are Feynman gate which is 2*2 reversible gate [9] which is as shown in Figure 1, and the fredkin gate [9], toffoli gate[9], peres gate [12] which are 3*3 reversible gates are shown in Figure 2, 3, 5.

The 3*3 toffoli gate (TG) as shown in Figure 3 is used to generate an AND and XOR functions. If c='0', then R=AB and if B='1' then R=A ⊕C. The Figure 4 shows the 3*3 toffoli gate in the reversible circuit representation.

The Control lines are denoted by ●, while the target lines are denoted by ⊕. This gate is two through gate because two of its outputs are identical with its inputs. Because of this, Toffoli gate is also known as two controlled NOT (2-CNOT). If the first two input bits are one, then the third output bit is the inverse of third input bit i.e., A = B = 1, then R = C. As discussed earlier, any reversible gate has an inverse or dual. The dual of Toffoli gate is also a Toffoli gate and so it is self-reversible. TG can be considered as a universal gate.

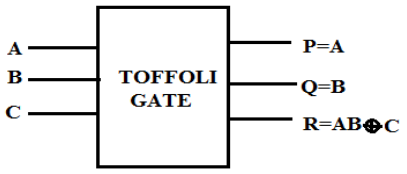

Figure 3. 3*3 Toffoli gate

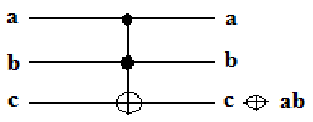

Figure 4. 3*3 Toffoli Gate (TG) representations in reversible circuit

The 3*3 peres gate (PG) as shown in Figure 5. It is used to generate an fulladder function by combining the two peres gates together. If the first peres gate c=’0’ then it will work as ahalfadder. so that the second peres gate will take the halfadder outputs as the inputs and cin.so the we can get the fulladder output. The Figure 6, shows the peres gate represntation in reversible circuit as a fulladder.

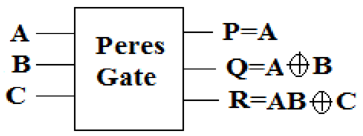

Figure 5. 3*3 Peres gate

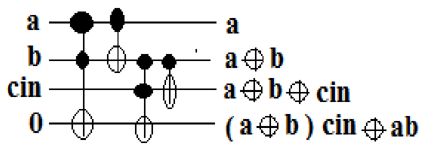

Figure 6. 3*3 Peres gate represntation in reversible circuit as a fulladder

The 4*4 Double Peres gate (DPG) as shown in Figure 7 is used as a full adder function. If D=cin and C=’0’ then it will work as a fulladder. The Figure 8 shows the DPG gate in the reversible circuit representation.

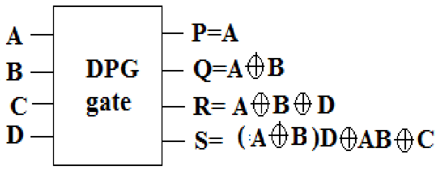

Figure 7. 4*4 Double Peres gate

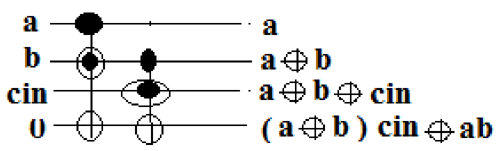

Figure 8. 4*4 Double Peres gate representation in reversible circuit

3. REVERSIBLE IMPLEMENTATION USING TOFFOLI GATES

The reversible implementation of the BCD adder is done by using the toffoli gates. The conventional BCD adder consists of three blocks: 4- bit binary adder, 6-correction circuit and a final adder, which is a modified special adder is shown in the Figure 9.

Reversible conventional BCD adder- Figure 10 shows the conventional BCD adder implemented using toffoli gates. The circuit uses 28 gates, 8 constant inputs and results in 12 garbage outputs. The no of garbage outputs and constant inputs are more in this implementation. The BCD adder implementation using DPG gates in [12] uses 8 constant inputs and results in 10 garbage outputs. The BCD adder implementation using Peres gates in [11] uses 13 constant inputs and results in 18 garbage outputs. The constant inputs and garbage outputs can be reduced by implementing the optimization algorithm.

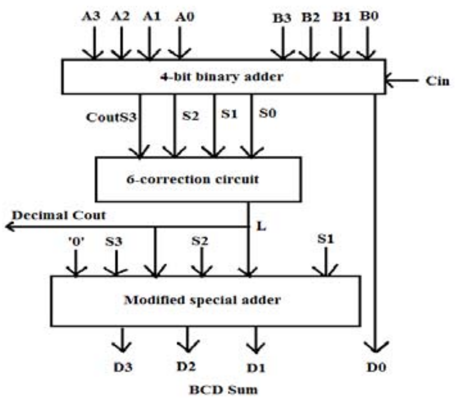

Figure 9. Conventional BCD adder

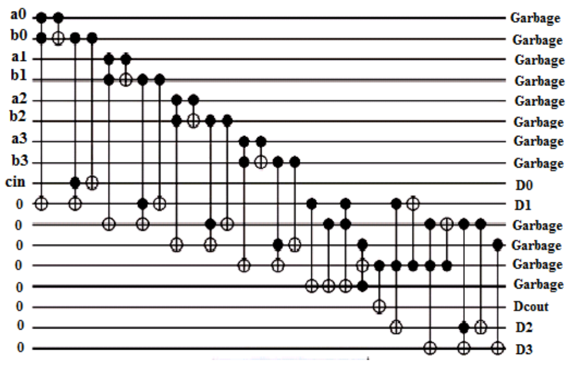

Figure 10. Toffoli gate implementation of conventional BCD adder circuit

4. OPTIMIZATION ALGORITHM

The basic steps given in the algorithm [2] are as follows:

a) At first, an appropriate sub-circuit is determined.
b) Tried to re-synthesize the sub-circuit so that one of the (garbage) outputs returns a constant value.
c) If this was successful, the re-synthesized sub-circuit is inserted into the original circuit.
d) Finally, the newly created constant output is merged with a line including a constant input.

The algorithm terminates if no appropriate sub-circuit can be determined anymore. The following the respective steps are described in detail.

4.1. Determine an Appropriate Sub-circuit

The appropriate sub-circuits are characterized by the fact that they include at least one garbage output which can be later used to replace a constant input. Therefore, it is important to know when lines of a circuit are used for the first time and when they are not needed anymore, respectively. This is formalized by the following two functions.

Definition 1. Let $G = g_1 \dots g_d$ be a reversible circuit. Furthermore, let $l \in \{1 \dots n\}$ be a line of this circuit. Then, the function firstly used (l) returns $i \in \{1, \dots, d\}$ if g_i is the first gate connected with line1. Accordingly, the function lastly used (l) returns $i \in \{1 \dots d\}$ if g_i is the last gate connected with line1.

Using these functions, the flow to determine appropriate sub-circuits can be described as follows:

a) Traverse all circuit lines lg of the circuit $G = g_1 \dots g_d$ that include a garbage output.
b) Check if line l_g can be merged with another line l_c including a constant input, i.e. if there is a constant input line l_c so that firstly used(l_c)>lastly used(l_g). If this check fails, continue with the next garbage output line l_g.
c) Check if circuit Gk 1 = $g_1 \dots g_k$ with k = lastly used (l_g) can be modified so that line lg outputs a constant value. If this check fails, continue with the next line lg in Step 2. Otherwise, Gk1 is an appropriate sub-circuit.

Note that the order in which the garbage output lines lg are considered typically has an effect. We consider the lines with the smallest value of lastly used (l_g) first. This is motivated by the fact that firstly used (l_c) > lastly used (l_g) is a necessary condition which, in particular, becomes true for small values of lastly used (l_g). Besides that, the check in Step 3 is strongly related to the resynthesis of the sub circuit which is described next.

4.2. Re-synthesize the Sub-circuit

The next step is to re-synthesize the sub-circuit so that one garbage output returns a constant value (instead letting it a don’t care). Generally, any available synthesis approach can be applied for this purpose. But since we want to reduce the number of circuit lines, an approach that generates additional circuit lines [6] should be avoided. Here only sub-circuits with a limited number of primary inputs are considered. Starting at the output of line l_g, the circuit is traversed towards the inputs of the circuit. Each passed gate as well the lines connected with them are added to the following consideration. The traversal stops, if the number of considered lines reaches a given threshold λ.

From the resulting cascade, a truth table description is determined. Afterwards, the truth table is modified, i.e. the former garbage output at line l_g is replaced by a constant output value. It is thereby important that the modification preserves the reversibility of the function. If this is not possible, the sub-circuit is skipped and the next line with a garbage output is considered (see Step 3 from above). Otherwise, the modified truth table can be passed to a synthesis approach.

Note that the modification of the truth table is only possible, if constant values at the primary inputs of the whole circuit are incorporated. Constant inputs restrict the number of possible assignments to the inputs of the considered cascade. This enables a reversible embedding with a constant output.

Re-synthesizing the sub-circuit as described might lead to an increase in the number of gates. This is an expected behavior since circuit lines can be exploited to buffer temporary values [1]. If such lines are removed, additional gates may be required to recompute these values.

4.3. Insert the Sub-circuit and Merge the Lines

If re-synthesis was successful, the last two steps are Straight-forward. At first, the considered sub-circuit is replaced by the newly synthesized one. Afterwards, the considered garbage output line l_g is merged with the respective constant input line l_c, i.e. the respective gate connections as well as possible primary outputs are adjusted. Finally, the Line l_c is removed since it is not needed anymore.

The above steps are applied to the implementation in Figure 10. The resulting implementation is shown in Figure 11. This implementation uses 32gates, 4constant inputs and results in 8 garbage outputs.

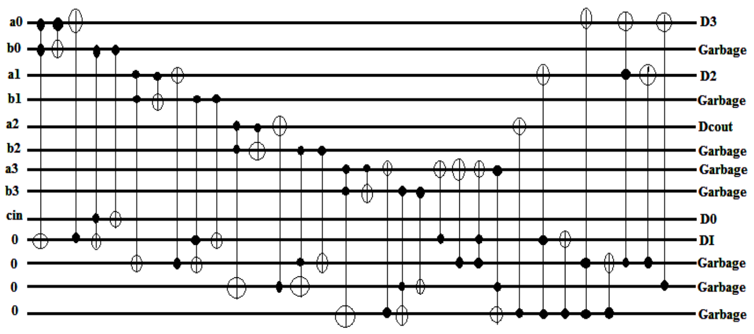

Figure 11. Optimized Toffoli gate implementation of BCD adder

The above steps are applied to the implementation the BCD adder using DPG gate.The resulting implementation is shown in Figure 12. This implementation uses 24gates, 4constant inputs and results in 8 garbage outputs.

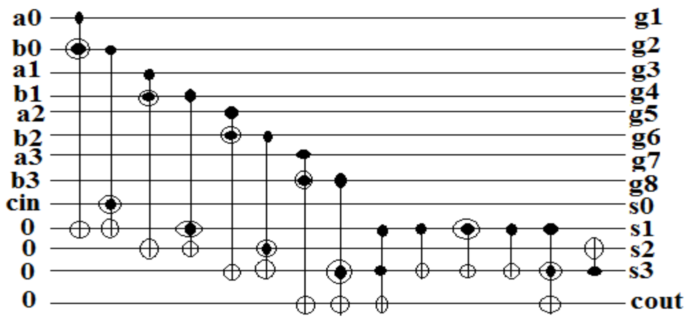

Figure 12. Optimized Double Peres gate implementation of BCD adder

The above steps are applied to the implementation the BCD adder using Peres gate. The resulting implementation is shown in fig13.This implementation uses 32gates, 10constant inputs and results in 14 garbage outputs.

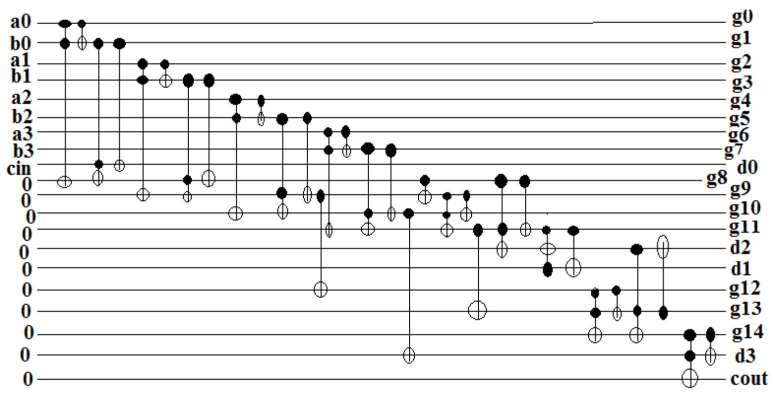

Figure 13. Optimized Peres gate implementation of BCD adder

5. EXPERIMENTAL RESULTS

The proposed approach for line reduction has been implemented in Spartan-3e. As synthesis method for step (b) of the optimization; two different approaches have been evaluated, namely

a) An exact synthesis approach that realizes a circuit with minimal number of gates but usually requires a significant amount of run-time.
b) A heuristic synthesis approach that does not ensure minimality but is very efficient regarding run-time.

The result is tabulated in Table 1 & Table 2. This work shows that we can achieve an improvement factor of 2 for No. of lines. Depending on the approach used for resynthesis, this might lead to an increase in the number of gates. However, if exact synthesis is applied, the increase remains small-in some cases even a reduction can be observed.

Table 1. Comparative Analysis for Unoptimized designs

Parameters	Unoptimzed Toffoli BCD adder[8]	Unoptimized peres gate BCD adder[11]	Unoptimized dpg gate of bcd adder[12]
No. of gate	28	35	22
Garbage outputs	12	18	10
Constant inputs	8	13	8
No.of lines	17	23	25

Table 2. Comparative Analysis for optimized designs

Parameters	Optimized toffoli BCD adder[this work]	Optimzed peres gate BCD adder[this work]	optimized dpg gate of bcd adder[this work]
No. of gate	32	32	24
Garbage outputs	8	14	8
Constant inputs	4	10	4
No. of lines	13	20	13
Delay	5.679	6.619	4.419
Number of Slice LUTs	1%	1%	1%
Number of bonded IOBs	15%	18%	11%

6. CONCLUSION

Keeping the number of lines in the reversible BCD adder circuits as small as possible is an important issue; in particular the design is to reduce the number of garbage outputs, which is the most important factor for reversible circuit cost. However, during synthesis of complex reversible circuits, often additional lines are appended to the circuit. In this paper, we presented a post-optimization approach for circuit line reduction. The general idea is thereby to identify garbage outputs and re-synthesize them so that they can be connected with a constant input.

REFERENCES

[1] DM Miller, R Wille, R Drechsler. *Reducing reversible circuit cost by adding lines*. In Int'l Symp. On Multi-Valued Logic. 2010.
[2] Robert Wille, Mathias Soeken, Rolf Drechsler. Reducing the number of Lines in Reversible Circuits. *In IEEE conf*. 2010: 647-652.
[3] D Maslov, GW Dueck. Reversible cascades with minimal garbage. *IEEE Trans. on CAD*. 2004; 23(11): 1497–1509.
[4] R Landauer. Irreversibility and heat generation in the computing process. *IBM J. Res. Dev.,* 1961; 5: 183.
[5] P Gupta, A Agrawal, NK Jha. An algorithm for synthesis of reversible logic circuits. *IEEE Trans. on CAD*. 2006; 25(11): 2317–2330.
[6] R Wille, R Drechsler. *BDD-based synthesis of reversible logic for large functions*. In Design Automation Conf., 2009; 270–275.
[7] CH Bennett. Logical reversibility of computation. *IBM J. Res. Dev*., 1973; 17(6): 525–532.
[8] Rekha K, K Poulose Jacob, Sreela Sasi. Reversible Binary Coded adders Using Toffoli Gates. In Advances Computational Algorithm and Data Analysis. *Springer Science Business media* B.V. 2009.
[9] Md Hafiz Hasan Babu, AR Chowdhury. Design of a Reversible Binary Coded Decimal Adder by using reversible 4-Bit Parallel adder. *VLSI Design,* Kolkata, India. 2005; 255-260.
[10] RK James, TK Shahana, KP Jacob, S Sasi. *Improved Reversible Logic Implementation of Decimal Adder*. IEEE 11th VDAT Symposium. Kolkata, India. 2007.
[11] HR Bhagyalakshmi, MK Venkatesha. Optimized Reversible BCD Adder Using New Reversible Logic Gates. *Journal of computing*. 2010; 2(2). ISSN 2151-9617.
[12] HR Bhagyalakshmi, MK Venkatesha. Optimized design of BCD adder and Carry skip BCD adder using reversible logic gates. *International Journal on Computer Sciences and engineering.* 2011; 3(4). ISSN 0975-3397.