Reversible Implementation of Square-root Circuit

Sayeeda Sultana
Department of Electrical and Computer Engineering
McGill University
Montreal, Canada
sayeeda.sultana@mail.mcgill.ca

Katarzyna Radecka
Department of Electrical and Computer Engineering
McGill University
Montreal, Canada
katarzyna.radecka@mcgill.ca

***Abstract*—in this paper we present a novel reversible implementation of a square-root circuit with an array structure. In scientific computations such as numerical analysis, computer graphics, complex number computations, square root is an important operation. In classical irreversible arena we find different realizations of square root circuit. Since reversible circuit is emerging as an alternative to classical circuit, here we introduce a novel reversible realization of this operation. As a basic module, we propose a reversible controlled adder/subtractor (RCAS) block based on 2's Complement computation. In our design we use an array of such RCAS blocks which perform addition or subtraction based on the result generated from digit-by-digit square root operation. To our best knowledge this is the first methodical approach for implementing reversible square root circuit. The new structure of the circuit and different parameters – number of gates, garbage bits and quantum cost for n-bit realization is presented here.**

Keywords-Reversible logic, Controlled adder/Subtractor, 2's complement computation, Square-root Circuit

I. INTRODUCTION

Reversible circuits emerge as a plausible alternative to classical designs due to their information loss-less computation performed with considerably less energy. Their close relation to quantum circuits elevated the ongoing research interests. Further, they find applications in vast areas of computing such as low power designs, adiabatic circuits, cryptography, digital signal processing and optical computing.

A reversible function realizes a unique one-to-one mapping of inputs to outputs, while providing no data loss during the computation [1-2]. So far, the synthesis of reversible circuits from irreversible or reversible specification has been done intensively and finding an optimized reversible realization is still a challenging issue. Moreover, a great part of the existing algorithms are restricted to small functions but optimized in garbage bits and gate counts, while some approaches successfully address large functions though quite costly in terms of gate count, additional lines and quantum cost, especially from irreversible specification which include Positive Polarity Reed Muller, BDD based, ESOP based realization [3-7]. In addition, a great effort has been made towards implementing the basic reversible arithmetic units such as adders, subtractors and multipliers by finding a direct translation from classical to reversible forms [8-10].

In scientific computations next to basic mathematical operations of addition, subtraction, multiplication and division, square-root is most useful and vital. For example, numerical analysis, complex number computations, statistical analysis, computer graphics and signal processing are among the fields where square root is of relevance [11]. Although, the realization of a square root in classical circuits is well established, the way of implementing this operation in emerging technologies is not yet sound. We find only one example of reversible 8-bit square root circuit as benchmark result in [13]. However, this is a discrete example of square root operation, not a regular structure or generalized method of building reversible square root circuit. In this paper, we propose a structured methodology of implementing this arithmetic circuit. Based on its classical realization, we generate the reversible embedding which is an array structure of basic blocks, and can realize square root circuits of any size. In particular, we follow the classical non-restoring array structure of square-root circuit [11], which performs 2's Complement addition/subtraction controlled by a digit-by-digit square root result. Here, we design a *reversible controlled adder/subtractor* (RCAS) block, which performs 2's Complement computation, and can be used in a modular way to execute a square root operation.

II. BACKGROUND

Definition 1: *An nxn reversible circuit realizes an n-input/n-output function where each input vector maps bijectively to an output vector. The reversible circuits allow no fanouts and no feedback paths.*

Hence, the iterative cascading preserving the rules listed in Def. 1 can be applied to building any reversible circuit using reversible gates. Reversible synthesis of an irreversible specification aims on embedding an arbitrary (irreversible) function with *I* inputs and *O* outputs (generally *I≠O*) to a reversible implementation, constructed solely from reversible gates. Often extra I/O signals must be added. The extra *A* inputs are referred as *ancilla* bits and the additional *G* outputs are *garbage* bits.

The cost of a reversible circuit is determined in terms of a gate count and a *quantum cost*. Since different reversible gates, the number of gates in the implementation is not a reliable parameter for comparison and hence most researchers rely on the quantum cost, which plays also an important role from the technological viewpoint.

Definition 2*: A quantum cost of a reversible gate T is defined as a number of elementary quantum operations*

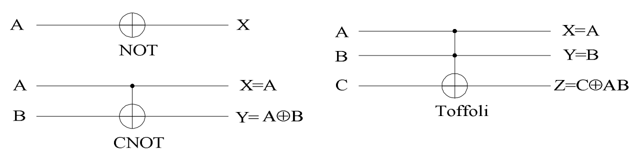

Figure 1: Standard reversible gates

performed by NOT, CNOT and controlled V or V^+ gates in order to realize this T gate.

Many reversible gates have been proposed over the years such as basic NOT, CNOT, Toffoli [1], as well as Fredkin [1] and Peres [2] gates. Recently, some reversible gates targeting specific implementations have been introduced. However, since in this work we use basic reversible (NOT, CNOT and Toffoli) as well as Peres gates for the construction of our RCAS block, a brief introduction of them is presented below.

A CNOT or Feynman gate, Fig.1 is a 2x2 gate with a single control input A used to invert the second input B when A=1. A Toffoli gate has a single target line C that is inverted if all the control lines are set to 1, Fig.1. The quantum cost of a Toffoli gate is 5. This 3x3 Toffoli gate is *universal* since any reversible function can be realized by cascading this gate only. A multiple controlled 3x3 Fredkin gate [1], Fig. 2(a), is a controlled swap gate with two target lines and the values of the target lines are interchanged if the control lines are set to 1. The quantum cost of this gate is 5. The Fredkin gate is used in many arithmetic designs. Another gate commonly used in reversible implementations, is the 3x3 Peres gate, Fig. 2(b). This gate implements the following mapping of the inputs (A, B, C) to the outputs ($X = A$, $Y = A \oplus B$, $Z = AB \oplus C$). The quantum cost of this gate is 4. The Peres gate has the capability to implement a half adder using only one gate instance.

III. Reversible Controlled Adder/Subtractor

In the proposed square root circuit, we explore the non-restoring algorithm and array structure of a controlled adder/subtractor (CAS) module realized in reversible embedding. The reversible controlled adder/subtractor (RCAS) module is similar to classical one, which performs 2's Complement addition and subtraction following the rules of reversibility in Def.1 with the aid of some extraneous bits.

A. Reversible Controlled Adder/Subtractor (RCAS)

The reversible controlled adder/subtractor (CAS) block is designed to perform an addition or subtraction depending on the value of the input control signal. The concept applied here is based on the use of an adder circuit to perform subtraction instead of having a dedicated subtractor. Hence, the operation X-Y is implemented as the 2's Complement $X+Y'$+1 [12]. A classical adder/subtractor module has 4 inputs (A/S, C_{in}, X and Y) and 2 outputs (S/D and C_{out}), its input/output count is unequal, and hence the original form is not reversible. Hence, to create a cascadable reversible CAS module, some garbage bits must be added and the value assignments of them must be such that every input combination maps to a unique output pattern, while, at the same time, preserving the original addition/subtraction operation. Note, that the minimum number of garbage signals is $|Log_2M|$, where M is the maximum number of output pattern repeated. In this case M=6, bringing the number of garbage signals to 3. As this would still imbalance the I/O compatibility (4 inputs vs. 5 outputs) a single ancilla bit set permanently to 0 is added as an input to equalize input-output count. The truth table of the reversible controlled adder/subtractor (RCAS) module is presented in Table 1.

A second issue, addressed in the RCAS construction is the lack of support for fan-out signals in reversible designs compared to their classical counterparts where the same control signal A/S can be fanned-out to all CAS blocks. To overcome this shortcoming, the garbage bit A/S_g, Table 1, is used to provide a copy of a control signal from one module to the next.

Table 1: Reversible controlled adder/subtractor truth table

const.	A/S	Cin	X	Y	S/D	Cout	A/S_g	Y_g	g2
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	1	1
0	0	0	1	0	1	0	0	0	1
0	0	0	1	1	0	1	0	1	0
0	0	1	0	0	1	0	0	0	0
0	0	1	0	1	0	1	0	1	1
0	0	1	1	0	0	1	0	0	1
0	0	1	1	1	1	1	0	1	0
0	1	0	0	0	1	0	1	0	1
0	1	0	0	1	0	0	1	1	0
0	1	0	1	0	0	1	1	0	0
0	1	0	1	1	1	0	1	1	1
0	1	1	0	0	0	1	1	0	1
0	1	1	0	1	1	0	1	1	0
0	1	1	1	0	1	1	1	0	0
0	1	1	1	1	0	1	1	1	1

The implementation of RCAS module is shown in Fig. 3 using one CNOT gate and two Peres gates (highlighted by dotted lines). When A/S is se to 0, then CNOT passes the true copy of Y while A/S_g acts as a garbage output. When A/S is 1 then Y' is available at the output of CNOT, and the full adder adds X, Y' and C_{in} which is set to '1' for subtraction. The garbage output A/S_g of a given RCAS block is reused for a control signal fed to the consecutive RCAS block. To obtain a copy of input Y for further operation we add another CNOT gate between lines A/S and $A/S \oplus Y$ with control at A/S_g , which performs the operation: $A/S \oplus Y \oplus A/S = Y$. The quantum cost of the RCAS module is 10.

IV. Reversible Square Root Circuit

A square root circuit is not modular, unlike an adder/subtractor, which has a regular structure suitable for cascading. However, still many hardware elements introduced in the previous sections are common for a square root and an adder/subtractor, and hence can be reused. For example, an array structure of a non-restoring square root circuit presented

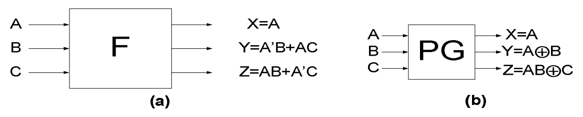

Figure 2: (a) Fredkin gate (b) Peres gate

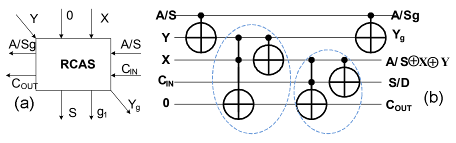

Figure 3: RCAS module (a) block diagram (b) reversible realization

in [11] uses the classical controlled adder/subtractor blocks, Fig.4. The 8-bit non-restoring circuit realizes a digit-by-digit scheme, where at each iteration computed in each row, only one digit of square root is performed [11]. Based on this structure we create our reversible square root circuit using reversible controlled adder/subtractor block (RCAS) presented in Section III. A few additional CNOT gates are added to provide fan-out signals. Note that the order the RCAS blocks are placed in an n-bit square root circuit in reversible embedding is introduced in order to comply with reversibility properties, Def. 1, while preserving the correctness of the execution of the square root operation.

In the proposed reversible square-root circuit, we incorporate the RCAS block from Fig. 3 and reuse the copy of input signal Y of previous stage for the next stage square root operation. For example, Y outputs of CAS3 and CAS4 in row 2 are connected to Y inputs of CAS8 and CAS9 in row 3, Fig. 4(b). The RCAS guarantees the generation of all the original irreversible fan-out signals required for our square root circuit with minimal cost of the implementation. Before we explain a generalized design we present the procedure to realize a small reversible square root circuit first.

A 4-bit reversible square root circuit with the input $x_1x_2x_3x_4$, the 2-bit square root output q_1q_2 and the 4-bit remainder output $r_1r_2r_3r_4$ is presented in Fig.5. It incorporates the top two rows of Fig. 4. However in the classical implementation in the first row, the control input signal A/S (set to 1) is propagated through CAS1 and CAS2, and is fed back to CAS2 as carry-in signal. The carry-out of CAS2 serves as a C_{in} input to CAS1. Note, that the propagated A/S_g from CAS1 is the same signal for the both CAS1 and CAS2 blocks. Further, the carry-out of all CAS blocks is transmitted from right to left (for example C_{out} of CAS2 is connected to C_{in} of CAS1, C_{out} of CAS6 to C_{in} of CAS5, etc.). Since no feedback is allowed in reversible embedding, we change the order of the RCAS signals propagation into the direction from right to left. Hence the right-most RCAS1 block at the top row in Fig. 5(b) takes the control input A/S. In a classical design, Fig. 4(b) the A/S signal of CAS1 is set to '1' in order to calculate the first digit of square root. We also set A/S input of RCAS1 to '1', Fig. 5(b), as C_{in} of the RCAS1 block is connected to the control input A/S. Hence, a required copy of A/S signal in the reversible embedding is obtained by using a CNOT gate (right most gate CNOT$_{\text{cin}}$ in the top row of Fig. 5(a), with the I/O mapping: A/S, '0'$\rightarrow A/S_g$, $A/S\oplus 0$). The outputs of this CNOT are connected to

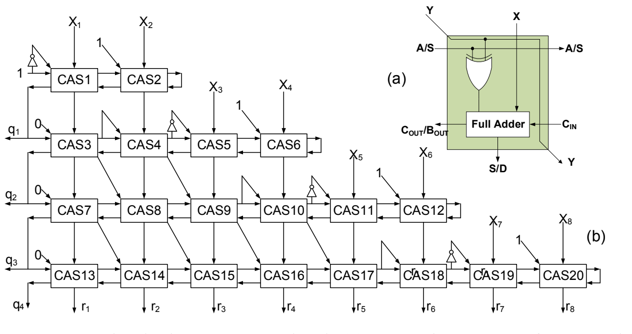

Figure 4: Classical square-root circuit (a) Internal structure of CAS and (b) 8-bit square root circuit [11]

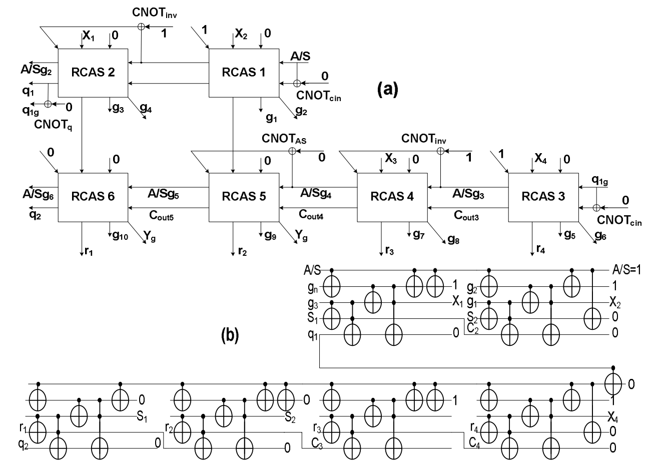

Figure 5: 4-bit reversible square root cirtcuit (a) Block diagram and (b) reversible implementation

inputs A/S and C_{in} of RCAS1 respectively. The outputs A/S_g and C_{out} of the RCAS2 block are connected to the left RCAS1 block. To provide the inverted A/S as well as A/S itself, each inverter in a classical implementation is replaced by a CNOT gate marked as a CNOT$_{\text{inv}}$ in Fig. 5(a) with the I/O mapping: A/S, '1'$\rightarrow A/S_g$, $A/S\oplus 1$). In addition, since each square root bit from each row (q_i) is the control signal for the next row (Fig. 4), we need a fan-out of q_i. In reversible implementation, we use a CNOT gate to generate a copy of a required signal. For example, to obtain a copy of q_1, we use a CNOT gate (CNOT$_{\text{q}}$ in Fig. 5(a)) with the I/O mapping: q_1, '0'$\rightarrow q_1$, $q_1\oplus 0$.

A. N-bit Reversible square root circuit:

The classical array structure in Fig. 4 can easily be extended to incorporate any size of the square root operation. In general, for a $2n$-bit square root circuit, in order to generate an n-bit square root output we need n rows of CAS blocks, each row having $2i$ CAS blocks, where i is the order of rows (i = 1, 2,..., n) [11]. The architecture of a $2n$-bit square root is a direct extension of the schematics presented in Fig. 4.

Similarly we can implement our reversible array structure of $2n$-bit square root circuit. We need $n(n+1)$ RCAS blocks, arranged in n rows with $2i$ RACS blocks in each row (order of rows i = 1, 2,..., n). In each row the right-most RCAS block control signal (A/S) has a fan-out to the C_{in} input. To copy the A/S signal we need n extra CNOT gates (CNOT$_{\text{cin}}$ in Fig. 5(a)) of the I/O configuration: (X, '0' $\rightarrow X$, $X\oplus 0$). Also, the input Y of the 2nd most right RCAS block in each row is connected to the inverted A/S signal. Hence, we need additional n CNOT gates (CNOT$_{\text{inv}}$ in Fig. 5(a)) realizing the I/O mapping: X, '1' $\rightarrow X$, $X\oplus 1$). Likewise, the input Y of the 3rd RCAS block from the right in each row excluding the 1st row is connected to A/S. Hence the additional n-1 CNOT gates (CNOT$_{\text{AS}}$ in Fig. 5(a)) implementing the function (X, '0' $\rightarrow X$, $X\oplus 0$) are required. Finally, n-1 CNOT (CNOT$_{\text{q}}$ gates in Fig. 5(a)) performing the mapping: (X, '0' $\rightarrow X$, $X\oplus 0$) are used to obtain a copy of a square root digit (q_i) from each row except the final row.

B. Analysis of n-bit circuit parameters:

Each RCAS block in Fig. 5 requires two CNOT and two Peres gates, while generating 3 garbage outputs. Note, the actual number of the garbage bits is 2, as the garbage signal A/S_g is reused for the propagation of the input control signal A/S between two cascaded RCAS blocks. Hence, the implementation of a $2n$-bit input/ n-bit output circuit requires $2n(n+1)$ Peres and $2n(n+3)$-2 CNOT gates, totaling in the $4n^2+8n-2$ number of reversible gates. The quantum cost of the overall design is $10n^2+14n-2$. The garbage outputs generated for an n-bit square root circuit is n^2+6n-2. The circuit parameters for different sizes are shown in table 2.

Table 2: Costs of Square root for different size

Circuit size	# of gates	#of garbage outputs	Quantum cost
4	30	14	66
8	94	78	214
16	318	110	750
32	1150	350	2782

V. SIMULATION RESULTS

In this section we present the simulation results for the proposed designs of the RCAS block Fig. 6 and an 8-bit square root circuit, Fig. 7. All designs were implemented in VHDL and simulated using Quartus II 9.1 sp1 web edition [14]. The RCAS is modeled in the behavioral manner, while the remaining designs are implemented using the structural code with RCAS block as component.

Fig. 6 illustrates the simulations of our RCAS block with inputs: *X*, *Y*, *Cin*, control signal *AS*, and constant inputs '*zero*', and outputs *ASout, Cout*, *SD*, *G*1(a copy of *Y*) and *G*2. Note, that every input combination has an expected (unique) output pattern. This, in addition to the correctness of behavior, confirms reversibility of the RCAS function.

We present the simulation result of 8-bit square root circuit in Fig. 7. The simulator used random values (decimal) for an input *X*. The node *q* (highlighted) shows the decimal value of square root of the input *X*, while the output *r* represents the remainder value. For example, for the input *X* = 232 (decimal), the square root quotient is 15 and remainder *r* is 7.The number of inputs/outputs is 50. We observe total number of garbage bits is 38 which comply with our theoretical analysis. However, only 4 garbage outputs are shown here. The rest 34 garbage bits and some constant inputs are hidden due to space limitation.

VI. CONCLUSION AND FUTURE WORK

In this paper, we present a novel design methodology to implement reversible square root circuit with an array structure. As a building block, we first created a reversible controlled adder/subtractor module performing reversible 2's Complement addition/subtraction. The quantum cost of this module was 10. Next we implemented an array structure with the RCAS modules to perform square root operation. The methodology was generalized to realize square root circuits of

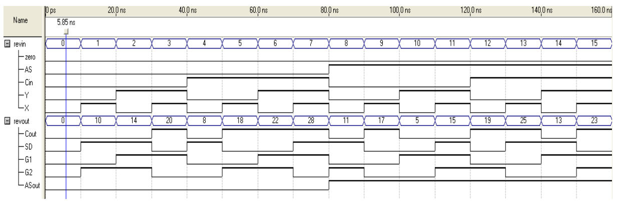

Figure 6: Simulation result of RCAS for square root circuit

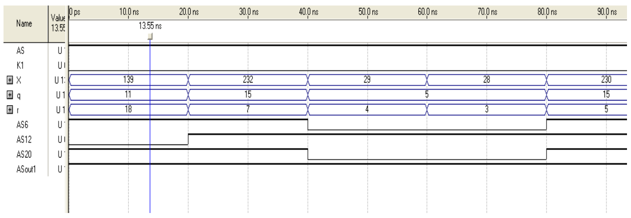

Figure 7: Simulation result of 8-bit Reversible Square Root circuit

any size with less quantum cost. The benchmark circuit for 8-bit square root in (sqrt8) [13] utilizes 40 multiple controlled Toffoli gates with overall quantum cost 622. In our approach, the number of reversible gates (Peres and CNOT) is 94 and overall quantum cost is only 214. Note, design in [13] requires less number of gates as this uses Toffoli gates with more control lines (above 5 controls) which has higher quantum cost (QC = 125 for Toffoli with 6 controls) whereas we use only Peres (QC=4) and CNOT (QC=1) gates. Hence, we rely on quantum cost as comparison parameter and in that respect our approach is better (65% improvement).

Since quantum circuits are reversible in nature and a promising candidate alternative to classical circuits, any development in reversible circuits will play an important role. Hence, we attempt to realize classical arithmetic circuits in reversible embedding. In future, we will extend our work to include reversible multipliers and arithmetic logic units.

REFERENCES

[1] E. Fredkin, T. Toffoli, "Conservative Logic", Intl. Journal of Theoretical Physics, Vol. 21, No. 3-4, pp. 219-253, 1982.

[2] A. Peres, " Reversible logic and quantum computers," Phys. Rev. A, Gen. Phys., Vol 32, No. 6, pp. 3266-3276, Dec. 1985.

[3] D. Große, R.Wille, G. Dueck and R. Drechsler, "Exact Multiple Control Toffoli Network Synthesis with SAT Techniques", IEEE Transactions on CAD, Vol. 28, No.5, pp. 703-715, 2009.

[4] D.Maslov, G.W.Dueck and D. M. Miller, " Techniques for the Synthesis of Reversible Toffoli Networks", ACM Trans. on Design Automation of Electronic System, Vol. 12, No.4, Sept.2007, pp. 42:1-42:28.

[5] P. Gupta, A. Agrawal and N.K. Jha, " An Algorithm for Synthesis of Reversible logic Circuits", IEEE Transactions on CAD of Integrated Circits and Systems. Vol. 25, No.11, pp. 2317-2330, 2006.

[6] R. Wille and R. Dreschler, "BDD-based Synthesis of Reversible Logic Circuits for Larger Functions", Proc. Design Automation Conference, pp.270-275, July 2009.

[7] J. E. Rice, K. B.Fazel, M. A. Thornton and K. B. Kent, "Toffoli Gate Cascade Generation Using ESOP Minimization and QMDD-based Swapping", Proc. Intl. Symposium. on MVL, pp. 63-72, May 2009.

[8] H. Thapliyal and M.B Srinivas, " Novel Design and Reversible Logic Synthesis of Multiplexer Based Full Adder and Multipliers", 48th Midwest Symp. on Circuits and Systems, Vol. 2, pp. 1593-1596, 2006.

[9] M. Haghparast and K. Navi, " Design of a novel reversible multiplier circuit using HNG gate in nanotechnology", American Journal of Applied Sciences, Vol.5, 2008.

[10] H. Thapliyal and N. Ranganathan, " Design of Efficient Binary Subtractors Based on a New Reversible Gate", Proc. of 2009 IEEE Computer Society Annual Symposium on VLSI, pp. 229-234.

[11] S. Samavi, A. Sadrabadi, A. Fanian, " Modular array strucure for no-restoring square root circuit", Journal of Systems Architecture, vol. 54, pp. 957-966, 2008.

[12] V. Carl Hamacher, Safwat G. Zaky and Zvonko G. Vranesic, Computer Organization" New York : McGraw-Hill, ©1984, ISBN: 0072320869.

[13] http://revlib.org/, Reversible benchmark circuits.

[14] https://www.altera.com/download/software/quartus-ii-we/9