International Journal of Computer Applications (0975 – 8887)
Volume 93 – No 14, May 2014

Reversible Architecture of Computer Arithmetic

Sayeeda Sultana
Department of Electrical and Computer Engineering
McGill University
Montreal, Canada

Katarzyna Radecka
Department of Electrical and Computer Engineering
McGill University
Montreal, Canada

ABSTRACT

Reversible logic plays an important role in emerging low power designs and quantum computing. This paper presents an efficient way to realize reversible arithmetic circuits especially targeting toward reversible arithmetic logic unit (RALU). In literature for reversible logic, not a significant advancement is found in integrating both logical and arithmetical functions, commonly known as arithmetic logic unit (ALU), a key feature of any computing system architecture. Here, this work presents a novel reversible arithmetic logic unit (ALU) performing basic functions similar to classical ALU such as addition, subtraction, AND, OR and XOR operations. Additional functions such as, NAND, NOR, XNOR and logical functions with single input inverted, overflow detection and comparison can also be performed with this design. The integration of these operations in single module using less number of control signals is not available in any of existing approaches. The design and analysis based on different parameters of reversible circuits – number of gates, garbage bits and quantum cost as well as simulation results are presented here. The proposed design offers efficient programmability and more flexibility than other methods.

General Terms

Arithmetic design and logic structures.

Keywords

Arithmetic logic unit, Reversible logic, Reversible controlled adder/subtractor, Quantum cost.

1. INTRODUCTION

Recently, reversible circuit synthesis has started to emerge as an important topic, bringing alternative solutions to classical networks. The motivation behind reversible computation comes from its theoretical abilities to address burning issues of modern circuit designs such as power consumption and emerging new technologies. Namely, reversible circuits dissipate less energy, and are closely related to quantum circuits, which, in the near future, could become a competitor to current classical circuits. In 1961, Landauer showed that irreversible circuits regardless of the underlying technology always consume power, and consequently dissipate heat at the rate of at least $kTln2$ for each bit of information erase. In the above equation k is Boltzmann's constant and T is the temperature [1]. Further, Bennett showed that, in principle, arbitrarily small or zero energy dissipation is only possible if no information is lost during computation [2]. This holds for reversible circuits as input and output data is processed without losing any of the original information. Further, as every quantum operation is inherently reversible, reversible circuits constitute a subclass of quantum circuits [3]. Thus, any development in this domain can be directly applied to future technologies. Finally, the use of reversible circuits is already found in low power CMOS designs, adiabatic circuits [4, 5], cryptography [6], optical computing [7] and digital signal processing [8, 9] requiring that all the information encoded in the inputs be preserved in outputs.

A reversible function realizes a unique one-to-one mapping of inputs to outputs with equal number of bits. The irreversible nature of most of the original algorithms makes the synthesis of reversible circuits from irreversible specifications a challenging task. A large part of the existing algorithms, although optimized in garbage bits and gate counts, are restricted to small functions, while some approaches successfully address large functions but are costly in terms of gate count, additional lines and quantum cost [10-14]. A synthesis solution for large reversible circuits is presented in [15] by avoiding ancilla bits. A recent survey enlightened the key features, motivation and developments in the area of reversible logic synthesis [16].

In addition to the reversible realization of logical functions, a great deal of work has been done aiming to implement the basic reversible arithmetic units such as adders, subtractors and multipliers by finding a direct translation from classical truth table to reversible forms using basic standard reversible gates as well as dedicated newly proposed gates [17-24]. Apart from that we also see quantum/reversible circuits for arithmetic operations from addition to multiplication and modular exponentiation [25-28], playing an important role in quantum Shor's algorithm.

In this work, an efficient and versatile reversible arithmetic logic unit (RALU) is presented, which is very close to its classical counterpart. Integration of logical functions or arithmetic units or both is still a challenge. However, a reversible computing architecture with the instruction set, control logic and address calculation has been demonstrated in recent work [29]. An integrated logic unit using approach in [12] performs eight logic functions. In benchmark circuits [30] performing several logic and arithmetic functions in one module can be found. One such example is a unit, which performs AND, OR and XOR only. The other benchmark circuit named mini ALU using BDD-based method [14] includes OR, AND and addition operation. Some designs generated from SyRec programming language incorporate multiplication and division too [30]. An arithmetic unit proposed in [31] performs ADD and SUBTRACT operations as well as increment and decrement of the input by one. A V-shaped low power reversible ALU is developed in [32] for programmable reversible/quantum computing, which performs modular arithmetic like addition, subtraction, negative subtraction, XOR and no operation (NOP). However, the modular arithmetic result does not reflect the overflow (carry out) of arithmetic operations. Moreover, this design does not include logic functions more common to classical ALU such as AND/NAND or OR/NOR. Recently a reversible

ALU design is proposed in [33] which include many operations close to classical ALU design.

In this work, a new integrated module of a RALU is introduced, which encapsulates most of the operations in classical realization with less number of control lines. This module intends to perform the basic mathematical operations of addition, subtraction, as well as logic operations AND and OR. Further, an XOR function (not available in classical ALU), which is very useful in reversible circuits, is also included here. Finally, some negated logical functions such as NAND, NOR and XNOR including implication are realized in this design.

In this approach, the basic RALU operates on single-bit data, which is capable to realize various arithmetic functions, and can be cascaded into an n-bit design. Later the modifications are added to the RALU to detect overflow and to perform comparison (set-less-than) operation to detect whether a number is less than another number. Thus the proposed design includes more functions with less number of lines and acceptable quantum cost.

2. 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 fan-outs 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 irreversible specification aims on embedding an arbitrary (irreversible) function with *I* inputs and *O* outputs (generally $I \neq O$) to a reversible implementation constructed solely from reversible gates. Often extra I/O signals must be added, where the extra *A* inputs are referred as *ancilla* bits, while the additional *G* outputs are *garbage* bits.

The cost of a reversible circuit is determined in terms of gate counts, number of garbage bits and overall quantum cost. In fact, the quantum cost plays 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 performed by NOT, CNOT and controlled V or V^+ gates in order to realize this gate.*

Many reversible gates have been proposed over the years such as basic NOT, CNOT, Toffoli [34], as well as Fredkin [35] and Peres [36]. Recently, some new reversible gates targeting specific implementations such as adder, subtractor, multiplier etc. have been introduced [17-24]. In this work basic reversible

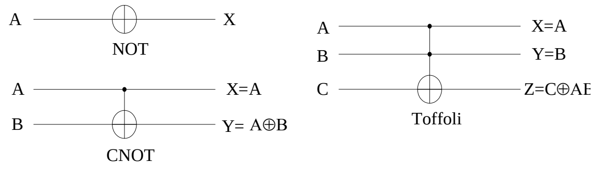

Figure 1: Standard reversible gates

gates such as CNOT, Peres and Fredkin gates are used for the construction of basic RALU design. A brief introduction of some standard reversible gates is presented below.

A CNOT gate shown in Fig.1 is a 2x2 gate with a single control input *A*, which inverts the second input *B* when its value is true. This gate is also known as a Feynman gate. A

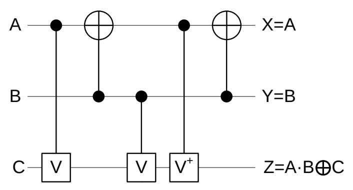

Figure 2: Quantum Implementation of Toffoli gate

multiple-control 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 3x3 Toffoli gate is 5, Fig.2. A Toffoli gate implements a function $Z = AB \oplus C$ with two control inputs *A* and *B*, which are copied at the outputs *X* and *Y*. That way, they can fan-out signals *A* and *B* to the rest of the circuit. Otherwise, if values of *X* and *Y* are not used at other places of the circuit, they are treated as garbage outputs.

A 3x3 Toffoli gate is *universal* since any reversible function can be realized by cascading this gate only. For example, AND, NOT, XOR can be obtained from Toffoli gate *T* (*A*,*B*,*C*):

$$AND: T(A, B, 0) = AB$$
$$NOT: T(A, 1, 1) = A'$$
$$XOR: T(A, 1, X) = A \oplus X$$

A multiple controlled 3x3 Fredkin gate [35], Fig.3, is a controlled swap gate with two target lines. This gate realizes the mapping of the inputs (*A*, *B*, *C*) to the outputs ($X = A$, $Y = A'B + AC$, $Z = AB + A'C$). Note, that the values of the target lines are interchanged if the control lines are set to 1. The quantum cost of this gate is 5 as assumed in [31].The Fredkin gate is an important part of many arithmetic circuit designs.

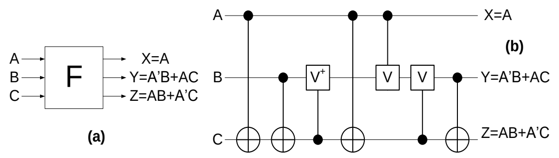

Figure 3: Fredkin gate

Another gate commonly used in reversible implementations, is the 3x3 Peres gate, Fig.4. This gate implements the mapping of

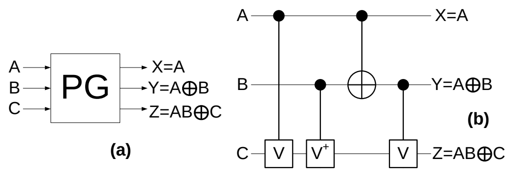

Figure 4: Peres gate

the inputs (*A*, *B*, *C*) to ($X = A$, $Y = A \oplus B$, $Z = AB \oplus C$). Its quantum cost is 4, which is lower than the two gates previously presented. The main advantage of a Peres gate is its capability to implement a half adder functions using only one instance of the gate when $C=0$.

3. REVERSIBLE ARITHMETIC AND LOGICAL OPERATION BASICS

An ALU is an integral module of multiple one- and two-input arithmetic and logic functions. Instead of constructing several single-function circuits this integrated module offers programmability with less gate cost. However, the incorporation of several functions into a single unit requires additional control lines and circuit resources. In this paper, the implementation of a reversible version of a conventional irreversible ALU is presented. For the ALU design special attention should be made on including as many arithmetic and logic operations as possible in a simple design with maximum efficiency and minimum possible cost. Hence, the reversible ALU presented in this paper includes most operations available in conventional irreversible ALU. The stepwise development of the proposed RALU is described next.

3.1 The Logical Operations

In a classical design the most common logical functions included in basic ALU are AND and OR. In proposed RALU the reversible equivalent of these two functions are AND: (A, B, ‘0’ $\rightarrow A$, B, $0\oplus AB$) implemented with single Toffoli gate and OR: (A, B, ‘0’ $\rightarrow A$, B, $0\oplus A\oplus B\oplus AB$) implemented with one Toffoli gate and two CNOT gates. However, to ensure reversibility the AND and OR embeddings require some extra signals as garbage outputs. Another important logical operation, i.e., the bitwise *exclusive-OR* (XOR), which is elementary in reversible logic is also added. Extending ALU by this operation is obvious, and increases the flexibility and applicability of RALU. Moreover, suitable control signal facilitates performing NAND, NOR and XNOR operations too. For example, for the implemented function $0\oplus A\oplus B\oplus C_{nt}$, when control C_{nt} is false then it performs bitwise XOR and when C_{nt} is true then we get XNOR operation.

3.2 The Arithmetic Operations:

The two basic arithmetic operations included in any ALU are an addition and a subtraction of two binary numbers. The main problem in designing reversible adder or subtractor is that the function is not bijective, and hence like other designs in literature, it is important to find proper reversible embedding with the aid of extra signals. In proposed RALU, an adder and a subtractor are implemented in a single module with a control signal. The structure of the reversible controlled adder/subtractor (RCAS) module presented in next section is realized in reversible embedding, performing 2’s Complement computation, and execute addition or subtraction based on a control signal.

An adder/subtractor block is a combinational circuit, which adds or subtracts two binary numbers X and Y depending on the value of the input control signal. For addition, a general full adder block adds three bits X, Y, Z and generates two outputs: *sum*(S) and *carry-out* (Co) according to the logic equations $S = X\oplus Y\oplus Z$ and $Co = XY\oplus Z(X\oplus Y)$. A subtractor performs a subtraction on three bits X (*minuend*), Y (*subtrahend*) and Z (*subtrahend*), and results in a *difference* D and a *borrow* B_o calculated according to the logic equations $D = X\oplus Y\oplus Z$ and $B_o = X'Y\oplus Z(X\oplus Y)'$ [24].

3.3 Reversible Controlled adder/subtractor

The concept of reversible controlled adder/subtractor presented 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 $X+Y'+1$, i.e., 2’s Complement. The block diagram of a general irreversible controlled adder/subtractor is shown in Fig.5.

Note, that as 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 cannot be used directly as a reversible element. Therefore, to create a cascadable reversible CAS module, some garbage bits must be added to the original irreversible CAS structure. A second issue, addressed in the reversible CAS construction is the lack of support for fan-out signals in reversible designs. In

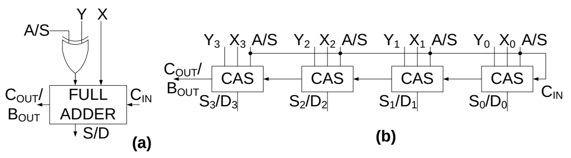

Figure 5: Simple adder/subtractor design

the classical adder/subtractor implementations, the same control signal A/S is fanned-out to all CAS blocks, Fig.5 (b). To overcome this shortcoming, one of the garbage bits, i.e., A/S_g, is used to provide a copy of a control signal from one module to the next.

With the questions of fan-outs and I/O count compatibility resolved, the reversible controlled adder/subtractor (RCAS) block is constructed using reversible gates. The full adder is implemented by cascading two Peres gates (highlighted), which results in generating the target outputs S/D and C_{out} where $S/D = A/S\oplus Y\oplus X\oplus C_{in}$ and $C_{out} = X(A/S\oplus Y)\oplus C_{in}(A/S\oplus Y\oplus X)$, in addition to garbage outputs A/S_g, $g1$ and $g2$, Fig. 6 [37]. When A/S is set to 0, then CNOT (first gate, Fig. 6) 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. The quantum cost of the RCAS module is 9 (8 for two Peres plus 1 for CNOT gate).

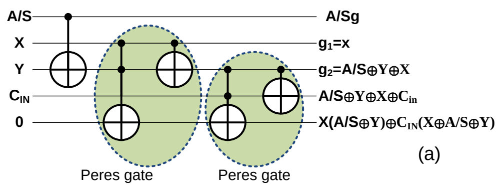

Figure 6: Reversible Implementation of controlled adder/subtractor

RCAS blocks can be cascaded to construct an n-bit adder/subtractor. In the proposed architecture, the A/S_g and C_{out} outputs of a previous stage propagate to the input control signal A/S and the C_{in} of the next RCAS. For a 2’s Complement subtraction $X+Y'+1$, an extra value 1 needs to be added to the least significant bit position (LSB) of the result. Hence, the original LSB block is modified by adding a CNOT gate between A/S and a constant ‘0’ ancilla bit. Note that when A/S is set to 0 for an addition, a value 0 is passed through the CNOT gate to the RCAS$_0$ resulting in the addition of X_0 and Y_0 only. When A/S is set to 1 for the subtraction, a value 1 is added to X_0+Y_0' as required by a 2’s Complement notation. The addition of CNOT gate increases quantum cost by 1.

If a result of an n-bit addition/subtraction does not fall within the allowed range, then an arithmetic overflow occurs. Although, when adding unsigned numbers, the output carry-out C_{out} signal coming from the most significant RCAS block serves as an overflow indicator, for the signed numbers the carry-out at the sign-bit position is not sufficient. In the case of signed numbers the overflow can occur, when adding two

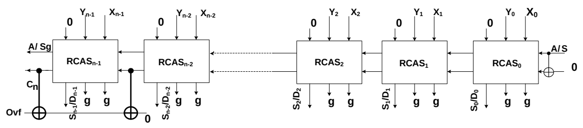

Figure 7: Reversible Implementation of controlled adder/subtractor with overflow detection

numbers of the same sign with the carry out signal from the MSB position being 0, i.e., not indicating the overflow.

When considering the 2’s Complement addition (subtraction), the overflow occurs if the carry-in (borrow-in) to the most significant RCAS is the different than the carry-out (borrow-out) generated by that block, i.e., $Ovf = C_{n-1} \oplus C_n$. To implement the above check two additional CNOT gates at the outputs of the two most significant RCAS modules are added. The first CNOT gate placed at the C_{out} of RCAS$_{n-2}$ block provides a copy of carry input C_{n-1} to the RCAS$_{n-1}$. The second CNOT performs XOR of C_n and C_{n-1}, where C_n is the carry output from an *n*th bit RCAS. If *Ovf* is equal to 1, then the addition is incorrect, falling outside the assumed range. An *n*-bit signed numbers (2’s Complement) reversible adder/subtractor with overflow detector is given in Fig. 7.

The proposed reversible *n*-bit adder/subtractor requires $3n+1$ gate (Peres and CNOT), $2n+1$ garbage bits, and has the quantum cost of $9n+1$. Table 1 summarizes the comparison of the proposed 16-bit design with binary adder/subtractors presented in [24]. These are three designs constructed from Fredkin, Peres and TR gates, and differ in the number of reversible gates, garbage outputs, ancilla inputs and quantum cost. As the Add/Sub- Design III implemented with Peres and CNOT gates clearly outperforms the Design I (Fredkin and CNOT), and the Design II (TR and CNOT gates) [24], the Design III is mainly considered for comparison with proposed designs (n-bit adder/subtractor with/without the overflow detector). The design presented in this work is better than adder/subtractor circuits [24] in terms of reversible gates, garbage outputs and the quantum cost. For example, for an *n*-bit Design III [24] the quantum cost is $10(n-1)+6 = 10n-4$ with $(n-1)$ full adder/subtractor each contributing 10 to the quantum cost, and one LSB half adder/subtractor with quantum cost of 6. In contrast, the proposed *n*-bit design has a cost of $9n+1$. Hence, the improvement in the quantum cost of this design is $n-3$ for $n > 3$. The gain grows with increasing size of the adder/subtractor.

Table 1 Comparison of different adder/subtractor

Designs	Reversible gates	Garbage outputs	ancilla	Quantum cost
Add/Sub- I [24]	124	78	47	327
Add/Sub- II [24]	63	47	16	218
Add/Sub- III [24]	63	47	16	156
Proposed Design	49	33	17	145
With overflow	51	33	18	147

4. REVERSIBLE ARITHMETIC LOGIC UNIT

In proposed reversible ALU design the arithmetic and logical operations are first performed in parallel, and then the desired result is selected by a multiplexer (MUX)- a circuit that chooses one out of several inputs depending on a control. The two steps of the design are defined as: function generation and function selection, Fig. 8.

4.1 Function Generator:

The first module of the proposed reversible ALU generates four arithmetic-logic functions in parallel as well as transmits

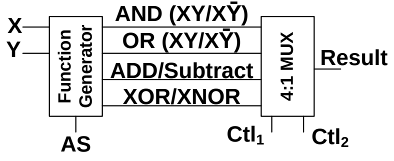

Figure 8: Reversible ALU two steps block diagram

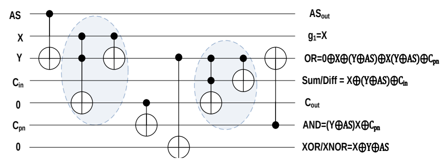

Figure 9: Reversible ALU function generator

an input *X* unchanged. Another operand *Y* is controlled by a control signal *AS*, which determines the inverted operation or a subtraction. The implementation of this generator block is presented in Fig. 9. The control signal C_{pn} defines the functionality of AND and OR when C_{pn}=0), otherwise NAND or NOR function is performed. Next if the control signal *AS* is ‘0’ then the device performs the following operations simultaneously: AND/NAND, OR/NOR, addition and XOR. On the other hand, if the control signal *AS* is ‘1’ then results for AND/NAND with single input inverted, OR/NOR with single input inverted, Subtraction and XNOR are generated, Table 2.

The RCAS in Fig. 6 is used for arithmetic operations.

Table 2: RALU Operations with control inputs

Cpn	AS	Ctl2	Ctl1	Operation
0	0	0	0	AND
0	0	0	1	OR
0	0	1	0	ADD
0	0	1	1	XOR
1	0	0	0	NAND
1	0	0	1	NOR
0	1	0	0	XY’
0	1	0	1	Y->X
1	1	0	0	X->Y
0	1	1	0	SUB
0	1	1	1	XNOR

However, in order to implement with fewer number of gates for extended functions AND, OR and XOR, two CNOT gates in between two Peres gates (AND and XOR) and one extra CNOT gate after the 2nd Peres gate are added, Fig. 9.The quantum cost of this function generator is 12.

4.2 Function Selector

To select a desired function output a 4:1 reversible multiplexer shown in Fig. 10 is proposed in this work based on embedded reversible specification and is realized using Positive Davio expansion, which confirms minimum number of lines i.e. Result=$(F_1 \oplus Ctl_1(F_1 \oplus F_2)) \oplus Ctl_2(F_1 \oplus Ctl_1(F_1 \oplus F_2) \oplus F_3 \oplus Ctl_1(F_3 \oplus F_4))$.The number of reversible gates for this MUX is 6, while number of lines is 6. No ancilla bit is added. The quantum cost is 18.

Another implementation of a function generator is obtained by using Fredkin gates. A Fredkin gate itself is a reversible equivalent of 2:1 *MUX*. Hence, an alternative to 4:1 *MUX* proposed in the previous paragraph is to use three 2:1 *MUXs*

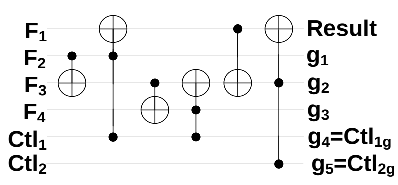

Figure 10: Reversible ALU function selector (MUX)

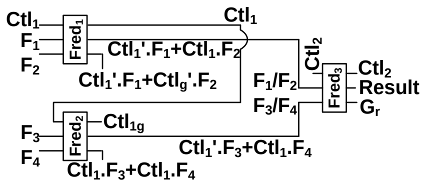

Figure 11: Reversible ALU function selector using Fredkin gates

(Fredkin gates). The advantage of having a Fredkin gate multiplexer lies in a smaller quantum cost (15) as well as less logic depth. Further, a Fredkin gate generates the selected function as well as other functions available in the outputs. For example in Fig. 11, if the controls are set as Ctl_1='0' and Ctl_2 ='1' then *Fredkin*1 gate selects F_1 in line 2 while F_2 is also available at line 3. Similarly *Fredkin*2 selects F_3 in line 4 and F_4 in line 5. The final *Fredkin*3 gate selects F_3 as a resulting output while F_1 is also available as garbage output G_r. Thus, though it selects only one function as a result in target line, the rest of the functions can be obtained from its garbage outputs.

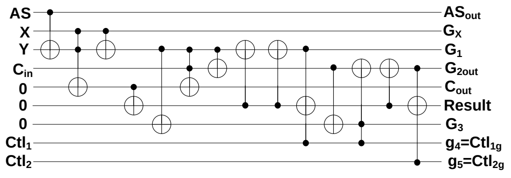

Figure 12: Reversible ALU design I (4:1 MUX)

4.3 1-bit Reversible Arithmetic Logic Unit (RALU):

Two designs of reversible arithmetic logic unit for single bit are proposed in this section, one with 4:1 MUX and another using Fredkin selector. The design 1 of a 1-bit RALU is shown in Fig. 12. The circuit comprising 4:1 multiplexer requires two Peres gates, 3 Toffoli gates and 7 CNOT gates (total 12 reversible gates). The number of lines is 9 and overall quantum cost is 30. On the other hand, the design 2 with Fredkin multiplexer, Fig. 13, requires two Peres gates, 4 CNOT gates and three Fredkin gates (9 gates in total) with overall quantum cost 27. In Fig. 13(b) all outputs as well as target function output (XOR) are available to monitor and the garbage outputs (AND, SUM and OR) for control inputs Ctl_{1g}=1 and Ctl_{2g}=1 are also shown.

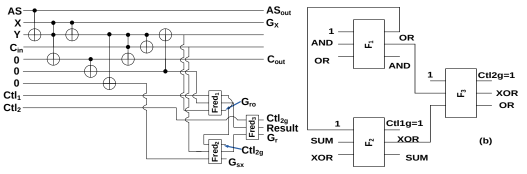

Figure 13: RALU design II (using Fredkin selector)

Note, that all RALU blocks when placed in an *n*-bit reversible arithmetic logic unit in reversible embedding must comply with reversibility properties, Def. 1, while preserving the correctness of the execution of the arithmetic and logical operations.

4.4 n-bit RALU:

In the proposed circuit, cascading a number of the 1-bit RALU modules, it is possible to construct RALU of any size. The copy of all control signals (AS, Ctl_2, Ctl_1) available at the module outputs are utilized for the next stage ALU operation. Thus RALU guarantees the generation of all required fan-out signals for controlling the selection of functions through garbage signals and thus offers the minimal cost of the implementation.

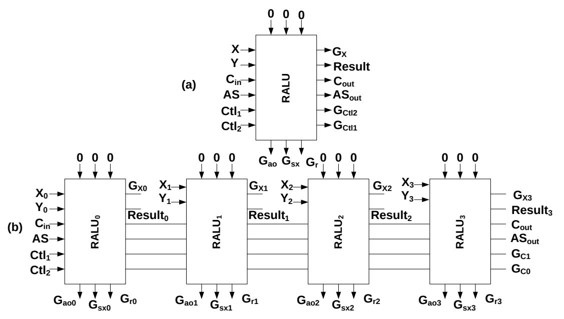

Figure 14: 4-bit RALU

A 4-bit reversible arithmetic logic unit with inputs $X_3X_2X_1X_0$ and $Y_3Y_2Y_1Y_0$ is presented in Fig. 14 (b). The basic block, i.e., the RALU module is shown in Fig. 14(a). Control signal *AS* transmits the true or inverted copy of input signal *Y* as well as defines the addition or subtraction operation. The other input *X* is transmitted unchanged at the output as garbage G_X. All the functions outputs are available at each RALU module outputs as G_{ao}, G_{sx} and G_r. The output *Result* presents the desired function selected by the control signals. While cascading for the addition or subtraction operations, the carry out C_{out} of a previous stage is propagated to the next stage. Thus four outputs are reused to provide next stage signals. This is a minimization of garbage outputs. Hence, a 4-bit reversible implementation of arithmetic logic unit requires 24 lines. Note that for subtraction operation the input carry in C_{in} should be set to '1'.

4.5 Analysis and Comparisons of circuit parameters for n-bit RALU:

By cascading RALU blocks the same way as in Fig. 14 any size of the arithmetic logic operation can be easily realized. The *n*-bit RALU with five basic arithmetic-logical operations requires elementary reversible logic gates (Peres, Toffoli, Feynman gates). Each RALU module with a 4:1 MUX requires 12 gates with quantum cost 30. Hence, for the *n*-bit realization the number of gates is 12*n*, and the quantum cost is 30*n*. On the other hand, with Fredkin multiplexer, the number of reversible gates for an *n*-bit design is 9*n*, and quantum cost is 27*n*. The number of garbage bits is also linear in the size of inputs (5*n*+4). Note, that the addition of the control and the combination of multiple functions do not necessarily require a large number of gates and ancilla.

Table 1 Comparison of 1-bit ALU

Circuit	Operation	# of lines	# of gates	QC
Logic Unit, Gupta [12]	AND, OR, NAND, NOR, XOR, const.	5	18	114
Mini ALU [30]	AND, OR, ADD, no-op.	10	20	60
Proposed Design	AND, OR, XOR,NAND, NOR, AND/OR with inverted input, ADD, SUB	9	9	27

Table 4: Different 32-bit reversible ALU realizations

Circuit	Operations	Ancilla	Gates used	Lines	Gates	QC
Logic Unit [30]	AND, OR, XOR	yes	CNOT, Generalized Toffoli	299	571	1223
		no		203	385	6562
ALU SyReC [30]	ADD, SUB, MULT, DIV	yes	CNOT, Generalized Toffoli	331	15950	1336477
		no		235	15764	1851487
Simple ALU SyRec [30]	ADD, SUB, MULT, XOR	yes	CNOT, Generalized Toffoli	331	4413	27009
		no		235	4227	152852
V-Shape [32]	Modular arithmetic (ADD, SUB, NSUB), XOR, no-op	no	CNOT, Toffoli and Fredkin	69	190	694
ALU [33]	ADD, SUB, OR, NOR, AND/NAND (or XOR/XNOR)	yes	CNOT, Fredkin, HNG, MRG/POAG	196	254	830
Proposed design	AND, NAND, OR, NOR, ADD, SUB, XOR, XNOR, implication	yes	CNOT, Peres, Fredkin, Toffoli	164	288	864

In literature, there are different realizations of logic or arithmetic units as benchmark circuits [30] implementing different operations. Table 3 compares various reversible circuit parameters of proposed design with existing realizations for a 1-bit RALU. Table 4 presents a summary of existing 32-bit reversible ALU or LU realizations with the proposed design. Note that the operations performed by each method are not the same; hence a solid comparison is not possible. The multiplication or division operations are not included as the complexity of the design increases and that's why in CMOS design usually these operations are not integrated in ALU. Thus, the proposed implementation is very close to the classical ALU with acceptable cost. For a 32-bit realization, this design is better than Logic Unit, which does not even include arithmetic operations.

The V-shape design [32] is more economical and efficient for programmable reversible *computing*. From Table 4 it can be observed that the design contrast the proposed RALU in many operations. For example, the method in [32] calculates a modular addition and subtraction, so no carry output is considered, while the design presented in this work calculates complete result having a sum and carry output to indicate arithmetic overflow condition. Moreover, the proposed *RALU* includes more logic operations such as *AND/NAND, OR/NOR*. This requires an extra circuitry, which is absent in method [32]. To present a meaningful comparison a modification to original reversible ALU is performed to include the operations performed in [32], and the extra logic functions in this new design are discarded. In Fig. 15, the new design consisting of original reversible controlled adder/subtractor (*RCAS*: quantum cost 9, one *CNOT* and two Peres gates), one Toffoli, one *CNOT* and one Fredkin gate is presented. The functions with different control signals are presented in Table 5. The overall quantum cost is 20. Hence, the quantum cost for 32-bit design 640, which is less than the V-shaped design (QC= 694].

Recently two designs of reversible ALU were presented in [33] based on two newly proposed gates MRG and Peres-AND-OR (PAOG). These designs perform similar arithmetic and logical functions to ours. For example, a reversible ALU with MRG and HNG gates perform OR, NOR, XOR, XNOR, ADD, SUB operations, and a reversible ALU with *PAOG* and *HNG* gates perform AND, NAND, OR, NOR, ADD and SUB operation. Note that the first design excludes AND, NAND operation while the 2nd design excludes XOR/XNOR operations. However, our RALU integrates all the functions of these two designs. The total cost of an *n*-bit ALU in [33] is 26*n*-2. To make a fair comparison, the excess functions

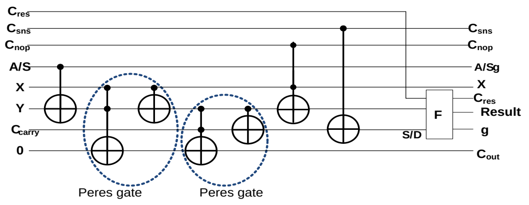

Figure 15: Proposed RALU comparable to [32]

(AND/NAND or XOR/XNOR) are not considered, and then the total quantum cost is calculated as 21*n* for a design with a Fredkin selector and 23*n* for a design with a multiplexer. For a 32-bit reversible ALU proposed design has a quantum cost of 672, while the design in [33] has quantum cost 830. Moreover, for a 1-bit ALU the design in [33] requires 10 lines whereas this new RALU needs 9 lines (actually 8 lines if one function is not considered for a proper comparison). Thus 2 lines are saved per bit of an RALU. Thus, proposed RALU is more economical than other methods.

4.6 RALU with overflow detector and set-less-than function

When RALU performs an addition or a subtraction operation, one should consider the allowed range for the given number representation used. This means that it is important to check whether the result is within the acceptable range. For an unsigned number, the C_{out} output represents the overflow of the operation. However, in case of signed numbers' operations (2's Complement Computation) a controlled adder/subtractor requires an extra circuit to monitor an overflow. Similar to the

Table 5: RALU (Fig. 15) operations with control inputs (X is unchanged)

C_{res}	Csns	Cnop	AS	Ccarry	ALU Operation
1	0	0	0	0	$Y +_n X$ *ADD*
1	1	0	1	0	$Y -_n X$ *SUB*
1	0	0	1	1	$X -_n Y$ *NSUB*
0	0	0	0	0	$Y \oplus X$ *XOR*
0	0	1	0	0	Y *NOP*
0	0	1	1	0	$\overline{Y}$
0	0	0	1	0	$\overline{Y \oplus X}$
1	0	0	0	1	$Y +_n X +_n 1$
1	0	0	1	0	$X -_n Y -_n 1$

RCAS design with the overflow detector, Fig. 8, the proposed RALU is modified at the most significant bit position. A copy

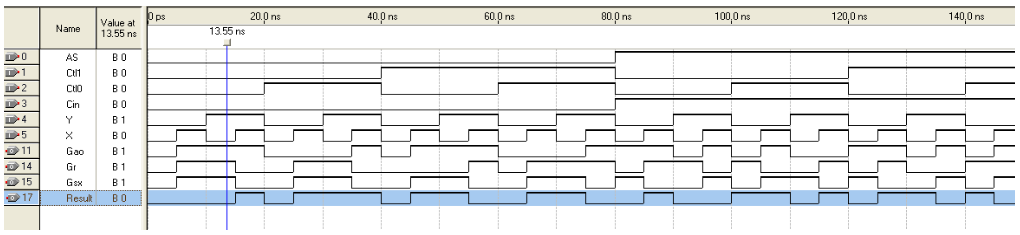

Figure 17: Simulation result of 1-bit RALU using Fredkin MUX

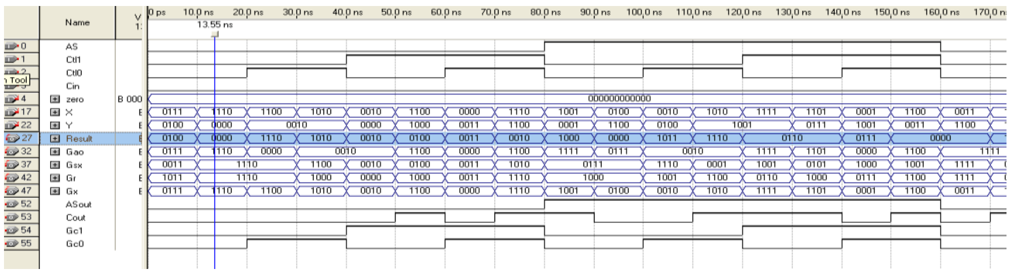

Figure 18: Simulation result of RALU_Fredkin_4bit

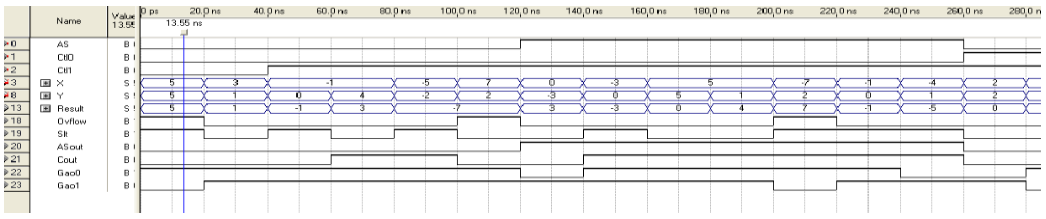

Figure 19: Simulation result of 4-bit RALU with overflow and Set less than

of a carry-input of the RALU block is obtained using a $CNOT_{in}$ gate, and the carry-in is *XOR*-ed with carry output of the block ($CNOT_{ovf}$) to detect overflow, Fig.16.

This overflow detector can be employed to add another operation to original RALU. The set-less-than is usually available in classical arithmetic logic unit, and is used to compare if a number *X* is less than the number *Y* (*X*<*Y*). As discussed earlier, during the subtraction of two signed numbers, i.e., *X*-*Y*, the sign of the result XOR-ed with the overflow signal indicates whether *X* is smaller than *Y*. The copy of a sign bit (most significant digit of a difference) is obtained with $CNOT_{sign}$ gate. The set-less-than output *Slt* is generated by the $CNOT_{slt}$ gate, Fig. 16. The overall quantum cost is increased only by 4 with the inclusion of two functions: the overflow detection and comparison.

5. SIMULATION RESULT

The reversible functionality of all designs is verified through simulations, i.e 1-bit RALU blocks (Fig. 17), 4-bit reversible arithmetic logic circuits (Fig.18) with Fredkin selector and finally RALU with overflow detector and set-less-than operation (Fig. 19). All of the above designs were implemented in VHDL and simulated using Quartus II 9.1 sp1 web edition [39]. The RCAS module is modeled in the behavioral manner, while the remaining designs are implemented using structural code with RCAS block as component.

Fig. 17 shows the simulations of a 1-bit RALU using a Fredkin multiplexer. Note that every input combination has an expected (unique) output pattern. For each control inputs combination, we simulate 4-input patterns of *X* and *Y*. The highlighted '*Result'* represents correct function output values for corresponding control signals. Moreover, the non-selected outputs are available at garbage outputs *Gao*, *Gsx* and *Gr*.

Fig. 18 presents the simulation results of a 4-bit RALU using Fredkin gates, with 24 inputs and 24 outputs. As before, the two random values are the inputs *X* and *Y,* the output '*Result'*. The time interval 0-20ns represents bit-wise AND (*XY*), 20-40ns bit-wise OR, 40-60ns Sum, 60-80ns XOR, 80-100ns AND with *Y* inverted (XY'), 100-120ns OR with *Y* inverted (*X*+*Y*'), 120-140ns Subtraction and finally 140-160ns XNOR operation. The garbage outputs *Gx*, *ASout*, Gc_1 and Gc0 are

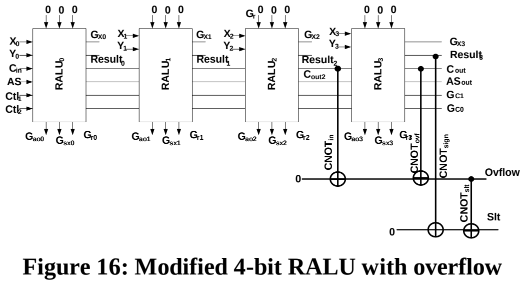

Figure 16: Modified 4-bit RALU with overflow detection and set-less-than operation

the copies of inputs X, AS, Ctl1 and Ctl0 respectively. The garbage outputs Gao, Gsx and Gr represent non-selected outputs (AND/OR for values of Ctl0), (*Sum/XOR* according to Ctl_0) and *AND/Sum* or *OR/XOR* (based onCtl_1), which is not selected by '*Result'*.

The simulation results of RALU with the overflow detector and the set-less-than function are shown in Fig. 19. The node '*Result'* shows the outputs according to the control signals for some random values of inputs set by the simulator.

6. CONCLUSION

Reversible logic is considered to be compatible with future computing technologies, which dissipate less energy. Finding an efficient reversible implementation of classical computer arithmetic especially the arithmetic logic unit is still a challenging issue. In this paper, the reversible architectures of computer arithmetic logic unit are presented which offer smaller overhead than designs proposed by other authors. Starting with the basic RCAS module, a complete and new RALU is constructed, which is similar to the basic classical ALU. Two different realizations are proposed and then analyzed their effectiveness. This integrated module is better than any existing reversible arithmetic logic unit incorporating more operations. The modular structure of the proposed *n*-bit RALU offers economical and acceptable values of reversible circuit parameters comparable to other benchmark circuits.

7. REFERENCES

[1] R. Landauer, "Irreversibility and heat generation in the computational process". IBM Journal of Research and Development, 1961. 5(3): pp. 183-191.

[2] C. H. Bennett, "Logical reversibility of computation", IBM Journal of Research and Development, 1973. 17(6): pp. 525-532.

[3] M. A. Nielsen, I. L. Chuang, "Quantum Computation and Quantum Information", Cambridge University Press, 2000.

[4] W. C. Athas and L.J. Svensson, "Reversible Logic Issues in Adiabatic CMOS", Workshop on Physics and Computation, 1994, pp. 111-118.

[5] S. Burignat and A.D. Vos, "Test of a Majority-based Reversible (Quantum) 4-bits Ripple Carry Adder in Adiabatic Calculation", 18th International Conference on Mixed Design of Integrated Circuits and Systems, 2011, pp. 368-373.

[6] H. Thapliyal and M. Zwolinski, "Reversible Logic to Cryptographic Hardware: a New Paradigm", CoRRabs/cs/0610089, 2006.

[7] H. Thapliyal and N. Ranganathan, "Mach-Zehnder interferometer based design of all optical reversible binary adder", Design, Automation and Test in Europe, 2012.

[8] M. Skoneczny, Y van Rentergem and A. D. Vos, "Reversible Fourier Transform Chip", 15th International Conference on Mixed Design of Integrated Circuits and Systems, 2008.

[9] A. D. Vos, S. Burignat and M. K. Thomsen, " Reversible Implementation of a discrete Integer Linear Transformation", Journal of Multiple-Valued Logic and Soft Computing, vol. 18, no. 1, pp. 25-35, 2012.

[10] 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, pp. 42:1-42:28, Sept.2007.

[11] D. Grobe, R.Wille, G. Dueck and R. Drechsler, "Exact Multiple Control Toffoli Network Synthesis with SAT Techniques", IEEE Trans. On CAD, vol.28, no.5, pp. 703-715, 2009.

[12] P. Gupta, A. Agrawal and N.K. Jha, "An Algorithm for Synthesis of Reversible logic Circuits", IEEE Trans. on CAD of Integrated Circuits and Sys.vol. 25, no.11, 2006.

[13] V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes. "Synthesis of reversible logic circuits", IEEE Trans. on CAD, 22(6):710–722, 2003.

[14] R. Wille, R. Drechsler," BDD-based Synthesis of Reversible logic for Large Functions" Design Automation Conf., 2009.

[15] N. Alhagi, M. Hawash, M. A. Perkowski, "Synthesis of Reversible Circuits with No Ancilla Bits for Large Reversible Functions Specified with Bit Equations", Proc. of the 40th IEEE International Symp. on Multiple-Valued Logic, pp. 39-45, 2010.

[16] M. Saeedi and I. L. Markov, " Synthesis and Optimization of Reversible Circuits- A Survey", ACM Computing Surveys 2012.

[17] L. Ni, Z. Guan and W. Zhu, "A General method of Constructing the Reversible Full Adder", 3rd Intl. Symp. on Intelligent Inf. Technology and Security Informatics, pp. 109-113, 2010.

[18] 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.

[19] H. Thapliyal, M.B Srinivas, "Novel Reversible TSG gate and its application for designing reversible carry look ahead adder and other adder architectures", Proc. of 10th Asia-pacific computer system architecture Conference, 3740, 2005.

[20] M. Haghparast and K. Navi, "Design of a novel reversible multiplier circuit using HNG gate in nanotechnology", Am. J. Applied Sci., vol.5, 2008, 282.

[21] M. Ehsanpour, P. Moallem, A. Vafaei, "Design of a Novel Reversible Multiplier Circuit Using Modified Full Adder", 2010 Intl. Conf. on Computer Design and Applications, vol.3 .

[22] H. Thapliyal, M.B Srinivas and H.R. Arabnia, "Reversible Logic Synthesis of Half, Full and Parallel Subtractors", Proc. of Intl. Conf. on embedded Sys. and App., June 2005, Las Vegas, pp. 165-181.

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

[24] H. G. Rangaraju, U. Venugopal, K.N. Muralidhara and K. B. Raja, " Low Power Reversible Parallel Binary Adder/Subtractor", Intl. J. of VLSI design & Comm. Sys. (VLSICS), Vol. 1, no. 3, 2010, pp 23-34.

[25] V. Vedral, A. Barenco and A Ekert, " Quantum Networks for Elementary Arithmetic Operations", Phys. Rev. A, vol. 54, no. 1, pp. 147-153, 1996.

[26] S. A. Cuccaro, T. G. Draper, S. A. Kutin and D. P. Moulton, " A new Quantum Ripple-Carry Addition Circuit," quant-ph/0410184, 2004.

[27] Y. Takahashi, S. Tani and N. Kunihiro, " Quantum Addition Circuits and Unbounded Fan-out, " Quantum

Information and Computation, vol. 10, no. 9&10, pp. 872-890, 2010.

[28] I. L. Markov and M. Saeedi, "Constant-Optimized Quantum Circuits for Modular Multiplication and Exponentiation", Quantum Information and Computation, vol. 1. no. 5&6, pp. 872-890, 2012.

[29] M. K. Thomsen, H. B. Axelsen and R. Gluck, " A Reversible Processor Architecture and its Reversible Logic Design", Reversible Computation, Lecture Notes in Computer Science, vol. 7165, 2012, pp 30-42.

[30] Benchmark circuits. http://www.revlib.org

[31] R. Aradhaya, K. N. Muralidhara, B. Kumar, " Design of Low Power Arithmetic Unit Based on Reversible Logic", International Journal of VLSI and Signal Processing Applications, vol. 1, no. 1, pp. 30-38, 2011.

[32] M. K. Thomsen, R. Gluck, H. B. Axelsen, "Reversible arithmetic logic unit for quantum arithmetic", J. Phys. A: Math. Theor., vol. 43, no. 38, 2010.

[33] M. Morrison and N. Ranganathan, " Design of a Reversible ALU based on Novel Programmable Reversible Logic Gate Structures", IEEE Computer Society Annual Symposium on VLSI, 2011, pp. 126-131.

[34] T. Toffoli, " Reversible Computing", Technical Memo, MIT/LCS/TM-151, Boston 1980.

[35] E. Fredkin, T. Toffoli, "Conservative Logic", Int. J. Theor. Physics, vol. 21, no. 3-4, pp. 219-253, 1982.

[36] A. Peres, " Reversible logic and quantum computers," Phys. Rev. A, Gen. Phys., vol 32, no. 6, pp. 3266-3276, Dec. 1985.

[37] S. Sultana, K. Radecka, " Reversible adder/subtractor with overflow detector", Intl. Midwest Symp. on Circuits and Systems (MWSCAS 2011), pages 1- 4.

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

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