2011 11th IEEE International Conference on Nanotechnology
Portland Marriott
August 15-18, 2011, Portland, Oregon, USA

Design of a Reversible Floating-Point Adder Architecture

Michael Nachtigal, *Student Member*, *IEEE*, Himanshu Thapliyal, *Student Member*, *IEEE*,
and Nagarajan Ranganathan, *Fellow*, *IEEE*
Department of Computer Science & Engineering
University of South Florida, Tampa, FL USA

***Abstract*—The study of reversible circuits holds great promise for emerging technologies. Reversible circuits offer the possibility for great reductions in power consumption, and quantum computers will require logically reversible digital circuits. Many different reversible implementations of logical and arithmetic units have been proposed in the literature, but very few reversible floating-point designs exist. Floating-point operations are needed very frequently in nearly all computing disciplines, and studies have shown floating-point addition to be the most oft used floating-point operation. In this paper we present for the first time a reversible floating-point adder that closely follows the IEEE754 specification for binary floating-point arithmetic. Our design requires reversible designs of a controlled swap unit, a subtracter, an alignment unit, signed integer representation conversion units, an integer adder, a normalization unit, and a rounding unit. We analyze these major components in terms of quantum cost, garbage outputs, and constant inputs.**

***Index Terms*—Reversible logic, floating-point, addition, arithmetic, quantum computing**

I. Introduction

Reversible computing differs from conventional computing in that it performs the computation in a logically reversible way: The output of a (fully) reversible circuit always uniquely identifies the input. Circuits can take advantage of this logical reversibility to reduce power by reusing the information instead of discarding it: Landauer showed that any time a bit of information is discarded, it equates to some quantum of energy lost as heat [1]. Moreover, Bennett has given a theoretical model of a reversible computer [2].

Modern computers use conventions for representing non-integer numbers, the most widespread of which is the IEEE754 Standard for Floating-Point Arithmetic [3]. This standard defines binary representation for floating-point numbers of varying precision, giving specific examples of the binary32 (or *single precision*) format, binary64 (or *double precision*) format, and it defines operations on floating-point numbers. Floating-point addition is the most frequently used floating-point operation [4], and yet to our knowledge no attempt has been made at a reversible floating-point adder architecture; only a reversible binary32 floating-point multiplier architecture has been proposed [5].

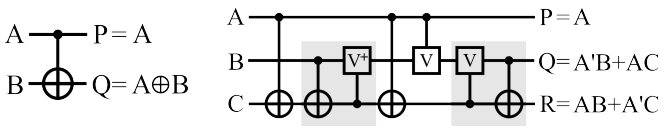

Fig. 1. Quantum implementations and Boolean output functions of the Feynman gate (a) and the Fredkin gate (b). The Feynman gate is often used to implement a simple fan-out, and the Fredkin gate is often used as a reversible 2-to-1 multiplexer, or as a controlled swap unit.

The remainder of this paper is structured as follows. Section II reviews a few important fundamentals of reversible logic design, including coverage of some of the important reversible logic primitive gates. Section III briefly outlines a floating-point addition algorithm and architecture, then gives details about each of the major reversible components of the architecture. Section IV presents our final measurements, a brief analysis of the architecture, and direction for future work. Section V concludes the paper with our list of references.

II. Reversible Logic Primitives

Many traditional logic gates such as the AND, OR, NAND, NOR, and XOR gates are fundamentally irreversible. That is to say that the output combination of any of these gates does not expose the input combination that caused the output. Thus we have a need for primitive reversible logic gates. Researchers already have developed many such gates, including the Feynman, Toffoli, Peres, Fredkin, HNG, and TSG gates [6, 7, 8, 10]. Each gate is defined by the number of inputs and outputs it has (reversible gates must have equal numbers of inputs and outputs) and its Boolean output functions. See Fig. 1 for details on the Feynman and Fredkin gates.

To a quantum implementation of a reversible gate we assign a *quantum cost*. This quantum cost is equal to the sum of the number of Feynman gates, inverter gates, controlled-V gates, and controlled-V^+ gates used in its implementation. The controlled-V and controlled-V^+ gates are two-input quantum gates that perform unitary transformations. The V^+ operator is the adjoint of the V operator. The V and V^+ operators behave as follows:

$$V^+V^+ = VV = NOT$$
$$V^+V = VV^+ = I$$

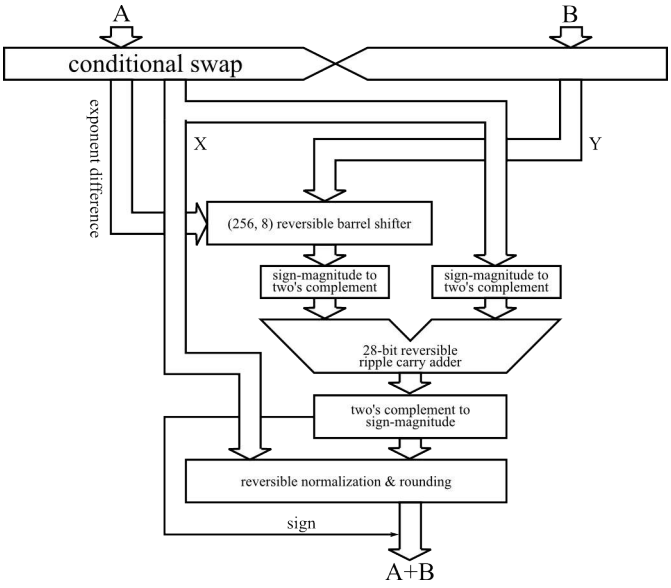

Fig. 2. Our proposed design of a reversible floating-point addition architecture.

A few combinations of two-input quantum gates operating in series can have a combined cost that collapses to unity. These gates are called *merged qubit states* [9]. The gray boxes in Fig. 1(b) are unit-cost merged qubit states.

Reversible circuits can have outputs that go unused in the rest of the design. These outputs are called *garbage outputs* or simply *garbage*. Designers attempt to minimize the number of garbage outputs in designs, because these outputs only serve to maintain logical reversibility. Reversible circuits also can have inputs to which a constant logical 0 or 1 is applied; such inputs are called *constant inputs*, and designers seek to minimize them, too. These two properties —garbage outputs and constant inputs—are not inherent to a particular gate; they depend on the gate's use in a circuit. Take as an example a Feynman gate with the B input tied to 0, a configuration that provides two copies of the A input. If both copies of A are used in the circuit, then the gate has a single constant input, unit quantum cost, and zero garbage outputs.

III. Floating-Point Addition

A. The Algorithm

Given two floating-point numbers (each having sign, biased exponent, and trailing significand fields) to be added, the IEEE754 Standard for Floating-Point Arithmetic details how their sum can be found [3]: First, if the exponents are not equal, the smaller is incremented until it aligns with the larger. To align the floating-point number with the smaller exponent without altering its value, its respective trailing significand must be shifted one place to the right for every time the exponent is incremented. Once the exponents are equal, the significands can be summed. The sum is then normalized and rounded. Fig. 2 illustrates the block-level schematic of the architecture our proposed reversible floating-point adder design uses.

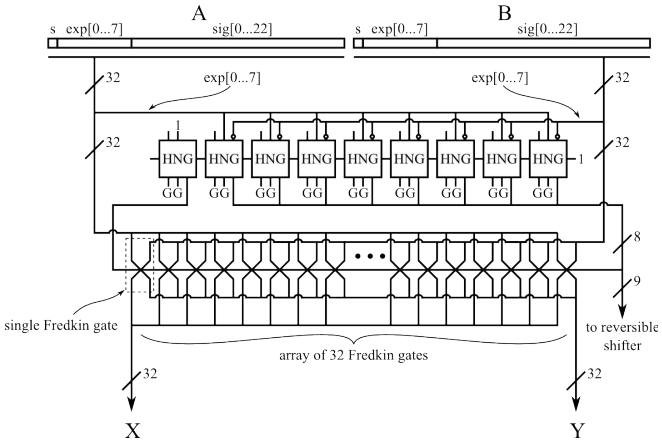

Fig. 3. Our proposed reversible conditional swap unit. The unlabeled inputs of the HNG gates are constant input bits, connected to constant logical zeros. Both entire 32 bit floating-point numbers are input into the array of Fredkin gates, with the eight bit exponent field of each fanned out to the input of the subtracter.

B. Reversible Conditional Swapping

The first step in our reversible floating-pointer adder architecture is to swap the floating-point operands conditionally and reversibly. The rest of the architecture operates assuming that X is the floating-point number with the greater exponent, and Y is the floating-point number that possibly needs to be aligned with X. The conditionally-swapped floating-point numbers, X and Y, are shown in Fig. 2 and Fig. 3.

The exponents of the two floating-point numbers both are unpacked and expanded to nine bits. In order to find their difference the exponent that is the minuend is complemented using two's complement, and with it the difference is calculated. The nine HNG gates implement the reversible subtracter circuit [10]. The sign bit of the difference of the exponents acts as the condition (control) by which the two entire floating-point numbers are swapped: If $\exp_A < \exp_B$, then the floating-point numbers will swap positions, otherwise $\exp_A \geq \exp_B$ and the floating-point numbers are passed through to the next stage without swapping. The array of Fredkin gates serves as a bank of multiplexers that perform the actual swap, and their select line comes from the most significant bit of the difference of the exponents. The exponent difference is passed onto the alignment stage.

C. Reversible Alignment of the Trailing Significands

The magnitude of the difference of the exponents holds the shift amount necessary to align Y's trailing significand (call it sig_Y), but the magnitude must first be extracted from this difference. We designed a reversible conversion unit for this purpose: It reversibly calculates the sign-magnitude representation of a two's complement integer, the sign of which we simply ignore. As Section III.D. details, the design happens also to operate as a conversion unit that reversibly calculates the sign-magnitude representation of a two's complement integer. The reversible conversion unit is illustrated in Fig. 5 and its measurements are given in Table I.

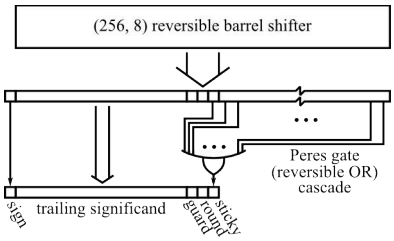

Fig. 4: After shifting, a cascade of 230 Peres gates acting as reversible OR gates computes the sticky bit. The shifted 24 bit trailing significand, guard, and round bits are preserved.

TABLE I
MEASUREMENTS FOR THE REVERSIBLE SIGN-MAGNITUDE-TWO'S COMPLEMENT CONVERSION UNIT

Component	Quantum cost	Garbage outputs	Constant inputs
n-bit Conversion	$5n-4$	n	n

Once the magnitude is extracted from the exponent difference, it will be input as the shift amount to a reversible right-shifter. Researchers already have devoted time and study to the design of reversible shifters, and especially to reversible barrel shifters [11, 12, 13, 14]. We choose the efficient reversible barrel shifter design found in [13] for our alignment unit in order to avoid introducing a state-based control unit that a register-based multi-cycle shifter would require.

The IEEE754 specification requires that only three extra bits be preserved during the right shifts in alignment, called the *guard*, *round*, and *sticky* bits. This barrel shifter reversibly computes the shifted value in a single cycle, but incurs a large cost in the form of the additional logic needed to calculate the value of the sticky bit after the shift. Because our design is asynchronous, it trades the register-and-control design a sequential alignment unit would require for a single OR gate of very large fan-in: The sticky bit is the logical OR of all the shifter output bits between the 27th and the 256th output bits, inclusive. We implement the reversible OR function with high fan-in as a cascade of Peres gates, each operating as a two-input OR gate. Refer to Fig. 4 for details on the reversible sticky bit calculation unit. We omit the details of the previously proposed design of the reversible (256, 8) barrel shifter and present only its quantum cost, garbage outputs, and constant inputs in Table II.

TABLE II
COST MEASUREMENTS FOR THE REVERSIBLE ALIGNMENT UNIT

Component	Quantum cost	Garbage outputs	Constant inputs
(256, 8) reversible barrel shifter [13]	3712	1800	1792
Sticky bit cascade	920	460	230
Total	4632	2260	2022

D. Reversible Addition of the Trailing Significands

The trailing significands of the floating-point operands are stored in sign-magnitude representation, therefore after they have been aligned appropriately they will be converted to two's complement form to be summed. That sum will need to be converted from two's complement back to sign-magnitude for storing the trailing significand of the sum in floating-point format.

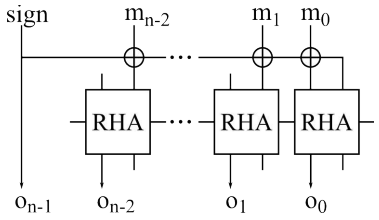

Fig. 5. Our reversible conversion unit. With input in sign-magnitude representation, the output will be the two's complement representation, and vice-versa. By ignoring the sign output, this unit operates as a reversible absolute-value unit for integers in two's complement. The ⊕s and the single fanout of the sign bit are Feynman gates.

As Section III.C. mentions, we have designed a reversible conversion unit that serves simultaneously as a sign-magnitude-to-two's complement unit and a two's complement-to-sign-magnitude unit. Of note is that our unit performs a reversible mapping, f, with the following fixed point:

$$f(10\ldots0)=10\ldots0$$

This implies, for an n bit signed integer:

$$f([-2^{n-1}]_{2's\,complement})=[-0]_{sign-magnitude}$$
$$f([-0]_{sign-magnitude})=[-2^{n-1}]_{2's\,complement}$$

Our design requires two of the 28 bit reversible converters (one for each trailing significand), and a single 29 bit converter for the output of the 28 bit full adder. Before the ensuing reversible normalization step, this 29 bit sign-extended sum is converted back to sign magnitude with a single instance of a 29 bit reversible conversion unit. We provide the quantum cost, garbage outputs, and constant inputs measurements for the reversible conversion units in Table III.

TABLE III
COST MEASUREMENTS FOR THE REVERSIBLE SIGNED CONVERSION UNITS

Component	Quantum cost	Garbage outputs	Constant inputs
28 bit (x2)	136 (272)	28 (56)	28 (56)
29 bit	141	29	29
Total	413	85	85

For the 28 bit full adder unit we use a reversible ripple carry adder design consisting of a single RHA in the least significant sum position followed by 27 reversible full adders (RFAs). For the RHA we use the Peres gate, and for the RFA we use the HNG gate, the gates that implement the required respective functionality with the smallest known quantum cost [9, 10]. This adder design is a straightforward translation from an irreversible design, so we omit a diagram.

TABLE IV
COST MEASUREMENT OF THE RIPPLE CARRY ADDER

Component	Quantum cost	Garbage outputs	Constant inputs
28 Bit Adder	166	55	28

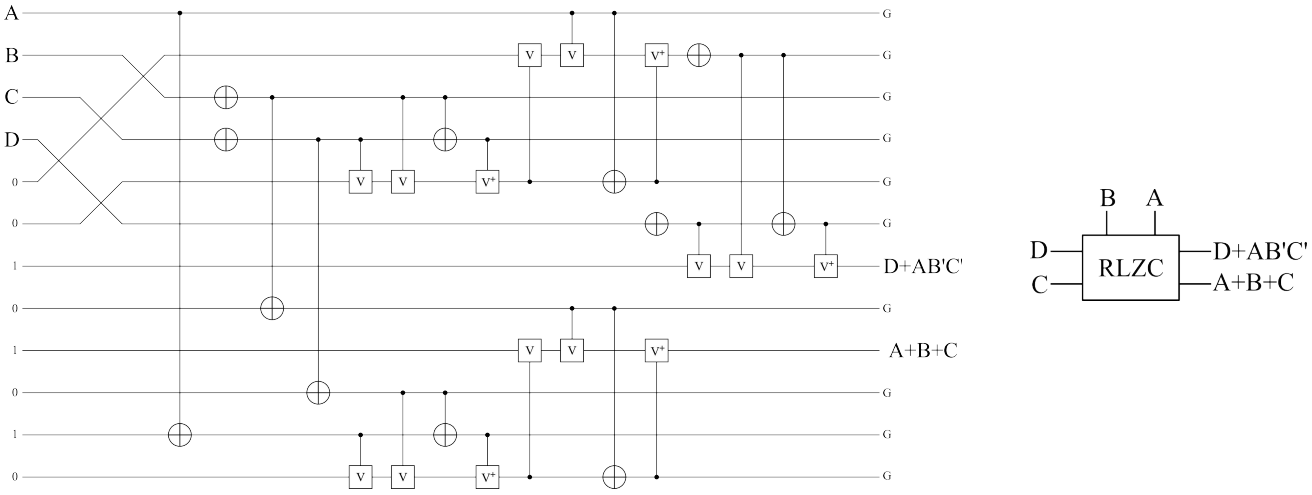

Fig. 6. Our proposed reversible leading zero counter (RLZC) cell. Each of these units has quantum cost 27, 10 garbage outputs, and 8 constant inputs. The left image shows the quantum implementation and the right image shows our block-level diagram of the RLZC cell.

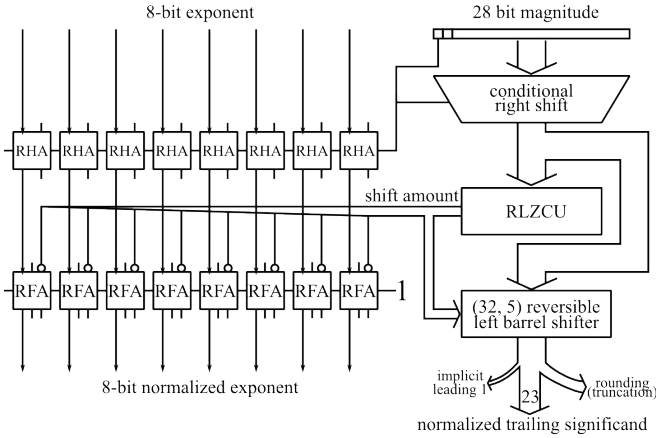

Fig. 7: Our entire reversible post-addition normalization unit. If the magnitude of the sum needs shifting, it will need either shifting one place to the right, in which case the exponent is incremented, or one or more places to the left, in which case the shift amount is subtracted from the exponent.

E. Reversible Normalization

After the sum of the trailing significands has been converted back into sign-magnitude representation, the sign bit is connected directly to the final stage as the sign of the floating-point sum, and the magnitude may need to be normalized. This normalization step may involve either left shifting or right shifting. If a right shift is required, only a right shift of a single position will be required. Otherwise a left shift of possibly several placed may be required. A synchronous floating-point adder architecture might accomplish this behavior using synchronous leading-one detectors and synchronous shift registers, but in keeping with our asynchronous reversible design methodology, we design a completely asynchronous reversible normalization unit.

If normalization requires a right-shift, then a right-shift of a single position will suffice, and the exponent will need to be incremented. The first stage, approximately the top half in Fig. 7, performs this operation. If no right shift is necessary, the exponent is not altered by this stage. If a right shift is necessary, the exponent is incremented, and the next stage, which is conditional left-shifting, will not do anything.

If normalization requires shifting the sum to the left, then first the number of leading zeros must be counted to use as a shift amount. For this we propose a novel reversible design of an asynchronous leading zero counter inspired by an existing irreversible design [20]. The reversible design requires regular repetition and interconnection of a primitive reversible cell design. We give the quantum implementation and block diagram of our proposed reversible leading zero counter (RLZC) cell in Fig. 6. In Fig. 8 we show–using an eight bit example–how multiple copies of the RLZC are interconnected to create the full reversible leading zero counter unit (RLZCU) that asynchronously produces the shift amount.

For an n bit input operand, an ($n=2^k, k$) reversible leading zero counter unit (RLZCU) requires $n-1$ RLZCs and $n-k-1$ Feynman gates for wire fanout [20]. The generalized quantum cost, garbage outputs, and constant inputs are given in Table V.

TABLE V
MEASUREMENTS FOR THE REVERSIBLE LEADING ZERO COUNTER UNIT

Component	Quantum cost	Garbage outputs	Constant inputs
(n, k) RLZCU	$28n-k-28$	$10n-9$	$9n+k-9$

The complete reversible normalization unit works as follows: The eight bit exponent passes through a conditional reversible incrementation unit, then through a reversible subtracter. If the 28 bit magnitude has a value of one in its most significant bit (MSB), it must be right shifted one place. Therefore the MSB of the magnitude connects to both the select line of a bank of reversible 2-to-1 multiplexers and to the carry input of the exponent incrementation unit; if the MSB is a one, the trailing significand is shifted right one place, and the exponent incremented. Otherwise the trailing significand and exponent are passed unaltered to the next phase.

The leading zeros of the trailing significand are counted reversibly and asynchronously using our RLZCU, the output of which connects to both the shift amount input of a (32, 5) reversible barrel left shifter and the minuend input of a

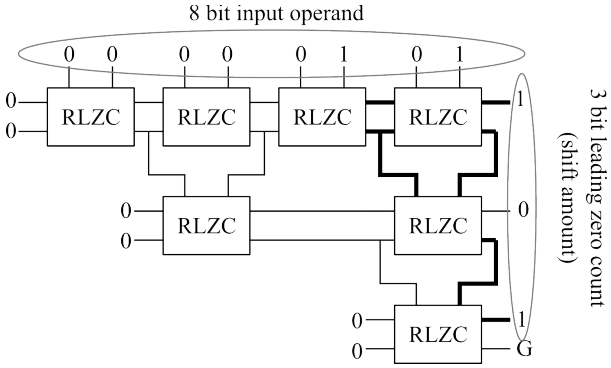

Fig. 8: An example showing how multiple copies of the RLZC cell are regularly arranged to compute the shift amount from an operand word. In this example a three bit shift amount ($101_2 = 5_{10}$) is computed from the eight bit binary input word 00000101, which has five leading zeros. The bold wires carry a logical 1 value.

reversible subtracter. This design ensures the correct shifts and exponent adjustments take place independent of the direction in which shifting is required. We reiterate that if a single right shift is required in the first phase, then the RLZCU will compute a shift amount of zero, and therefore the subtracter will pass the incremented exponent to its output unchanged. This is the required functionality of the normalization unit.

F. Reversible Rounding

Our reversible rounding unit performs the reversible *round toward zero* rounding algorithm specified in the IEEE754 standard [3]. This unit is a pseudo-unit, in that it consists of no extra hardware: It operates simply by transforming some of the input bits into garbage outputs by ignoring them altogether. Fig. 7 illustrates where the reversible truncation occurs after the reversible normalization.

Reversible normalization and rounding conclude the floating-point addition algorithm. The output of the reversible normalization and rounding units in our architecture contain the prepared exponent and trailing significand fields of the floating-point sum. The sign field comes directly from the reversible two's complement-to-sign-magnitude conversion unit prior to normalization (see Fig. 2).

IV. Results and Conclusion

Researchers have worked hard at designing, automating, and synthesizing reversible logic and quantum arithmetic gates [15, 16, 17, 18, 19], but very little has been focused on investigating the design of reversible floating-point units. In this work we present a novel reversible binary32 (single precision) floating-point adder architecture. This architecture we analyze in terms of quantum cost, garbage outputs, and constant inputs. Table VI shows that all three reversible metrics are dominated by the reversible alignment and reversible normalization components. Future related work could optimize one or more of these components. Additionally, more work can be done to expand this architecture to support more features of the IEEE754 standard, including support for subnormal numbers, both signaling and quiet NaNs, overflow & underflow detection, and the remaining rounding modes.

TABLE VI
Presentation of Final Measurements for Proposed Reversible Floating-Point Adder Architecture, Decomposed by Stage

Stage (§)	Quantum cost	Garbage	Constant inputs
Swap (III.B.)	238	19	27
Alignment (III.C.)	4632	2260	2022
Addition (III.D.)	166	55	28
Conversion (III.C.)	413	85	85
Normalization (III.E.)	2009	498	484
Rounding (III.F.)	0	9	0
Total	7458	2926	2646

V. References

[1] R. Landauer, “Irreversibility and heat generation in the computational process,” IBM Journal of Research and Development, 5, pp. 183-191, 1961.

[2] C. H. Bennett, “Logical reversibility of computation,” IBM Journal of Research and Development, pp. 525-532, November 1973.

[3] "IEEE Standard for Floating-Point Arithmetic," IEEE Std 754-2008, vol., no., pp.1-58, Aug. 29 2008.

[4] A. Malik, Seok-Bum Ko, "Effective implementation of floating-point adder using pipelined LOP in FPGAs," Electrical and Computer Engineering, 2005. Canadian Conference on, vol., no., pp.706-709, 1-4 May 2005.

[5] M. Nachtigal, H. Thapliyal, N. Ranganathan, “Design of a reversible single precision floating point multiplier based on operand decomposition,” 2010 10th IEEE Conference on Nanotechnology, pp. 233-237, 2010.

[6] E. Fredkin, T. Toffoli, “Conservative logic,” International Journal of Theoretical Physics, vol. 21, no. 3–4, pp. 219–253, 1982.

[7] A. Peres, “Reversible logic and quantum computers,” Phys. Rev. A, Gen. Phys., vol. 32, no. 6, pp. 3266–3276, Dec. 1985.

[8] H. Thapliyal, A.P. Vinod, "Transistor Realization of Reversible TSG Gate and Reversible Adder Architectures," Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on, vol., no., pp.418-421, 4-7 Dec. 2006.

[9] W.N.N. Hung, X. Song, G. Yang, J. Yang, M. Perkowski, "Optimal synthesis of multiple output Boolean functions using a set of quantum gates by symbolic reachability analysis," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol.25, no.9, pp.1652-1663, Sept. 2006.

[10] M. Haghparast, S. Jassbi, K. Navi, and O. Hashemipour, “Design of a novel reversible multiplier circuit using HNG gate in nanotechnology,” World Applied Sciences Journal, vol. 3, no. 6, pp. 974–978, 2008.

[11] N.M. Nayeem, M.A. Hossain, L. Jamal, H.M.H. Babu, "Efficient Design of Shift Registers Using Reversible Logic," 2009 International Conference on Signal Processing Systems, vol., no., pp.474-478, 15-17 May 2009.

[12] S. Gorgin, A. Kaivani, "Reversible Barrel Shifters," Computer Systems and Applications, 2007. AICCSA '07. IEEE/ACS International Conference on, vol., no., pp.479-483, 13-16 May 2007.

[13] I. Hashmi, H.M.H. Babu, "An Efficient Design of a Reversible Barrel Shifter," VLSI Design, 2010. VLSID '10. 23rd International Conference on, vol., no., pp.93-98, 3-7 Jan. 2010.

[14] S. Kotiyal, H. Thapliyal, N. Ranganathan, "Design of A ternary barrel shifter using multiple-valued reversible logic," Nanotechnology (IEEE-NANO), 2010 10th IEEE Conference on , vol., no., pp.1104-1108, 17-20 Aug. 2010.

[15] M.S. Islam, M.M. Rahman, Z. Begum, M.Z. Hafiz, "Fault tolerant reversible logic synthesis: Carry look-ahead and carry-skip adders," Advances in Computational Tools for Engineering Applications, 2009. ACTEA '09. International Conference on, vol., no., pp.396-401, 15-17 July 2009.

[16] M. Perkowski, M. Kukac, M. Pivtoraiko, P. Kerntopf, M. Folgheraiter, D. Lee, H. Kim, W. Hwangbo, J. Kim, Y. W. Choi, "A hierarchical approach to computer-aided design of quantum circuits," 6th Int'l Symposium on Representations and Methodology of Future Computing Technology, Proc. of, 2003.

[17] M. Haghparast, K. Navi, "A novel reversible BCD adder for nanotechnology based systems," American Journal of Applied Sciences, v.5, issue 3, pp.282-288.

[18] M. Haghparast, M. Mohammadi, K. Navi and M. Eshghi, "Optimized reversible multiplier circuit," Journal of Circuits, Systems, and Computers, vol. 18, no. 2, 1-13, Feb. 2009.

[19] M.Mohamadi, M.Eshghi, K.Navi, "Optimizing the reversible full adder circuit," IEEE EWDTS, Yerevan, Sep.7-10, (2007) 312-315.

[20] J.B. Shackleford, "Asynchronous leading zero counter employing iterative cellular array," US Patent 5 111 415, May 5, 1992.