Quantum Logic Implementation of Unary Arithmetic Operations

Mitchell A. Thornton, David W. Matula*,
Laura Spenner
Department of Computer Science and Engineering
Southern Methodist University
Dallas, Texas, U.S.A.
{mitch,matula,lspenner}@engr.smu.edu

D. Michael Miller**

Department of Computer Science
University of Victoria
Victoria, B.C., Canada
mmiller@cs.uvic.ca

Abstract

The mathematical property of inheritance for certain unary fixed point operations has recently been exploited to enable the efficient formulation of arithmetic algorithms and circuits for operations such as the modular multiplicative inverse, exponentiation, and discrete logarithm computation in classical binary logic circuits. This principle has desirable features with regard to quantum logic circuit implementations and is generalized for the case of MVL arithmetic systems. It is shown that the inheritance principle in conjunction with the bijective nature of many unary functions is used to realize compact quantum logic cascades that require no ancilla digits and generate no garbage outputs.

1. Introduction

Efficient implementations of basic arithmetic algorithms in the form of ALU circuitry are crucial for high-performance computing. Recently, a new class of algorithms suitable for realization as computer arithmetic circuits have been developed that exploit the mathematical property of *inheritance* resulting in logic circuits that are efficient in terms of classical binary logic area, delay minimization, and low power dissipation [1,2]. In the previous work reported in [1,2], the inheritance principle was used to formulate compact lookup-table based architectures in classical binary logic.

Here, we also exploit the inheritance principle, but in a different way that leads to compact quantum and reversible logic circuit cascades with desirable properties. The inheritance principle coupled with the bijective nature present in many unary fixed-point arithmetic operations is also very useful for the design of reversible and quantum logic cascades both for the case of binary and multiple-valued logic systems. This paper provides an overview of the inheritance principle and demonstrates its use in the design of reversible and quantum logic cascades in both binary and MVL logic with the desirable property of requiring no ancilla digits and yielding circuits that have no garbage outputs.

The bijective property is present in many unary functions of interest in arithmetic circuits such as the modular multiplicative inverse, the computation of the discrete logarithm, and the corresponding exponentiation functions.

This paper is organized as follows. Section 2 provides definitions and descriptions of the inheritance principle and provides the guidelines for arithmetic functions suitable for this technique. Section 3 provides examples of the modular multiplicative inverse, the discrete logarithm, and the exponentiation operations implemented in both binary and ternary arithmetic systems. These arithmetic operations are shown to yield bijective functions satisfying the inheritance principle. These implementations are given in the form of quantum logic cascades and the method for the synthesis of these circuits is described. Section 4 includes a discussion of the verification of these circuits. Finally, Section 5 contains conclusions of this work and a discussion of future efforts.

2. Arithmetic Operations and Inheritance

This section contains background information describing the mathematical properties of the arithmetic operations of interest including the definition of inheritance and how it is used to realize efficient quantum and reversible logic cascades for bijective functions. We utilize the modular function notation described in [11] where $|n|_{p^k} = j$ denotes the congruence relation of $n \equiv j \pmod{p^k}$ for $k \geq 1$ where k generally denotes the number of digits used in a fixed-point representation.

2.1 Unary Integer Arithmetic Operations

Unary integer arithmetic functions are of the form $f{:}\mathbb{Z}\rightarrow\mathbb{Z}$ where $\mathbb{Z}$ represents the set of signed integers. We are particularly interested in a subset of $\mathbb{Z}$, denoted as $\mathbb{Z}'$, that is composed of elements that are represented as radix p digit-strings of length k. Such digit strings are used for fixed-point operations in systems with a register size of k. Many such integer arithmetic functions of interest are bijective (i.e. onto and one-to-one). Examples of such functions are the multiplicative inverse function modulo p^k over the integers relatively prime to p, where p is the radix value of the number system of interest and the exponentiation and discrete logarithms of k-digit values with a base value of p.

Since single-precision fixed-point integer arithmetic functions are defined to have a domain and range space containing k-length digit strings, they may be considered to be modular operations over values in $\mathbb{Z}$ with modulus p^k. This statement allows unary arithmetic fixed-point operations to be considered to be modular operations over

* supported in part by the Semiconductor Research Corporation under contract 1399
** supported in part by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada.

the infinite set $\mathbb{Z}$ with the modulus operation restricting the domain and range elements to the finite set $\mathbb{Z}'$.

2.2 Modular Multiplicative Inverse

The relevant properties of the modular multiplicative are described that are later used in the formulation of the binary and MVL quantum cascades.

Definition 1 (Modular Multiplicative Inverse)

The multiplicative inverse modulo-p^k of a fixed-point value, a, is denoted by a^{-1} where both a and a^{-1} are expressed as digit strings of length k and $(a, a^{-1}) \in \mathbb{Z}'$ satisfies:

$$1 = | a \times a^{-1} |_{p^k} \qquad \square$$

As an example, for $p = 2$ and $k = 5$, Table 1 contains all possible values of a and a^{-1} expressed as integer values. Note that the multiplicative inverse only exists for ½ of the possible k-bit fixed-point strings, the odd values.

Table 1: Multiplicative Inverses Modulo-2^5

a (decimal)	a^{-1} (decimal)	a (binary)	a^{-1} (binary)
1	1	00001	00001
3	11	00011	01011
5	13	00101	01101
7	23	00111	10111
9	25	01001	11001
11	3	01011	00011
13	5	01101	00101
15	15	01111	01111
17	17	10001	10001
19	27	10011	11011
21	29	10101	11101
23	7	10111	00111
25	9	11001	01001
27	19	11011	10011
29	21	11101	10101
31	31	11111	11111

Lemma 1

The multiplicative inverse function modulo-p^k over the domain of integers $\mathbb{Z}$ is one-to-one.

Proof

Consider the Abelian group $G = (X,\times)$ where the set X consists of all k-digit strings of the form $x_{k-1}x_{k-2}...x_2x_1 1$ (i.e. $x_0 = 1$) with $x_i \in \{0,1,\ldots, p-1\}$ and the $\times$ operator represents the multiplicative product modulo-p^k. By the group axioms, it is known that there exists some $a^{-1} \in X$ such that $a \times a^{-1} = 1$ where 1 is the group multiplicative identity. Suppose there also exists some element $h' \in X$ satisfying $a \times h' = h' \times a = 1$, then

$$h = h \times 1 = h \times (a \times h') = (h \times a) \times h' = 1 \times h' = h'$$

thus $h = h'$ and a therefore has a unique inverse proving the multiplicative inverse function is one-to-one. $\square$

Theorem 1

The multiplicative inverse function modulo-p^k over the domain of set of integers, $\mathbb{Z}$, is bijective.

Proof

Due to the definition of the group $G = (X,\times)$, every element $a \in X$ has a corresponding inverse $a^{-1} \in X$, thus the multiplicative inverse modulo-p^k function is onto. From Lemma 1, inverses are shown to be unique and thus also one-to-one, therefore the function is bijective. $\square$

It is possible to prove that many other unary fixed-point operations such as exponentiation and the discrete logarithm are also bijective.

Examination of Table 1 shows that the multiplicative inverse function in the binary case has many interesting characteristics, such as the fact that the least significant bit (LSb) is always "1", the next two LSb's of a are equivalent to those of a^{-1}, etc. Such characteristics are also present for moduli for powers of $p > 2$. As an example, consider the k=4 case of multiplicative inverse values expressed in both the natural ternary system with a digit set of $\{0,1,2\}$ and the balanced ternary system with a digit set of $\{\bar{1},0,1\}$ as shown in Table 2.

Table 2: Multiplicative Inverses Modulo-3^4

Decimal a, a^{-1}	Ternary Standard	Ternary Balanced
(1, 1)	(0001,0001)	$(0001, 0001)$
(2, 41)	(0002,1112)	$(001\bar{1}, \bar{1}\bar{1}\bar{1}\bar{1})$
(4, 61)	(0011,2021)	$(0011, \bar{1}1\bar{1}1)$
(8, 71)	(0022,2122)	$(010\bar{1}, 0\bar{1}0\bar{1})$
(16, 76)	(0121,2211)	$(1\bar{1}\bar{1}1, 0\bar{1}11)$
(32, 38)	(1012,1102)	$(11\bar{1}\bar{1}, 111\bar{1})$
(64, 19)	(2101,0201)	$(\bar{1}101, 1\bar{1}01)$
(47, 50)	(1202,1212)	$(11\bar{1}1, 1011)$
(13, 25)	(0111,0221)	$(0111, 10\bar{1}1)$
(26, 53)	(0222,1222)	$(100\bar{1}, \bar{1}00\bar{1})$
(52, 67)	(1221,2111)	$(\bar{1}0\bar{1}1, \bar{1}111)$
(23, 74)	(0212,2202)	$(10\bar{1}\bar{1}, 0\bar{1}1\bar{1})$
(46, 37)	(1201,1101)	$(\bar{1}\bar{1}01, 1101)$
(11, 59)	(0102,2012)	$(011\bar{1}, \bar{1}1\bar{1}\bar{1})$

A compact multiplicative inverse table for $k = 3$ digits for the ternary case with a digit set of $\{0,1,2\}$ is given in Table 2. A fully expanded version of Table 2 would contain $3^4 = 81$ entries, however since this operation is commutative and signed operands yield results with the same sign, these values were omitted. Furthermore, only a single member of a subgroup with a common primitive generator function is included since through use of the generator, the remaining entries can be derived [12].

2.3 Discrete Logarithm and Exponentiation

Another unary operation of interest is provided by the discrete logarithm function (and its inverse, exponentiation). The discrete logarithm function can be used to reduce integer multiplication and division to addition/subtraction operations. A general description of the discrete logarithm is given followed by characteristics that are specific for the two cases of using a radix value of 3 and a radix value of 2.

For radix-p multiple-valued logic, we are interested in discrete logarithms having a k-digit string representation where p is an odd prime base of the digit string. Such a representation can be constructed from the fact [12] that every integer n for $0 \le n \le p^k-1$ has a factorization $n = |g^e p^m|_{p^k}$ where g is a generator for the odd prime p.

Radix Value 3 Discrete Logarithm

The value $g = 2$ is a generator for $p = 3$ (also for $p = 5$, and many other small primes). This means the residues $|2^e|_{3^k}$ cycle through the $2 \bullet 3^{k-1}$ integers that are relatively prime to the radix 3, for $e=0,1,\ldots,2\times3^k-1$. The factorization $n = |2^e 3^m|_{3^k}$ then allows for the pair (e,m) to form a logarithmically based representation of n for every $0 \le n \le 3^k-1$. In fact this pair is unique with $0 \le m \le k$ and $0 \le e \le 2\times3^{m-1}-1$. From these observations, a bijective unary function can be formulated [3] between the standard k-digit ternary strings representing each value n and the k-digit discrete logarithm encoded strings representing these unique (e,m) pairs.

The discrete logarithm is formulated in a ternary system and shown for the case of k=3 in Table 3. The k-digit discrete logarithm encoded string $g_{k-1}g_{k-2}\ldots g_1g_0$ is the concatenation of e and m where m is given by a string of low-order zeros with $0 \le m \le k$, and e is the mixed radix value $e_{k-m-1}e_{k-m-2}\ldots e_2e_1e_0$ where the exponent values are

$$e = e_{k-m-1}(3^{k-m-2}\times 2)+\ldots+e_2(3\times 2)+e_1(2)+e_0$$

with $e_0 \in \{1,2\}$ and $e_i \in \{0,1,2\}$ for $1 \le i \le k-m-1$. The mixed radix form for e given by the above expression is the key to obtaining the inheritance property for the encodings.

As an example, for the case of $k = 3$, shown in Table 3,

$$e = e_2e_1e_0 = e_2(3\cdot 2)+e_1(2)+e_0$$

where $e_1,e_2 \in \{0,1,2\}$ and $e_0 \in \{1,2\}$.

Each k-digit ternary value and it's corresponding (e,m) pair (the concatenation of the (e,m) pair is the discrete logarithm system or DLS value) is unique and all DLS values, (e,m), consist of k digits although the subfields of e and m may each vary in size. The size depends on the number of trailing zeros in the original ternary value since the number of trailing zeros yields the exponent of 3 (i.e. the m value). Additionally, the DLS obeys the inheritance principle as described in the multiplicative inverse example. Table 3 lists all values for the $k = 3$ digit string length case. In Table 3, the values do not appear in natural counting order, rather they appear in blocks in an order depending on the number of trailing zeros as this can make the DLS values easier to interpret. However, unlike the multiplicative inverse, all possible k-digit strings for the integers $\{0,1,2,\ldots,3^k-1\}$ are included in the domain and range of this bijective function.

Radix Value 2 Discrete Logarithm

The case of a DLS for radix-2 has an "exceptional case" formulation. There is no generator for the prime 2, however $g = 3$ is a "semi-generator" for $p = 2$. In this case, the residues $\{|3^e|_{2^k}\}$ cycle through ½ of the odd values. The complementary set of odd values is cycled through by $|2^k - 3^e|_{2^k}$. In order to represent all k-bit strings, a "sign" factor, $(-1)^s$ where $s \in \{0,1\}$ is introduced as another term of the factorization allowing all odd and even values to be represented by the three-term factorization $n = |(-1)^s 3^e 2^m|_{2^k}$. This factorization is unique determining the triple (s,e,m) provided $0 \le m \le k$, $0 \le e \le 3^{k-m-2}$, and $s \in \{0,1\}$. A bijective unary bitstring function can be formulated that maps the k-bit value of n to a k-bit (s,e,m) binary encoded triple employing the novel encoding described in [2]. The operation can thus be implemented as a cascade of quantum logic operators with no ancilla or garbage qubits. As discussed previously, the lengths of the bit fields representing e and m vary depending on the value of n, however, the (s,e,m) value is always a unique k-bit string for any value n. More details on the DLS bitstring encoding can be found in [2].

2.4 The Inheritance Principle

The inheritance property was formally defined in [1] for the binary case, but it is repeated here in terms of any positive integer radix for the sake of completeness.

Table 3: 3-Digit Integers and Corresponding DLS Values

Ternary Integer	e $e_2e_1e_0$ (ternary)	e (decimal)	$\|2^e3^m\|_{3^3}$ (decimal)	m (decimal)
002	001	1	2	0
011	002	2	4	0
022	011	3	8	0
121	012	4	16	0
012	021	5	5	0
101	022	6	10	0
202	101	7	20	0
111	102	8	13	0
222	111	9	26	0
221	112	10	25	0
212	121	11	23	0
201	122	12	19	0
102	201	13	11	0
211	202	14	22	0
122	211	15	17	0
021	212	16	7	0
112	221	17	14	0
001	222	18	1	0
020	010	-	6	1
110	020	-	12	1
220	110	-	24	1
210	120	-	21	1
120	210	-	15	1
010	220	-	3	1
200	100	-	18	2
100	200	-	9	2
000	000	-	0	3

Definition 2 (Inheritance Property)

Let $f(a_{k-1}a_{k-2}\ldots a_1a_0) = b_{k-1}b_{k-2}\ldots b_1b_0$ be a one-to-one mapping of the domain of all k-digit strings to the co-domain of all k-digit strings defined for all k>1. The function f exhibits the property of inheritance if

$f(a_{k-1}a_{k-2}\ldots a_1a_0) = b_{k-1}b_{k-2}\ldots b_1b_0$ implies that $f(a_{n-1}a_{n-2}\ldots a_1a_0) = b_{n-1}b_{n-2}\ldots b_1b_0$ for all $n \le k$. That is, the nth digit b_{n-1} of $f(a_{k-1}a_{k-2}\ldots a_1a_0)$ depends only on a subset of low order n-digit values $\{a_{n-1},a_{n-2},\ldots,a_1,a_0\}$ independent of the values of any the k-n elements of the subset $\{a_{k-1},a_{k-2},\ldots,a_n\}$. □

An example of a fixed-point unary function exhibiting the inheritance property is that of the fixed-point integer square function $f(x) = x^2 \pmod{2^k}$. It is easy to verify that as k increases, the least significant digits of f remain unchanged for a fixed value of x.

3. Arithmetic Circuit Architecture

In this section, binary and MVL quantum and reversible circuits are described that realize fixed-point unary arithmetic functions. In particular, we give specific examples for the multiplicative inverse function although any unary arithmetic function exhibiting the inheritance property could be used without loss of generality.

3.1 Binary Modular Multiplicative Inverse Circuit

A cascade of Controlled-NOT (CNOT) and Toffoli logic gates that implements the 5-bit fixed-point multiplicative inverse function modulo-32 is shown in Figure 1. The circuit inputs and outputs are the 5-bit values of a and a^{-1} expressed as $a = (x_4x_3x_2x_1x_0)_2$ and $a^{-1} = (y_4y_3y_2y_1y_0)_2$. The inheritance principle can be observed in the structure of the cascade by noting that more significant bits are formed as functions of lesser significant bits.

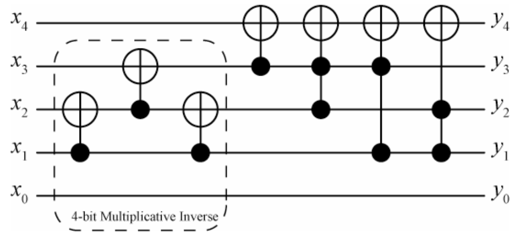

Figure 1: CNOT and Toffoli Gate Cascade Realizing the Multiplicative Inverse Modulo 2^5

3.2 Binary Discrete Logarithm Circuit

A cascade of Controlled-NOT (CNOT) and generalized Toffoli logic gates that implements the 5-bit function that maps an integer value to an (s,e,m) triple modulo-32 is shown in Figure 2.

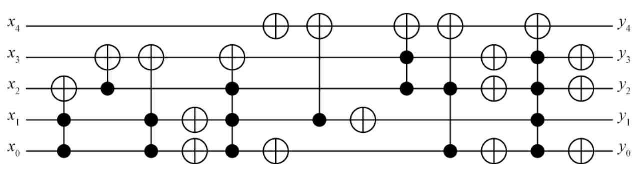

Figure 2: CNOT and Toffoli Gate Cascade Realizing the Discrete Logarithm Modulo 2^5 Function

3.3 Ternary Quantum Logic Gates

Many different ternary single qudit gates have been considered in the past [5,6,7,8,9,10]. Each of the gates in Table 4 can be defined by 1 of 6 possible permutations of the basis vectors $|0\rangle, |1\rangle,$ and $|2\rangle$. Each transfer matrix, $\mathbf{U}_{ijk}$, is denoted in Table 2 by the (i,j,k) permutation triple denoting the 3×3 transfer matrix computed as $\mathbf{U}_{ijk} = |0\rangle\langle i| + |1\rangle\langle j| + |2\rangle\langle k|$.

Because the six operations in Table 2 can be considered to be elements of the symmetry group S_3, group theory arguments can be used to prove functional completeness (universality) when controlled versions are available as is done similarly in [7]. Universal gate sets include {N, C1}, {N, C2}, {N, NC1}, and {N, NC2}. This is easily seen to be the case since C1×C1 = C2, C2×C2 = C1, N×C1 = NC1, and N×C2 = NC2. We note that other gates and universal sets are possible such as the ternary swap gate described in [7].

Table 4: Six Ternary Quantum Permutation Gates

Name	Permutation	Symbol	Reference
Identity	(0,1,2)	I	-
Modulo-add by 1	(1,2,0)	C1	[4,5,7]
Modulo-add by 2	(2,0,1)	C2	[4,5]
Negation	(2,1,0)	N	[5,7]
Neg. Mod. 1	(0,2,1)	NC1	[5,6]
Neg. Mod. 2	(1,0,2)	NC2	[5,6]

3.4 Ternary Modular Multiplicative Inverse Circuit

The mathematical principles for unary functions obeying the inheritance principle are independent of the number system used. As an example, the multiplicative inverse circuit is implemented in the ternary ($p = 3$) system with a digit set of {0,1,2}. For radix values of p, the multiplicative inverse only exists for $(p-1)/p$ of all possible values and hence only 2/3 of the ternary integer values.

A corresponding diagram of the circuit realizing the function in Table 3 is shown in Figure 3. Note that we use the notation for controlled qudit gates as that used in [5] where the controlled qudit value is indicated numerically at the control point. The circuit in Figure 3 was generated with a version of the automatic synthesis method from [6] adapted to QMDDs [13].

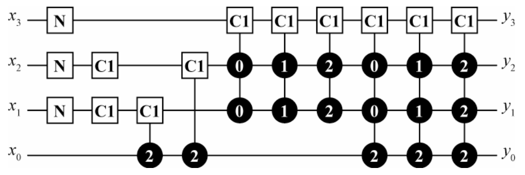

Figure 3: Multiplicative Inverse Modulo 3^4 Circuit Cascade in Ternary Logic

When a balanced ternary system is used (i.e. the digit set is {-1,0,1}), a modular digit encoding of 0→0, 1→(+1), 2→(-1) results in the multiplicative inverse circuit shown in Figure 4.

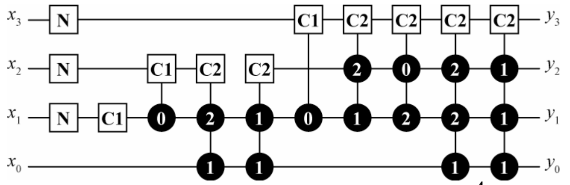

Figure 4: Balanced Ternary Inverse Modulo 3^4 Circuit Cascade in Logic with Modular Digit Value Encoding

The quantum ternary circuit in Figure 5 results when a digit encoding of $0\rightarrow(-1)$, $1\rightarrow0$, $2\rightarrow(+1)$ is used.

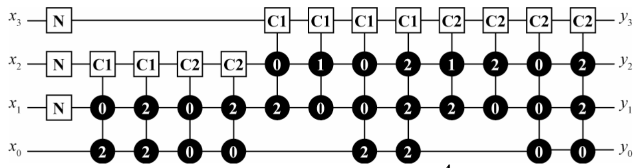

Figure 5: Multiplicative Inverse Modulo 3^4 Circuit Cascade in Balanced Ternary Logic

The modular encoding used to develop the circuit cascade shown in Figure 4 requires fewer gates than the encoding used to develop the circuit cascade shown in Figure 5. This indicates that a modular encoding is a better encoding to be used when developing the multiplicative inverse circuit cascades using primitive gates whose functions are also modular.

3.5 Ternary Discrete Logarithm Circuit Cascades

A ternary quantum cascade circuit was designed both manually and by the QMDD-based adaptation of the automated synthesis algorithm described in [6]. Both of these circuits were verified through equivalence checking using QMDDs [13]. The manually designed circuit is shown in Figure 6 and the automatically synthesized circuit is shown in Figure 7.

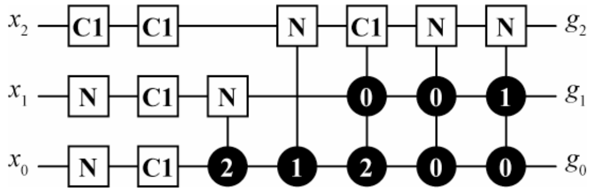

Figure 6: DLS Conversion Circuit Designed Manually

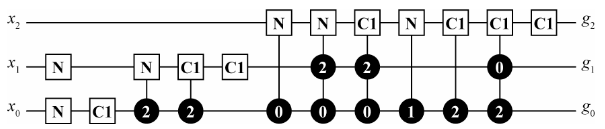

Figure 7: Automatically Synthesized DLS Circuit

In the circuit shown in Figure 6, the gate set was restricted to {**N**,**C1**} which is functionally complete over the set of ternary quantum operators. It is noted that the use of **C2** or **NC1** gates would allow for a lesser gate count. For example, the two **C1** gates in series at the x_2 input can be replaced by a single **C2** gate.

3.6 Mathematical and Physical Reversibility

As is well-known, all quantum circuits must necessarily implement functions that are bijective. In the case of binary functions, the resulting circuits are physically reversible since the unitary transfer matrices are symmetric. While the ternary quantum circuits also represent bijective functions, they are not physically reversible. The DLS circuits shown in Figures 7 and 8 are useful for demonstrating the differences between physical and mathematical reversibility.

The inverse function of the DLS operation is the exponentiation function. In Figure 2, a binary DLS circuit is shown. Since binary quantum circuits exhibit both physical and mathematical reversibility, the binary exponentiation circuit is the same circuit as that shown in Figure 2 when evaluated from right to left. However in the case of the ternary quantum circuits, the exponentiation circuit must be formulated through evaluating the DLS circuit from right to left and with one-to-one replacement of the individual quantum gates by their inverses.

In order to determine the gate replacements, it is necessary to find the replacement gate $\mathbf{U}_2$ for each $\mathbf{U}_1$ that satisfies:

$$\mathbf{U}_2\mathbf{U}_1 = \mathbf{I}$$

In the case of the binary quantum circuits used in this paper, all individual gates are self-inverses, hence the circuits are all physically reversible. In the work here, we are considering the 6 quantum ternary permutation gates represented by the set {I, C1, C2, N, NC1, NC2}. Of these 6 gates all are self-inverses except for C1 and C2, thus a physically reversible circuit can be realized by evaluating the circuit from right to left and interchanging all instances of C1 and C2 gates. Using this observation, the circuit that computes the exponential is derived from the DLS circuit shown in Figure 6 as that shown in Figure 8.

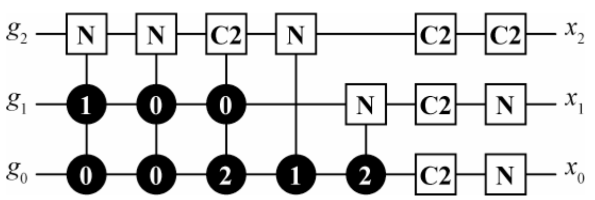

Figure 8: Exponential Circuit Derived from DLS Circuit in Figure 6

3.7 General Structure of Arithmetic Circuit Cascades

In [1,2], the inheritance principle is exploited to realize fixed-point unary arithmetic functions in classical binary logic using compact lookup-table (LUT) based architectures. In the results presented here we also exploit the inheritance principle, however the enhancements are present in the fact that all resulting circuits require no input ancilla bits nor are any garbage bits produced. It is also noted that extending the precision of such circuits (i.e. using extending the operand value set) requires only the design of the circuitry for the more significant bits as the portion of the circuit that generates the lesser significant bits is identical to the smaller precision circuits. This characteristic is shown in Figure 1 where the dashed line box encloses the subcircuit that generates the multiplicative inverse modulo-16 for the 4-bit function.

These observations can be generalized to describe the architecture of any cascade realizing a fixed-point unary function to be of the form of that shown in Figure 9 where each gate represents controlled $\mathbf{U}_i$ digit transfer matrices.

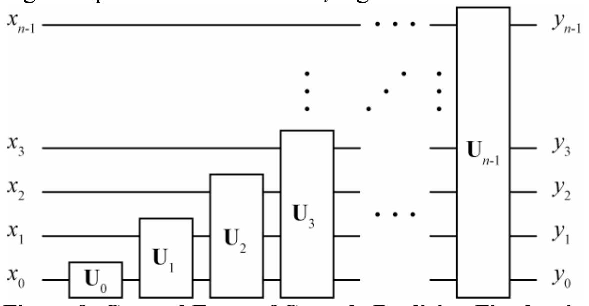

Figure 9: General Form of Cascade Realizing Fixed-point Unary Arithmetic Functions with Inheritance

It is noted that each transfer matrix $\mathbf{U}_i$ is not necessarily of maximum dimension as shown in Figure 9 and is dependent upon the arithmetic function under consideration. As an example, for the multiplicative inverse circuit shown in Figure 1, $\mathbf{U}_0$, $\mathbf{U}_1$, and $\mathbf{U}_2$ are all identity matrices since $y_i = x_i \forall i<3$. $\mathbf{U}_3$ is represented by the cascade of the leftmost three CNOT gates in the cascade (shown enclosed in the dotted lines). The general form of the transfer matrix representing $\mathbf{U}_3$ for the example in Figure 1 is:

$$\mathbf{U}_3 = \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix}$$

4. Circuit Synthesis and Validation

Some of the circuits presented in this paper are synthesized manually and others are produced using a QMDD-based modified version of the technique described in [6]. Through manual design of these circuits new heuristics were identified and augmented to the synthesis techniques described in [6].

The quantum circuits presented here are all verified through the use of the QMDD data structure [13]. In some cases they are exhaustively simulated and compared to the specified function truth table using techniques similar to those described in [4]. In other cases, when an existing circuit is already designed and verified through exhaustive simulation, the canonical nature of QMDD (for a given variable ordering) is used to perform functional equivalence checking.

5. Conclusions and Future Directions

This work illustrates how bijective unary fixed-point arithmetic functions can be realized in relatively compact binary and quantum logic cascades. The principle of inheritance is exploited to ease the burden of quantum cascade realization. Another benefit of this work is that several new ternary cascades are presented and verified that may be added to the relatively small collections of multiple-valued quantum logic circuit examples currently in existence today.

In the future we plan to use the results described here to further refine the updated automated synthesis technique originally described in [6] and to disseminate the resulting algorithms to the research community. Additionally, we plan to investigate how the multiplicative inverse and discrete logarithm operations extend naturally to radix 5. This will provide needed examples for investigation of cascades utilizing quintary quantum logic gates.

References

[1] D. W. Matula, A, Fit-Florea, and M. A. Thornton, Lookup Table Structures for Multiplicative Inverses Modulo 2^k, Proceedings of the *IEEE Symposium on Computer Arithmetic*, June 27-29, 2005, pp. 130-135.

[2] L. Li, A. Fit-Florea, M. A. Thornton, and D. W. Matula, Performance Evaluation of a Novel Table Lookup Method and Architecture for Integer Functions, Proceedings of the *IEEE International Conference on Application-specific Systems, Architectures, and Processors*, September 11-13, 2006, pp. 99-104.

[3] D. W. Matula, Discrete Log Number Systems, *SMU Internal Report*, 2007.

[4] D. Goodman, M. A. Thornton, D. Y. Feinstein, and D. M. Miller, Quantum Logic Circuit Simulation Based on the QMDD Data Structure, Proceedings of the *Workshop on Applications of the Reed-Muller Expansion in Circuit Design and Representations and Methodology of Future Computing Technology*, May 16, 2007, pp. 99-105.

[5] D. Gottesman, Fault-Tolerant Quantum Computation with Higher-Dimensional Systems, 1998, arXiv:quant-ph/9802007.

[6] D. M. Miller, D. Maslov, and G. Dueck, Synthesis of Quantum Multiple-Valued Circuits, *Journal of Multiple-Valued Logic and Soft Computing*, vol. 12, no. 5-6, 2006, pp. 431-450.

[7] G. Yang, X. Song, and M. Perkowski, Realizing Ternary Quantum Switching Networks without Ancilla Bits, *J. Phys. A: Math. Gen*., vol. 38, no. 44, November 2005, arXiv:quant-ph/0509192v1.

[8] A. DeVos, B. Raa, and L. Storme, Generating the Group of Reversible Logic Gates, *J. Phys. A.: Math. Gen*., vol. 35, no. 33, August 2002, pp. 7063-7078.

[9] J. Daboul, X. Wang, and B. C. Sanders, Quantum Gates on Hybrid Qudits, *J. Phys. A.: Math. Gen*., vol. 36, no. 14, 2003, pp. 2525-2536, arXiv:quant-ph/0211185v1.

[10] A. Muthukrishnan and C. R. Stroud Jr., Multi-Valued Logic Gates for Quantum Computation, *Phys. Rev. A* preprint, June 2000, arXiv:quant-ph/0002033v2.

[11] S. Szabó and R. I. Tanaka, **Residue Arithmetic and Its Application to Computer Technology**, McGraw-Hill Book Co., 1967.

[12] G. H. Hardy and E. M. Wright, **An Introduction to the Theory of Numbers**, Oxford Science Publications, 5th ed., 1979.

[13] D. M. Miller and M. A. Thornton, QMDD: A Decision Diagram Structure for Reversible and Quantum Circuits, Proceedings of the *IEEE Int. Symp. on Multiple-Valued Logic*, May 2006, CD, 6 pp.