Quantum Information and Computation, Vol. 24, No. 15&16 (2024) 1283–1312

SYNTHESIS OF BINARY-INPUT MULTI-VALUED OUTPUT OPTICAL CASCADES FOR REVERSIBLE AND QUANTUM TECHNOLOGIES

ISHANI AGARWAL
Department of Electrical and Computer Engineering, Portland State University
Portland, Oregon 97201, United States

MIROSLAV SARAIVANOV
Department of Electrical and Computer Engineering, Portland State University
Portland, Oregon 97201, United States

MAREK PERKOWSKI
Department of Electrical and Computer Engineering, Portland State University
Portland, Oregon 97201, United States

Received January 24, 2024
Revised December 12, 2024

This paper extends the decomposition from the group theory based methods of Sasao and Saraivanov to design binary input multivalued output quantum cascades realized with optical NOT, SWAP, and Fredkin Gates. We present this method for 3, 5, and 7-valued outputs, but in general it can be used for odd prime-valued outputs. The method can be extended to realize hybrid functions with different valued outputs. A class of local transformations is presented that can simplify the final cascade circuits. Using these simplifying transformations, we present an upper bound on the maximum number of gates in an arbitrary n-variable input and k-valued output function.

Keywords: optical computing, multi-valued logic, group theory, group decomposition, reversible logic, quantum layout

1 Introduction

The implementation of reversible logic in the computational industry is considered to be one of the most promising solutions to reduce power consumption per logical operation, and one of the major applications of reversible logic is in quantum computing [1, 2, 3]. Continuing to increase the capacity to transmit and process both quantum and classical information is necessary in the computational industry. A key solution is derived from Landauer's principle—which was experimentally proved in 2012 [4]—which stipulates a minimum energy dissipation for each irreversible operation [5, 6]. Achieving Landauer's limit necessitates the use of reversible computation, where inputs can be reconstructed from their outputs. As shown in [7], Landauer's assumption can be broken through optical interference, acting as a reversible element. Hence, reversible computing offers the potential to achieve zero dissipation by preventing entropy loss during computation. Moreover, with computational demand for reduced power per bit operation, we have to fundamentally revise computer design principles, transitioning from electron-based to photon-based systems, as it is originally discussed in [1]

with generative artificial intelligence models and cryptocurrency mining and exchanging [8].

Optical systems, particularly integrated photonics, offer distinct advantages over traditional electronics. Photons, being massless and traveling at the speed of light, enable faster data transmission with minimal latency, essential for scaling both quantum and classical systems. Additionally, optical components support high parallelism, improving the efficiency of information processing. Photons generate less heat than electrons, reducing power consumption and cooling requirements, making optical systems energy efficient [9].

Furthermore, multivalued logic is said to be the best solution in future all-optical signal processing systems as it can increase the data-carrying capacities, large information storage, and high-speed arithmetical operations [10]. With recent emerging technologies such as Rigetti Aspen, multivalued computers are becoming practical as seen in [11]. It has already been shown that multivalued logic can efficiently be applied to various optical computation solutions by using the polarization states of light along with its presence or absence [12, 13, 14, 15]. Our methodology is designed for optical quantum and non-quantum technologies, not non-optical technologies.

Potentially, there exist four types of optical reversible or optical reversible quantum circuits: (1) Functions with binary input and binary output (2) Functions with binary input and multivalued output (3) Functions with multivalued input and binary output (4) Functions with multivalued input and multivalued output. Several authors have addressed the problem of minimizing quantum circuits with binary inputs and binary outputs, such as [16, 17, 18, 19, 20] to list just a few. There also exist a few methods to synthesize functions with multivalued input and binary or multivalued output [19, 21, 22, 23, 24]. These methodologies were for non-optical technologies.

However, circuits with binary input and multivalued output are the least commonly studied but are the focus of this paper. There is currently no research on the circuit realization of binary input and multivalued output quantum circuits, particularly for optical technologies. We demonstrate our group theory-based methodology using quantum circuits with binary inputs and 3, 5, or 7 output values, but our method can be easily extended to other radices of functions. Group theory was previously used to design cascade circuits in [25, 26, 27, 28].

The only gates that we require in the proposed method are binary inverters, single binary-controlled multivalued-targets Fredkin gates (one control of binary logic and two targets of multivalued logic), and multivalued SWAP gates. Current optical computing systems have successfully implemented both binary and multivalued reversible logic and are able to construct circuits using combinations of CNOT and Fredkin gates [1, 29, 30, 31, 32, 33, 34, 35]. In particular, Figure 6 from [1] shows the basic realization of a Fredkin gate using a Mach-Zehnder interferometer.

In several other synthesis methods for multivalued quantum circuits as well as in binary quantum circuits, the number of SWAP gates grows rapidly as presented with examples in [36]. Because in some modern quantum layouts a qubit, A, has four neighbors, then selecting this qubit as a target and giving variables x_1, x_2, x_3, x_4 as its neighbors, we do not need SWAP gates to control multivalued qubit A with these four neighbors. Controlling other variables with these four neighbors may be more difficult and require SWAP gates, but still, the total number of SWAP gates is smaller for cascades than for other quantum circuit structures mapped to quantum layouts. While we give here our motivation for multivalued cascades,

the problem of layout is not further discussed in this paper. The quantum layout problem is discussed in [36, 37, 38, 39] and the physical implementation of multivalued quantum gates is further discussed in [10, 40, 41]. More specifically, multivalued Fredkin gates can be efficiently realized in several quantum technologies, including optical systems, but this paper is devoted only to optical systems [1, 29, 35, 42, 43, 44].

The fundamental realization of optical reversible circuits is the Fredkin gate [1]. The multivalued Fredkin Gate is a key stand-alone gate in optical computing and has a fidelity of 99.75% [1, 30, 45, 46, 47, 48, 49]. It can be used to design 16 Boolean logical operations and multivalued circuits. As all of the outputs of the 16 circuits have logical states of the 16 Boolean operations, there is no garbage in their design [10].

As compared to the M-S gate, which is a purely theoretical gate, in this paper we introduce a method to create quantum circuits only using gates that are experimentally realizable on optical hardware. Yet, note that the ternary Fredkin gate and SWAP gate can also be realized using Muthukrishnan-Stroud gates (M-S gates) [50, 51, 52, 53]. Decomposing the SWAP gates into a combination of Fredkin gates and inverters and then decomposing these Fredkin gates into M-S gates, our methodology can be extended to produce circuits that only require M-S gates and binary inverters. However, Fredkin gates are much simpler than M-S gates for higher-level valued logic. Our paper is focused on reversible optical logic and quantum optical logic, with a focus on multivalued logic, so we do not expand on the extension of decomposing the Fredkin gates and SWAP gates into M-S gates in the remainder of the paper.

To the best of our knowledge, no optical technology exists that demonstrates the implementation of a multivalued Toffoli-like gate. Hence, methodologies that synthesize quantum circuits using multivalued Toffoli-like gates are not practical in optical quantum technologies. Our approach is unique because it uses Fredkin as a base in contrast to most other researchers that use Toffoli as a base. An additional advantage of our methodology is that Fredkin and SWAP gates are of rather similar complexity for different radices, so our method remains without modification for higher radices. In contrast, for any radix greater than three, Toffoli gates get increasingly more complicated and are different from one radix to another and must be individually designed by the user. Additionally, ternary Fredkin and quaternary Fredkin are quite different in non-optical technologies, while they are very similar in optical technologies [1].

Again, this methodology applies to multivalued optical technologies but not to ternary non-optical technologies because it is quite expensive. The ternary realization of the gates used in this method is cheap in optical technologies but more expensive in other technologies. Our method is applicable to both multivalued reversible non-quantum logic and multivalued reversible quantum logic. Although our method relates directly to reversible non-quantum logic, many quantum algorithms—including Grover's algorithm and quantum walk—have oracles realized with reversible quantum logic. These oracles do not use non-permutative gates. However, they are part of complete quantum algorithms that include truly quantum gates such as the Hadamard and phase gates.

Please observe that in this paper, we employ the notation of quantum gates used in both quantum superconducting and optical domains. There are other notations used specifically for optical domains as in [54, 55].

This paper is organized as follows. In section 2, we introduce group theory and generate

groups that will be used in our approach. In section 3, we introduce group function decomposition as well as examples for one, two, and three-variable input circuits. Section 4 lists the local transformations that are used in our methodology. Section 5 presents additional examples of our methodology. Section 6 proves an upper bound on the maximum number of cells and gates in the canonical cascade and quantum circuit respectively in an arbitrary binary input and multivalued output function using our method.

2 Group Theory

A binary operation is an operation performed on two elements of a set to obtain a third element in the set. A group, denoted $\langle G, \star \rangle$, is an algebraic structure consisting of a set G with a binary operation, $\star$, that satisfies the four properties of closure, identity, associativity, and invertibility.

(i) Closure: $a \star b \in G$ for all $a, b \in G$.
(ii) Identity: There exists a unique $e \in G$ such that $e \star a = a \star e = a$ for all $a \in G$.
(iii) Associativity: $a \star (b \star c) = (a \star b) \star c$ for all $a, b, c \in G$.
(iv) Invertibility: For all $a \in G$, there exists $b \in G$ such that $a \star b = b \star a = e$ where e is the identity element.

The order of a group is equal to the number of unique elements in its set. An Abelian group, $\langle G, \star \rangle$, is a commutative group that satisfies $g_1 \star g_2 = g_2 \star g_1$ for all $g_1, g_2 \in G$. A cyclic group is a group that is generated by one of its elements, g. For example, the group $\langle g^0, g^1, g^2, g^3 \rangle$ where $g^4 = g^0$ is a cyclic group of order 4. The dihedral group, denoted D_n, is a non-Abelian group of order $2n$ consisting of the rotational and reflectional symmetries of a regular n-gon. In general, $D_n = \langle a^0, a^1 \dots a^{n-1}, g, ag, a^2 g \dots a^{n-1} g \rangle$ where $a^n = a^0$ and $g^2 = g^0$. $D_n = C_n \times C_2$ where C_k is the cyclic group of order k. In order to use group functions to realize logical functions, we must avoid groups in which two different input combinations can produce the same output value. This means that we need to create groups that are non-Abelian.

Consider the groups of order 2 through 14. In order for our method to be applicable, we require a non-Abelian group with two variables and an odd valued output as shown in Table 1. We present a proof for the requirement of an odd valued output in section 3.4.

A proof for the existence of non-Abelian groups in Table 1 can be found in [9, 14]. Hence, based on the table, the possible groups that can be used in our method are the groups of order 6, 10, and 14, or D_3, D_5, and D_7 respectively, and we will use these groups to demonstrate our methodology.

Table 1. Applicability of groups of order two through fourteen.

Order	Non-Abelian group?	Odd Valued Output?	Our Method Applicable?
2	No	—	No
3	No	—	No
4	No	—	No
5	No	—	No
6	Yes	Yes	Yes
7	No	—	No
8	Yes	No	No
9	No	—	No
10	Yes	Yes	Yes
11	No	—	No
12	Yes	No	No
13	No	—	No
14	Yes	Yes	Yes

2.1 Dihedral Group Generation

2.1.1 Group D_3

The group $D_3 = \langle a, g : a^3 = g^2 = e, gag = a^{-1} \rangle$. The group maps for elements a, g are shown in Figure 1. The group maps illustrate the behavior of the elements after composing them with either a or g. For example, when g is composed with g, the resulting element is I.

Fig. 1. Group Maps for elements a, g.

We construct the group D_3 using elements a and g with I as the identity element as shown in Figure 2. Elements I, g and a each have three horizontal rails where each rail represents a qubit. Element g swaps the bottom two rails, and element a maps the nth rail with the $n + 1$th rail where the bottom rail maps to the first rail.

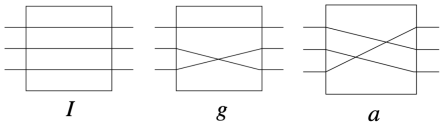

Fig. 2. D_3 elements I, g, a.

We can generate the other elements of this group by composing a and g. For example, a^2 can be generated as shown in Figure 3:

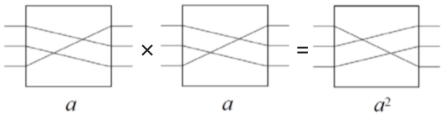

Fig. 3. Element a^2.

All the different elements of this group are shown in Figure 4.

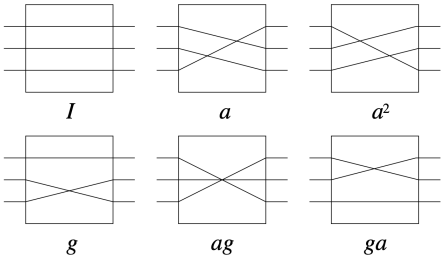

Fig. 4. Elements of D_3.

Since $gag = a^{-1}$, $a^3 = I$, and $g^2 = I$,

$$(ag)g = a(gg) = aI = a.$$

$$gag = a^{-1} = a^2.$$

$$a^2g = (g^{-1}g)a^2g = g^{-1}(ga^2g) = g^{-1}a^{-2} = ga.$$

The complete group map of D_3 is shown in Figure 5.

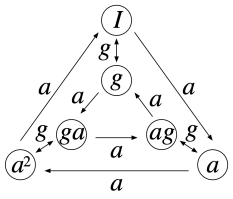

Fig. 5. Group Map of D_3.

Note that D_3 is non-Abelian as $ag \neq ga$.

2.1.2 Group D_5

The group D_5 can be generated in a similar manner as D_3. $D_5 = \langle a, g : a^5 = g^2 = I, gag = a^{-1} \rangle$. Elements a and g are shown in Figure 6 where a permutes the nth rail to the $n+1$th rail and g swaps the nth rail with the $p-n$th rail when $n \neq 0$.

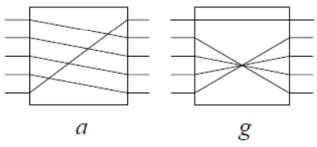

Fig. 6. D_5 elements a, g.

Elements a and g can be repeatedly composed to generate the complete dihedral group of order 10. The group map of D_5 is shown in Figure 7.

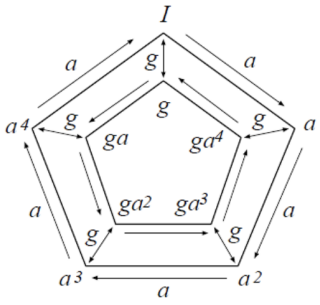

Fig. 7. D_5 Group Map.

Once again $gag = a^{-1}$ and $ga^{-1}g = a$.

2.1.3 Dihedral Group of Order N

In general, the group D_n for arbitrary prime number n can be generated using the elements a and g such that a permutes the nth rail to the $n+1$th rail where the last rail maps to the top rail, and g swaps the ith rail with the $p-i$th rail for all $i \neq 0$. The group map of D_n is illustrated in Figure 8.

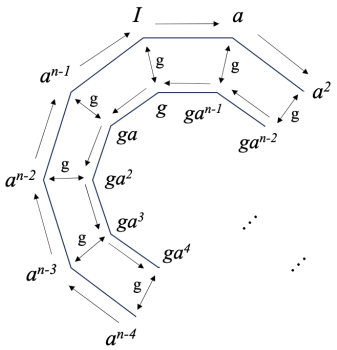

Fig. 8. D_n Group Map.

3 Group Function Decomposition

3.1 Single Input Variable

Let G be a group and $B = \{0, 1\}$. Then, $F : B^n \rightarrow G$ is a group function. By Shannon's Expansion [27], a group function $F(x) : B^n \rightarrow D_3$ decomposes as follows:

$$F(\hat{X}, x_n) = F_a(\hat{X}) g^{x_n} F_b(\hat{X}) g^{x_n}. \tag{1}$$

Here x_n is a two-valued input variable and $F_a(\hat{X})$ and $F_b(\hat{X})$ denote group functions that do not depend on x_n with $\hat{X} = (x_1, x_2, \ldots x_{m-1}) \in X^{m-1}$. A proof can be found in [9]. This decomposition is similar to Shannon's expansion for a classical binary logic function and is essential in the design of canonical cascades. Figure 9 shows the canonical cascade function of Eq. (1):

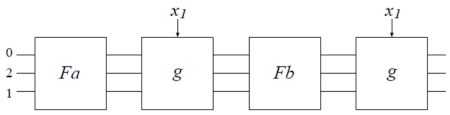

Fig. 9. One Variable Canonical Cascade.

The decomposition indicates that element g needs to be modified so that it can be controlled with a binary input variable. When $x = 1$, g^x swaps the bottom two lines and when $x = 0$, g^x performs the identity permutation. Figure 10 shows a representation of g^x. Note that the g^x element resembles a reversible Controlled SWAP or Fredkin gate.

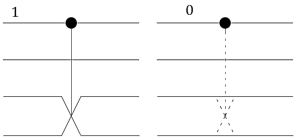

Fig. 10. Element g modified to g^{x_n} so it can be controlled.

Since $F_a(\hat{X})$ and $F_b(\hat{X})$ denote arbitrary group functions that do not depend on x_n with $\hat{X} = (x_1, x_2, \ldots x_{m-1}) \in X^{m-1}$, we can define them to be functions of the group element a. More specifically, assume $F_a(\hat{X}) = a^{f_a(\hat{X})}$ and $F_b(\hat{X}) = a^{f_b(\hat{X})}$ where a is the shift element from our group.

From Eq. (1) we get

$$F(\hat{X}, x_n) = F_a(\hat{X}) g^{x_n} F_b(\hat{X}) g^{x_n} = a^{f_a(\hat{X})} g^{x_n} a^{f_b(\hat{X})} g^{x_n}.$$

Equivalently,

$$a^{f(x_n)} = a^{f_a(\hat{X})} g^{x_n} a^{f_b(\hat{X})} g^{x_n}. \tag{2}$$

Therefore,

$$F(\hat{X}, 0) = a^{f_a(\hat{X})} g^0 a^{f_b(\hat{X})} g^0.$$

$$F(\hat{X},1) = a^{f_a(\hat{X})} g^1 a^{f_b(\hat{X})} g^1.$$

Since $g^0 = I$ and $gag = a^{-1}$, we get

$$F(\hat{X},0) = a^{f_a(\hat{X})+f_b(\hat{X})}.$$

$$F(\hat{X},1) = a^{f_a(\hat{X})-f_b(\hat{X})}.$$

For convenience, define $F(\hat{X},k) = a^{f(\hat{X},k)}$ to express the above equations in terms of their exponents only:

$$f(\hat{X},0) = f_a(\hat{X}) + f_b(\hat{X}).$$
$$f(\hat{X},1) = f_a(\hat{X}) - f_b(\hat{X}).$$

Converting the above expressions into matrix form we get,

$$\begin{bmatrix} f(\hat{X},0) \\ f(\hat{X},1) \end{bmatrix} = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix} \begin{bmatrix} f_a(\hat{X}) \\ f_b(\hat{X}) \end{bmatrix}$$

$$\Rightarrow \vec{F} = W_1 \vec{w}. \tag{3}$$

where $\vec{F}$ is the truth vector of the function and W_1 is the first Walsh Matrix. Walsh transform as well as matrix representation of this transform are presented in [8, 13, 21, 22, 57]. Note that $f_a(\hat{X})$ and $f_b(\hat{X})$ represent the exponents of a and literally describe the canonical form of the circuit cascade. If we let $\vec{w} = \begin{bmatrix} f_a(\hat{X}) \\ f_b(\hat{X}) \end{bmatrix} = \begin{bmatrix} w_a \\ w_b \end{bmatrix}$, then w_a and w_b are the exponents of a in our cascade. Hence, the canonical cascade can be found by solving Eq. (3) for $\vec{w}$ where $\vec{w}$ is the Walsh Spectrum of $\vec{F}$. Multiplying both sides of Eq. (3) by W_1^{-1} we get,

$$\vec{w} = (W_1^{-1}\vec{F}). \tag{4}$$

3.1.1 Examples

Example 1: Consider the function $f(x) = x + 1$, where $+$ denotes arithmetic addition. We will use the group D_3 to create our cascade. The truth vector for this function is $\vec{F} = \begin{bmatrix} f(0) \\ f(1) \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$. Hence, from Eq. (4) we have $\vec{w} = (W_1^{-1})\vec{F} = (W_1^{-1}) \begin{bmatrix} 1 \\ 2 \end{bmatrix}$.

We now compute the inverse of the first Walsh Matrix. Since $W_1 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}$,

$$W_1^2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = 2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 2I = 2W_1 W_1^{-1}.$$

Thus, $W_1^{-1} \equiv -W_1 \pmod 3$. Therefore,

$$\vec{w} = -W_1 \vec{F} = \begin{bmatrix} -3 & 1 \end{bmatrix}^T \equiv \begin{bmatrix} 0 & 1 \end{bmatrix}^T \pmod 3.$$

From Eq. (2) we have $a^{f(x)} = a^{w_a} g^x a^{w_b} g^x$. Replacing w_a and w_b with 0 and 1 respectively, we get

$$a^{f(x)} = a^0 g^x a^1 g^x = g^x a^1 g^x.$$

The corresponding cascade diagram with the internal structure of the gates is shown in Figure 11.

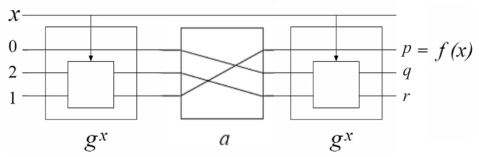

Fig. 11. Canonical Cascade for $f(x) = x + 1$.

Note that only the top target line is used for the function output and the last g gate does not affect that line. Hence we can remove the last g gate without changing our output. Hence, the final expression for this function is $g^x a^{-1}$. The reduced cascade is shown in Figure 12.

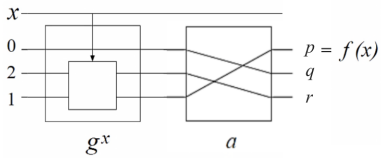

Fig. 12. Reduced canonical cascade for $f(x) = x + 1$ without final g gate.

We now create the circuit for the expression $g^x a^{-1}$. The truth table for the expression $g^x a^{-1}$ is shown in Table 2.

Table 2. $f(x) = x + 1$ cascade truth table.

x	p	q	r
0	1	0	2
1	2	0	1

When $x = 0$, 0 and 2 are swapped and then 2 and 1 are swapped. When $x = 1$, 2 and 0 are swapped. The corresponding circuit diagram is shown in Figure 13.

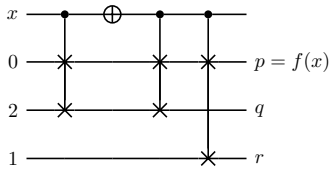

Fig. 13. $f(x) = x + 1$ circuit

Local Transformations: We now apply local transformations to simplify our cascade. Note that we can replace the two controlled swap gates on the first and second target wires with a swap gate. This reduction is shown in Figure 14.

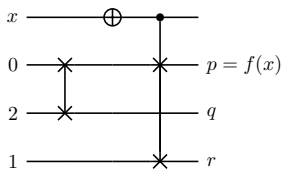

Fig. 14. $f(x) = x + 1$ reduced circuit

However, we can remove the first swap gate and reorder the input signals from "0, 2, 1" to "2, 0, 1." Then, we can remove the inverter and move the output to the bottom wire. The final circuit for $f(x) = x + 1$ is shown in Figure 15.

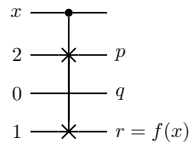

Fig. 15. $f(x) = x + 1$ final circuit

3.2 Two Input Variables

Eq. (1) can be extended to two variables:

$$F(x_1, x_2) = F_a(x_2)g^{x_1}F_b(x_2)g^{x_1}$$

but in this case F_a and F_b are functions of one variable. Figure 16 shows the canonical cascade:

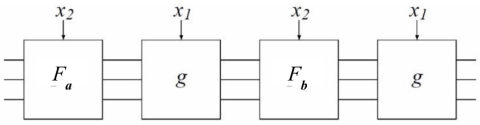

Fig. 16. Intermediate Cascade for two variable functions.

From section 3.1 we found that a one variable function F can be decomposed as $a^{w_a}g^x a^{w_b}g^x$. Since F_a and F_b are functions of one variable, we can replace them with $a^{w_a}g^{x_2}a^{w_b}g^{x_2}$ and $a^{w_c}g^{x_2}a^{w_d}g^{x_2}$ respectively. The canonical form for all functions with two input variables then becomes:

$$a^{f(x_1,x_2)} = ((a^{w_a}g^{x_2}a^{w_b}g^{x_2})g^{x_1})((a^{w_c}g^{x_2}a^{w_d}g^{x_2})g^{x_1}).$$

The cascade has ten cells and is shown in Figure 17.

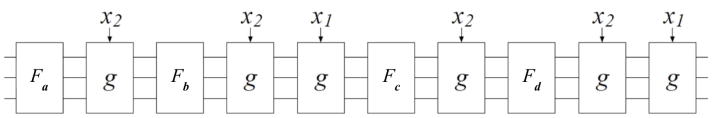

Fig. 17. Cascade for two variable functions after decomposition.

Next we will expand the Walsh matrix to two variables by assigning 0 and 1 to x_1 and x_2 for all four possible combinations and apply vectors to the exponents as was done in section 3.1 for the single variable case. After applying the property of $gag = a^{-1}$ from Section 2.1, the following four equations are derived:

$$a^{f(0,0)} = ((a^{w_a}g^0a^{w_b}g^0)g^0)((a^0g^{x_2}a^0g^{x_2})g^0) = a^{w_a+w_b+w_c+w_d}$$
$$\Rightarrow f(0,0) = w_a + w_b + w_c + w_d.$$

$$a^{f(0,1)} = ((a^{w_a}g^1a^{w_b}g^1)g^0)((a^{w_c}g^1a^{w_d}g^1)g^0) = (a^{w_a}a^{-w_b})(a^{w_c}a^{-w_d}) = a^{w_a-w_b+w_c-w_d}$$
$$\Rightarrow f(0,1) = w_a - w_b + w_c - w_d.$$

$$a^{f(1,0)} = ((a^{w_a}g^0a^{w_b}g^0)g^1)((a^{w_c}g^0a^{w_d}g^0)g^1) = (a^{w_a+w_b})g^1(a^{w_c+w_d})g^1 = a^{w_a+w_b-w_c-w_d}$$
$$\Rightarrow f(1,0) = w_a + w_b - w_c - w_d.$$

$$a^{f(1,1)} = ((a^{w_a}g^1a^{w_b}g^1)g^1)((a^{w_c}g^1a^{w_d}g^1)g^1) = (a^{w_a}a^{-w_b})g^1(a^{w_c}a^{-w_d})g^1 = a^{w_a-w_b-w_c+w_d}$$
$$\Rightarrow f(1,1) = w_a - w_b - w_c + w_d.$$

Putting the above equations in matrix form, we obtain

$$\begin{bmatrix} f(0,0) \\ f(0,1) \\ f(1,0) \\ f(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} w_a \\ w_b \\ w_c \\ w_d \end{bmatrix}.$$

Thus, just as in the one variable case, the elements of the w vector in the canonical cascade can be found using the equation

$$\vec{F} = W_2\vec{w}.$$

$$\vec{w} = (W_2^{-1}\vec{F}). \tag{5}$$

where W_2 is the second Walsh matrix. This will be demonstrated in the next section.

3.2.1 Examples

Example 2: Consider the function $f(x_1, x_2) = x_1 \oplus x_2$, a simple binary XOR function. We will use D_3 to create our cascade. The truth vector for this function is $\vec{F} = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}^T$. Hence, from Eq. (5) we have

$$\vec{w} = (W_2^{-1})\vec{F} = (W_2^{-1}) \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

We now compute the inverse of the second Walsh Matrix.

$$W_2^2 = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} = 4 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = 4I_2 = 4W_2 W_2^{-1}$$

$$\Rightarrow W_2^{-1} \equiv W_2 \pmod 3.$$

Thus,

$$\vec{w} = (W_2^{-1})\vec{F} = W_2 \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \\ -2 \end{bmatrix} \equiv \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \pmod 3.$$

$$\vec{w} = \begin{bmatrix} w_a \\ w_b \\ w_c \\ w_d \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \pmod 3.$$

The group decomposition expression for two variables in canonical form is

$$a^{f(x_1,x_2)} = a^{w_a} g^{x_2} a^{w_b} g^{x_2+x_1} a^{w_c} g^{x_2} a^{w_d} g^{x_2+x_1}.$$

After replacing the exponents for $w_a, w_b, w_c,$ and w_d with the values obtained for $\vec{w}$, we have the following canonical cascade function:

$$a^{f(x_1,x_2)} = a^{-1} g^{x_2} a^0 g^{x_2+x_1} a^0 g^{x_2} a^1 g^{x_2+x_1} = a^{-1} g^{x_1+x_2} a^1 g^{x_1+x_2}.$$

The corresponding cascade is shown in Figure 18.

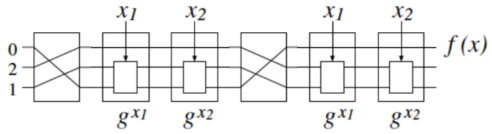

Fig. 18. Cascade for $f(x_1, x_2) = x_1 \oplus x_2$.

Local Transformations: We now apply local transformations to reduce our circuit. Notice that the last two g gates can be removed as they do not affect the top target wire as shown in Figure 19.

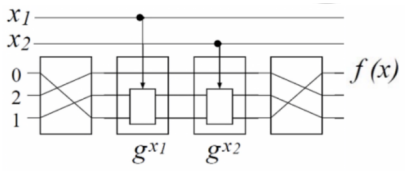

Fig. 19. Reduced Cascade for $f(x_1, x_2) = x_1 \oplus x_2$.

The last a gate can be removed and the output can be moved to the bottom wire, and the first a gate can be removed and the input signals can be rearranged. The final cascade for f is shown in Figure 20.

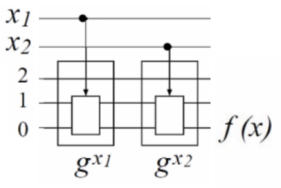

Fig. 20. Final Cascade for $f(x_1, x_2) = x_1 \oplus x_2$.

The quantum circuit for f is shown in Figure 21.

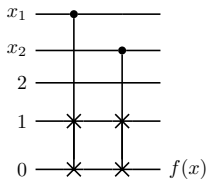

Fig. 21. $f(x) = x_1 \oplus x_2$ circuit

3.3 Three Input Variables

Expression 1 can also be extended to three variables. After decomposition, the following canonical expression is derived:

$$a^{f(x_1,x_2,x_3)} = [[((a^{w_a} g^{x_3} a^{w_b} g^{x_3})g^{x_2})((a^{w_c} g^{x_3} a^{w_d} g^{x_3})g^{x_2})]g^{x_1}][[((a^{w_e} g^{x_3} a^{w_f} g^{x_3})g^{x_2})((a^{w_g} g^{x_3} a^{w_h} g^{x_3})g^{x_2})]g^{x_1}]$$

$$= a^{w_a} g^{x_3} a^{w_b} g^{x_2+x_3} a^{w_c} g^{x_3} a^{w_d} g^{x_1+x_2+x_3} a^{w_e} g^{x_3} a^{w_f} g^{x_2+x_3} a^{w_g} g^{x_3} a^{w_h} g^{x_1+x_2+x_3}.$$

3.3.1 Examples:

Example 3: Let $f(x_1, x_2, x_3) = x_1 + x_2 + x_3$, a binary input, ternary output adder using the group D_3. The truth table for f is shown in Table 3:

Table 3. Modulo 3 Adder Truth Table.

x_3	x_2	x_1	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	2
1	0	0	1
1	0	1	2
1	1	0	2
1	1	1	0

The truth vector for this function is $\vec{F} = [0, 1, 1, 2, 1, 2, 2, 0]^T$. The inverse Walsh matrix ${W_3}^{-1} \equiv -W_3 \pmod 3$, so

$$\vec{w} = ({W_3}^{-1})\vec{F} \equiv -W_3\vec{F} \pmod 3$$

$$= [0, 1, 1, 0, 1, 0, 0, 0]^T \pmod 3.$$

Thus,

$$f(x_1, x_2, x_3) = g^{x_3} a^1 g^{x_2+x_3} a^1 g^{x_1+x_2} a^1 g^{x_1+x_2+x_3}.$$

Since the last g is unused, we can remove it to obtain $f(x_1, x_2, x_3) = g^{x_3} a^1 g^{x_2+x_3} a^1 g^{x_1+x_2} a^1$. The cascade is shown in Figure 22.

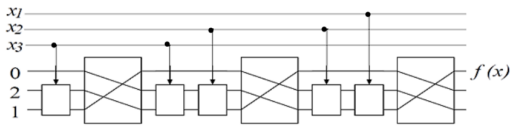

Fig. 22. Canonical Cascade for the modulo three adder of binary arguments, $f(x_1, x_2, x_3) = x_1 + x_2 + x_3$.

Local Transformations: The next phase will be to apply local transformations to simplify the cascade. Consider the first three gates from Figure 22 as shown in Figure 23.

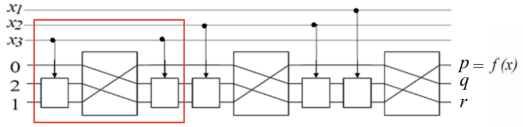

Fig. 23. $g^{x_3} a^1 g^{x_3}$.

The internal structure of these three gates is shown in Figure 24.

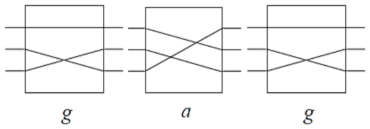

Fig. 24. Internal Structure of $g^{x_3} a^1 g^{x_3}$.

Using $gag = a^{-1}$ when the control qubit is 1, the truth table for the function $g^{x_3} a^1 g^{x_3}$ is shown in Table 4.

Table 4. $g^{x_3} a^1 g^{x_3}$ Truth Table.

x_3	p	q	r
0	1	0	2
1	2	1	0

The binary input, ternary output quantum circuit with binary controlled, ternary Fredkin gates for $g^{x_3} a^1 g^{x_3}$ is shown in Figure 25.

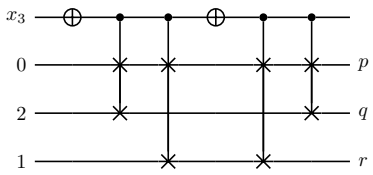

Fig. 25. $g^{x_3}a^1g^{x_3}$ circuit

Note that the second and third controlled swap gates can be removed and replaced with a single swap gate and an inverter. The reduced circuit is shown in Figure 26.

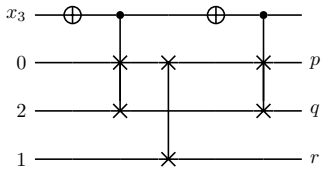

Fig. 26. Reduced $g^{x_3}a^1g^{x_3}$ circuit

Similarly, we can simplify the next three gates in our cascade, $g^{x_2}a^1g^{x_2}$ shown in Figure 27.

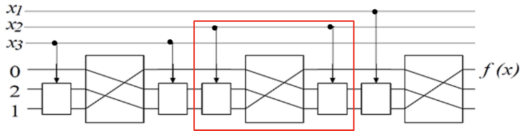

Fig. 27. $g^{x_2}a^1g^{x_2}$.

The circuit for these three gates is shown in Figure 28.

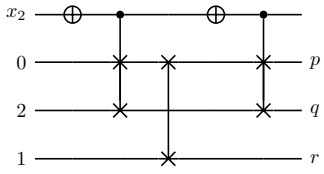

Fig. 28. Reduced $g^{x_2}a^1g^{x_2}$ circuit

Now consider the last a^1 gate as shown in Figure 29.

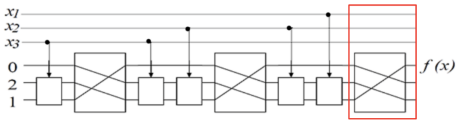

Fig. 29. Last a^1 gate.

It can be removed and we can move the output, $f(x)$, to the bottom wire instead as shown in Figure 30.

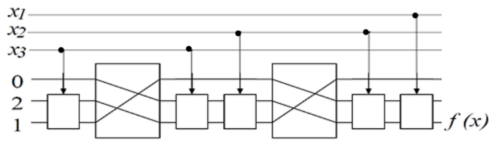

Fig. 30. Simplified cascade with output on bottom wire.

Combining our two reduced circuits and adding the circuit for g^{x_1} we obtain the final circuit for $f(x_1, x_2, x_3) = x_1 + x_2 + x_3$ in Figure 31.

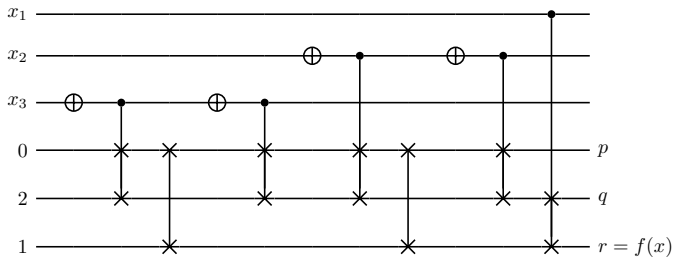

Fig. 31. $f(x_1, x_2, x_3) = x_1 + x_2 + x_3 = g^{x_3} a^1 g^{x_2+x_3} a^1 g^{x_1+x_2} a^1$ circuit

The quantum layout for this circuit is shown in Figure 32.

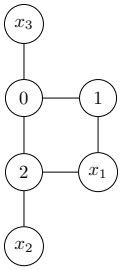

Fig. 32. Quantum Layout for $f(x_1, x_2, x_3) = x_1 + x_2 + x_3$

3.4 Proof of Odd Valued Output

We now prove that the group we use must have an odd valued output. From Eq. (4), in order to determine the Walsh Spectrum of our function, we need to compute the inverse of the Walsh matrix modulo n where n is our valued output. Since $W_i^2 = 2^i I$ where I is the identity matrix, we can write

$$W_i^2 = 2^i W_i W_i^{-1}.$$

Thus, $W_i^{-1} \equiv \frac{1}{2^i} W_i \pmod{n}$. It remains to find the inverse of 2^i modulo n. However, a number, k, only has a multiplicative inverse modulo n if k and n are relatively prime. Hence, in order for there to exist a multiplicative inverse of 2^i modulo n, n must be odd.

4 Local Transformations

The local transformations applied in our methodology are as follows:

(i) Remove any g gates at the end of the cascade as they do not affect the output wire.

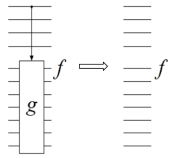

Fig. 33. Removing g gates at the end of the cascade.

(ii) Remove any a gates at the start of the cascade and reorder the input signals.

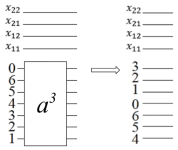

Fig. 34. Reordering the input wires.

(iii) Remove any a gates at the end of the cascade and move the output to a different wire.

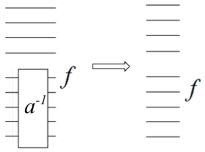

Fig. 35. Changing the output wire.

(iv) Replace any consecutive $g^x a^k g^x$ cells with a^{-k}, a NOT gate, and a^k. ($g^x a^k g^x = a^{-k}$ when $x = 1$ and $g^x a^k g^x = a^k$ when $x = 0$).

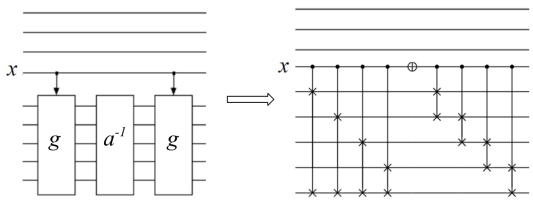

Fig. 36. Reduction of consecutive $g^x a^k g^x$ cells with $k = -1$.

(v) Replace any consecutive $a^k g^x a^k g^x$ cells with a NOT gate and a^{2k}. ($a^k g^x a^k g^x = a^k a^{-k} = I$ when $x = 1$ and $a^k g^x a^k g^x = a^{2k}$ when $x = 0$).

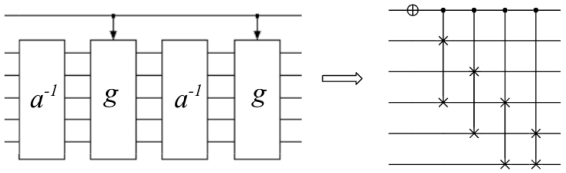

Fig. 37. Reduction of consecutive $a^k g^x a^k g^x$ cells with $k = -1$.

(vi) Replace any consecutive Controlled SWAP gate, a NOT gate, and another Controlled SWAP gate with a single swap gate

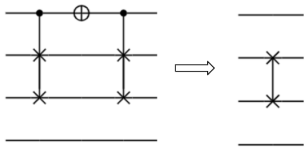

Fig. 38. Reduction of consecutive CSWAP, NOT, CSWAP gates.

(vii) Remove any SWAP gates at the start of the circuit and reorder the inputs of the wires

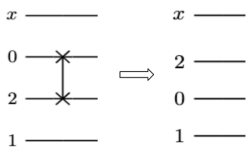

Fig. 39. Removing swap gates at the beginning of the cascade.

5 Examples for 5 and 7 valued Outputs

Example 4: Let $f(x_{22}, x_{12}, x_{21}, x_{11}) = 2(x_{22} + x_{21}) + (x_{12} + x_{11})$, a two-bit, four binary input variable, modulo 7 output adder. We will use the group D_7 to create the cascade. The truth table for f is shown in Table 5.

Table 5. Two-Bit Modulo 7 Adder Truth Table.

x_{22}	x_{12}	x_{21}	x_{11}	$\vec{F}$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	1
0	1	0	1	2
0	1	1	0	3
0	1	1	1	4
1	0	0	0	2
1	0	0	1	3
1	0	1	0	4
1	0	1	1	5
1	1	0	0	3
1	1	0	1	4
1	1	1	0	5
1	1	1	1	6

The truth vector is $\vec{F} = [0,1,2,3,1,2,3,4,2,3,4,5,3,4,5,6]^T$. Since ${W_4}^2 = 2^4 I_4 \equiv 2I_4 \pmod 7$, ${W_4}^{-1} \equiv \frac{1}{2}W_4 \pmod 7$. Thus,

$$\vec{w} = ({W_4}^{-1})\vec{F} \equiv \frac{1}{2}W_4\vec{F} \pmod 7$$

$$= [3,3,-1,0,3,0,0,0,-1,0,0,0,0,0,0,0]^T \pmod 7$$

After simplification, the canonical cascade is the following:

$$f = a^3 g^{x_{11}} a^3 g^{x_{11}} g^{x_{21}} a^{-1} g^{x_{21}} g^{x_{12}} a^3 g^{x_{21}} g^{x_{22}} a^{-1}$$

The canonical cascade circuit diagram for f is shown in Figure 40.

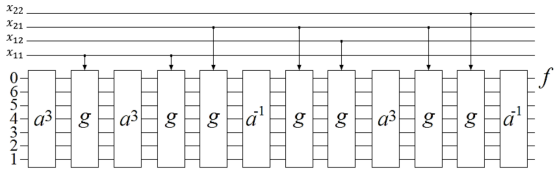

Fig. 40. Reduced Canonical Cascade for two bit modulo 7 adder, $f = 2(x_{22} + x_{21}) + (x_{12} + x_{11})$.

The internal structure of these gates is shown in Figure 41:

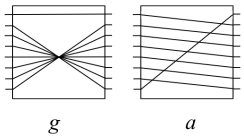

Fig. 41. Internal Structure of g and a gates.

Local Transformations: The next phase will be to apply local transformations to simplify the cascade. Consider the first four cells shown in Figure 42.

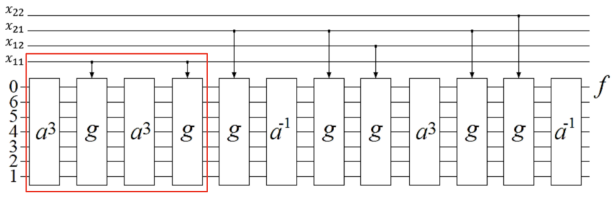

Fig. 42. $a^3 g^{x_{11}} a^3 g^{x_{11}}$.

Since $gag = a^{-1}$ when the control qubit is 1, when $x_{11} = 1$, the first four gates evaluate to $a^3 a^{-3} = I$. When the control qubit is 0, the first four gates evaluate to $a^3 a^3 = a^{-1}$. The circuit for $a^3 g^{x_{11}} a^3 g^{x_{11}}$ is shown in Figure 43.

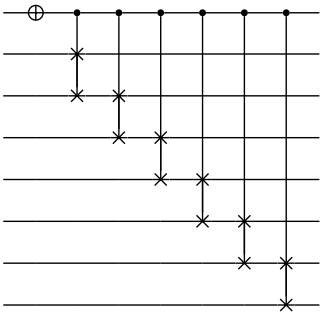

Fig. 43. $a^3 g^{x_{11}} a^3 g^{x_{11}}$ circuit

Similarly, we can simplify the three gates shown in Figure 44.

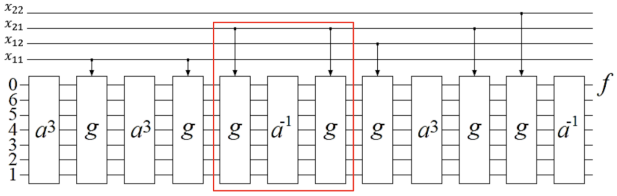

Fig. 44. $g^{x_{21}} a^{-1} g^{x_{21}}$.

When $x_{21} = 1$, $g^{x_{21}} a^{-1} g^{x_{21}} = a^1$. When $x_{21} = 0$, $g^{x_{21}} a^{-1} g^{x_{21}} = a^6$. The truth table for $g^{x_{21}} a^{-1} g^{x_{21}}$ is shown in Table 6:

Table 6. $g^{x_{21}} a^{-1} g^{x_{21}}$ Truth Table.

x_{21}	p	q	r	s	t	u	v
0	1	2	3	4	5	6	0
1	6	0	1	2	3	4	5

The circuit for $g^{x_{21}} a^{-1} g^{x_{21}}$ is shown in Figure 45.

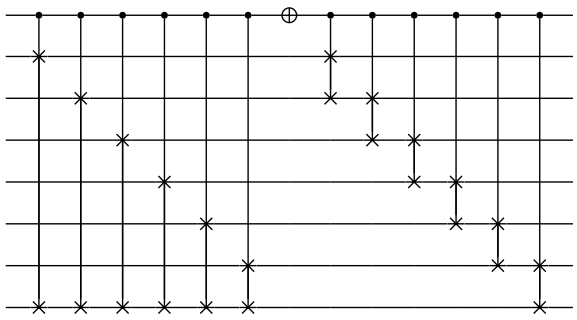

Fig. 45. $g^{x_{21}} a^{-1} g^{x_{21}}$ reduced circuit

Next, notice that we can remove the final a^{-1} gate by moving the output, $f(x)$, to the second wire.

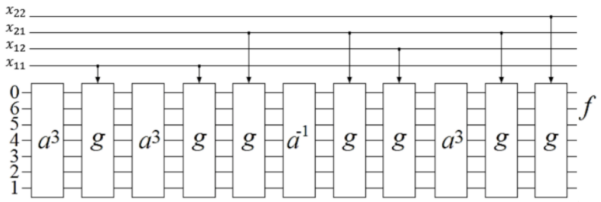

Fig. 46. Reduced cascade after moving output to the second wire.

Finally using our reduced circuits, we build the complete circuit for f as shown in Figure 47.

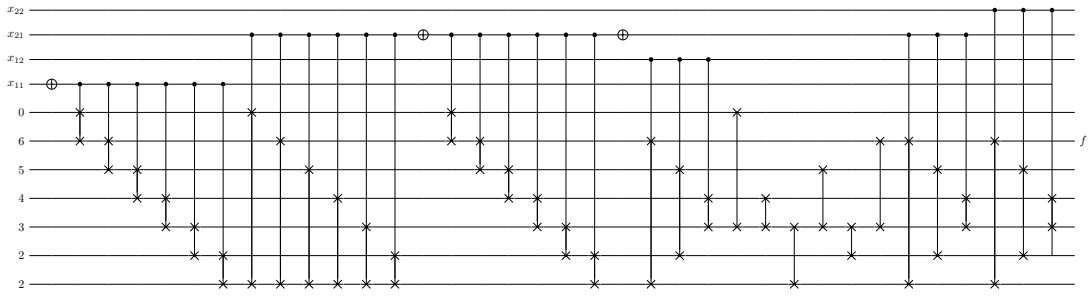

Fig. 47. $f(x_{22}, x_{12}, x_{21}, x_{11}) = 2(x_{22} + x_{21}) + (x_{12} + x_{11})$ Circuit

Example 5: Consider the binary input, modulo 5 output function f with the truth table as shown in Table 7.

Table 7. Function Truth Table.

x_{22}	x_{12}	x_{21}	x_{11}	$\vec{F}$
0	0	0	0	4
0	0	0	1	3
0	0	1	0	2
0	0	1	1	0
0	1	0	0	3
0	1	0	1	4
0	1	1	0	3
0	1	1	1	1
1	0	0	0	3
1	0	0	1	0
1	0	1	0	2
1	0	1	1	4
1	1	0	0	0
1	1	0	1	4
1	1	1	0	1
1	1	1	1	4

The truth vector is $\vec{F} = [4,3,2,0,3,4,3,1,3,0,2,4,0,4,1,4]^T$. Since ${W_4}^2 = 2^4 I_4 \equiv I_4 \pmod 5$, ${W_4}^{-1} \equiv W_4 \pmod 5$. Thus,

$$\vec{w} = ({W_4}^{-1})\vec{F} \equiv W_4\vec{F} \pmod 5$$

$$= [3,3,-1,0,3,0,0,0,-1,0,0,0,0,0,0,0]^T \pmod 5.$$

After simplification, the canonical cascade is the following:

$$f = a^3 g^{x_{11}} a^3 g^{x_{11}} g^{x_{21}} a^{-1} g^{x_{21}} g^{x_{12}} a^3 g^{x_{21}} g^{x_{22}} a^{-1}.$$

The canonical cascade circuit diagram for f is shown in Figure 48.

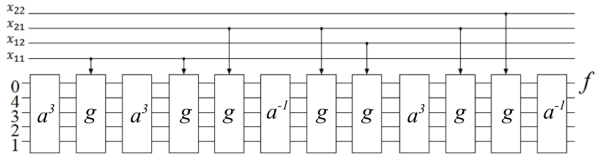

Fig. 48. Reduced Canonical Cascade for f.

Local Transformations: The next phase will be to apply local transformations to simplify the cascade. Consider the first four gates shown in Figure 49.

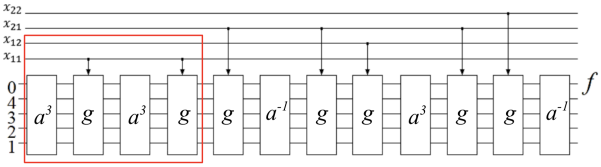

Fig. 49. $g^{x_{11}} a^3 g^{x_{11}}$.

Since $gag = a^{-1}$ when the control qubit is 1, $a^3 g a^3 g = a^3(a^{-3}) = I$ when $x_{11} = 1$. When $x_{11} = 0$, $a^3 g a^3 g = a^6 = a$. The circuit for $a^3 g^{x_{11}} a^3 g^{x_{11}}$ is shown in Figure 50.

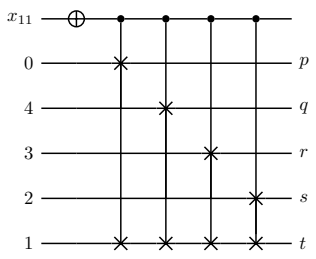

Fig. 50. $a^3 g^{x_{11}} a^3 g^{x_{11}}$ reduced circuit

Similarly, we can simplify the three gates shown in Figure 51.

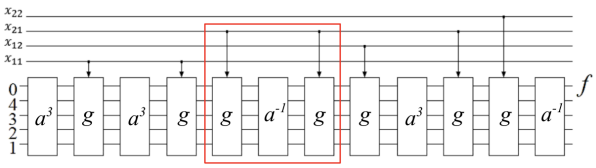

Fig. 51. $g^{x_{21}} a^{-1} g^{x_{21}}$.

The truth table for $g^{x_{21}} a^{-1} g^{x_{21}}$ is shown in Table 8:

Table 8. $g^{x_{21}} a^{-1} g^{x_{21}}$ Truth Table.

x_{21}	p	q	r	s	t
0	4	3	2	1	0
1	1	0	4	3	2

The circuit for $g^{x_{21}} a^{-1} g^{x_{21}}$ is shown in Figure 52.

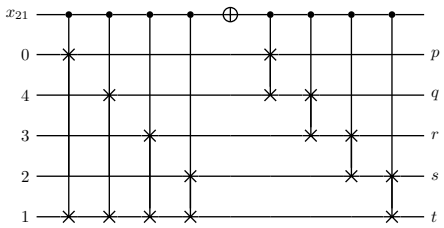

Fig. 52. $g^{x_{21}} a^{-1} g^{x_{21}}$ reduced circuit

We can also remove the last a^{-1} gate and move the output to the second wire. Using our reduced circuits, we build the final circuit for f as shown in Figure 53.

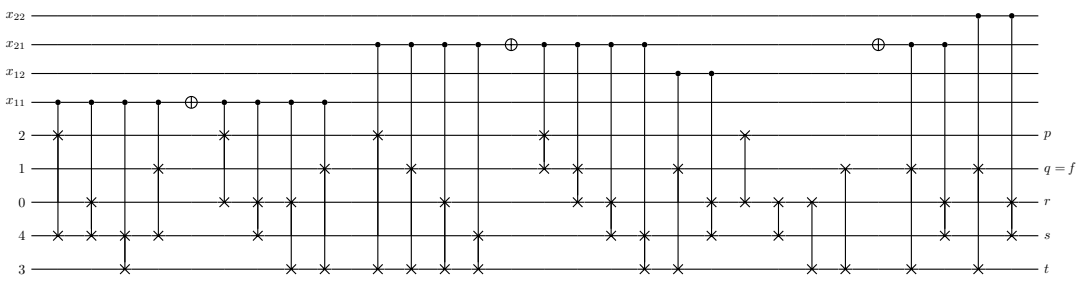

Fig. 53. Circuit for f

Example 6: Consider the function $f(x_1, x_2, x_3, x_4) = 2(x_1 + x_3) + (x_2 + x_4)$, the two-bit, four binary input variable, modulo 7 output adder from Eq. (5). We propose another method to realize this function using the group D_3 with multiple outputs. The map of this function is shown in Figure 54:

x_1, x_2 \ x_3, x_4	00	01	11	10
00	0	1	3	2
01	1	2	4	3
11	3	4	6	5
10	2	3	5	4

Fig. 54. Map for function $f(x_1, x_2, x_3, x_4) = 2(x_1 + x_3) + (x_2 + x_4)$

The values from 0 to 6 can be written as follows in ternary: $0 - 00_3$; $1 - 01_3$; $2 - 02_3$; $3 - 10_3$; $4 - 11_3$; $5 - 12_3$; and $6 - 20_3$. Replacing the values in our map in Figure 54, we obtain Figure 55:

x_1, x_2 \ x_3, x_4	00	01	11	10
00	00	01	10	02
01	01	02	11	10
11	10	11	20	12
10	02	10	12	11

Fig. 55. Ternary map for function $f(x_1, x_2, x_3, x_4) = 2(x_1 + x_3) + (x_2 + x_4)$

We can split this map into two by considering the first and second digits separately as shown in Figure 57.

x_1, x_2 \ x_3, x_4	00	01	11	10
00	0	0	1	0
01	0	0	1	1
11	1	1	2	1
10	0	1	1	1

(a) Submap of First Digit

x_1, x_2 \ x_3, x_4	00	01	11	10
00	0	1	0	2
01	1	2	1	0
11	0	1	0	2
10	2	0	2	1

(b) Submap of Second Digit

Fig. 57. Two submaps of ternary map of f

We will now compute the canonical expression for submap a from Figure 57. The truth vector of submap a is $[0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 2]^T$. After simplification, the canonical expression is the following:

$$g^{x_3+x_4}a^1g^{x_2+x_4}a^1g^{x_4}a^1g^{x_1+x_2+x_3}a^1g^{x_3+x_4}a^1g^{x_4}a^1g^{x_2+x_3+x_4}a^1g^{x_4}a^1g^{x_3+x_4}a^1g^{x_4}a^2.$$

Similarly, the truth vector for submap b is $[0\ 1\ 2\ 0\ 1\ 2\ 0\ 1\ 2\ 0\ 1\ 2\ 0\ 1\ 2\ 0]^T$. After simplification, the canonical expression is the following:

$$g^{x_4}a^1g^{x_3+x_4}a^2g^{x_2+x_3}a^1g^{x_4}a^2g^{x_3}a^1g^{x_1+x_2+x_3+x_4}a^2g^{x_3}a^1g^{x_4}a^2g^{x_2+x_3}a^1g^{x_3+x_4}a^2.$$

Using these two expressions, we can create the cascade and circuit for the function.

6 Upper Bound on Number of Gates

Lemma 1: The maximum number of cells in an arbitrary n-variable input cascade is $3 * 2^n - 4 - n$.

Proof: Recall that

$$F(x_1, x_2, \ldots x_n, x_{n+1}) = F_a(x_2, \ldots x_{n+1})g^{x_1}F_b(x_2, \ldots x_{n+1})g^{x_1}.$$

Let $A(n)$ denote the number of a gates in the n-variable input cascade before reduction. Then,

$$A(n+1) = 2A(n).$$

We can solve this recurrence relation using its characteristic equation [63]:

$$\lambda - 2 = 0$$

$$\Rightarrow A(n) = k2^n.$$

Since $A(1) = 2$, we have $k = 1$, so $A(n) = 2^k$

Let $G(n)$ denote the number of g gates in the n-variable input cascade before reduction. Then,

$$G(n+1) = 2G(n) + 2.$$

This recurrence relation has characteristic equation:

$$\lambda^2 - 3\lambda + 2 = 0$$

$$(\lambda - 1)(\lambda - 2) = 0$$

$$\Rightarrow G(n) = k_1 2^n + k_2 1^n.$$

Since $G(1) = 2, G(2) = 6$, we have $k_1 = 2, k_2 = -2$, so $G(n) = 2 * 2^n - 2$.

Note that the first and last a gates and the last g gates can always be removed to reduce the cascade. Thus, after reduction, the maximum number of a gates is $2^n - 2$ and the maximum number of g gates is $2 * 2^n - 2 - n$. Hence, the total number of gates in the cascade is $3 * 2^n - 4 - n$. □

Note that for a k-valued output function, each a gate can be realized using $k-1$ SWAP gates and each g gate can be realized using $\frac{k-1}{2}$ multivalued Fredkin Gates. Hence, by Lemma 1 the maximum number of SWAP or Fredkin gates in an n-variable input with k-valued output quantum circuit is $(k-1)(2^n - 2) + \frac{k-1}{2}(2 * 2^n - 2 - n)$. More specifically, with a maximum of $(k-1)(2^n - 2)$ SWAP gates and $\frac{k-1}{2}(2 * 2^n - 2 - n)$ Fredkin gates.

7 Conclusion

In the past, several authors developed methods to use group theory to design classical binary logical cascades [2, 9, 10, 16]. In this paper, we have extended the decomposition from [2, 3] and have created binary input and multivalued output quantum cascades using NOT, SWAP, and Controlled SWAP gates. The choice of our gates was determined by what is currently implementable on optical technologies, and the design of our cascades was motivated by practical quantum layouts. We have also created a method to realize a function with different valued outputs and have developed seven local transformations to simplify the final cascade circuits. Through these simplifications, we have shown that an arbitrary n-variable input cascade has a maximum of $3 * 2^n - 4 - n$ cells, and the maximum number of individual multivalued Fredkin and SWAP gates in an arbitrary n-variable input and k-valued output function is $(k-1)(2^n - 2) + \frac{k-1}{2}(2 * 2^n - 2 - n)$.

Acknowledgements

We gratefully acknowledge the comments and suggestions of the reviewer. Their detailed feedback helped us improve our paper greatly.

References

1. P. Litwin, J. Wroński, K. Markowski, D. Lopez-Mago, J. Masajada, and M Szatkowski (2024), *Ternary logic in the optical controlled-SWAP gate based on Laguerre-Gaussian modes of light*, Optics Express, Vol.32, pp. 15258-15268.
2. Hurst (1984), *Multiple-valued logic—Its status and its future*, IEEE Transactions on Computers, Vol.33, pp. 1160-1179.
3. K.C. Smith (1988), *A multiple valued logic: a tutorial and appreciation*, Computer, Vol.21, pp. 17-27.
4. A. Bérut, A. Arakelyan, A. Petrosyan, S. Ciliberto, R. Dillenschneider, and E. Lutz (2012), *Experimental verification of Landauer's principle linking information and thermodynamics*, Nature, Vol.483, pp. 187-189.
5. R. Landauer (1988), *Dissipation and noise immunity in computation and communication*, Nature, Vol.335, pp. 779-784.

6. R. Landauer (1961), *Irreversibility and heat generation in the computing process*, IBM Journal of Research and Development, Vol.5, pp. 183-191.
7. R, Hamerly, L. Bernstein, A. Sludds, M. Soljačić, and D. Englund (2019), *Large-scale optical neural networks based on photoelectric multiplication*, Phys. Rev. X, Vol.9, pp. 021032.
8. J. Li, N. Li, J. Peng, H. Cui, and Z. Wu (2019), *Energy consumption of cryptocurrency mining: A study of electricity consumption in mining cryptocurrencies*, Energy, Vol.168, pp. 160-168.
9. E. Pelucchi, G. Fagas, T. Aharonovich, D. Englund, E. Figueroa, Q, Gong, H. Hannes, J. Liu, C.Y. Lu, N. Matsuda, and J.W. Pan (2022), *The potential and global outlook of integrated photonics for quantum technologies*, Nature Reviews Physics, Vol.4, pp. 194-208.
10. T. Chattopadhyay (2011), *All-optical modified Fredkin gate*, IEEE Journal of Selected Topics in Quantum Electronics, Vol.18, pp. 585-592.
11. M. Ringbauer, M. Meth, L. Postler, R. Stricker, R. Blatt, P. Schindler, and T. Monz (2022), *A universal qudit quantum processor with trapped ions*, Nature Physics, Vol.18, pp. 1053-1057.
12. Y. Imagi and Y. Ohtsuka (1987), *Optical multiple-output and multiple-valued logic operation based on fringe shifting techniques using a spatial light modulator*, Applied Optics, Vol.26, pp. 274-277.
13. J. Yi, H. Huacan, and L. Yangtian (2005), *Ternary optical computer architecture*, Physica Scripta, Vol.T118, pp. 98–101.
14. T. Chattopadhyay (2010), *All-optical symmetric ternary logic gate*, Optics & Laser Technology, Vol.42, pp. 1014-1021.
15. S. Mukhopadhyay (2010), *Role of optics in super-fast information processing*, Indian Journal of Physics, Vol.84, pp. 1069-1074.
16. W.N. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski (2006), *Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.25, pp. 1652-1663.
17. M. Saraivanov and M. Perkowski (2018), *Multi-valued Quantum Cascade Realization with Group Decomposition*, International Symposium on Multiple-Valued Logic
18. M. Saraivanov (2013), *Quantum Circuit Synthesis using Group Decomposition and Hilbert Spaces*, MS thesis, Portland State University
19. D.M. Miller and G.W. Dueck (2021), *Search-based transformation synthesis for 3-valued reversible circuits*, Reversible Computation: 12th International Conference.
20. M. Soeken, G.W. Dueck, M.M. Rahman, and D.M. Miller (2016), *An extension of transformation-based reversible and quantum circuit synthesis*, IEEE International Symposium on Circuits and Systems.
21. M. Hawash and M. Perkowski (2012), *Using Hasse diagrams to synthesize ternary quantum circuits*, IEEE International Symposium on Multiple-Valued Logic.
22. C. Moraga (2016), *Aspects of Reversible and Quantum Computing in a p-Valued Domain*, IEEE Journal on Emerging and Selected Topics in Circuits and Systems, Vol.6, pp. 44-52.
23. M. Khan, H. Thapliyal, and E. Munoz-Coreas (2017), *Automatic synthesis of quaternary quantum circuit*, The Journal of Supercomputing, Vol.73, pp. 1733-1759.
24. Y.M Di and H.R. Wei (2013), *Synthesis of multivalued quantum logic circuits by elementary gates*, Phys. Rev. A, Vol.97, pp. 012325.
25. T. Sasao (2003), *Cascade realizations of two-valued input multiple-valued output functions using decomposition of group functions*, International Symposium on Multiple-Valued Logic.

26. E. Bernard and H.S. Stone (1967), *Decomposition of group functions and the synthesis of multirail cascades*, Symposium on Switching and Automata Theory.
27. M. Yoeli and J. Turner (1967), *Decompositions of group functions with applications to two-rail cascades*, Information and Control, Vol.10, pp. 565-571.
28. P. Delsarte and J. Quisquater (1973), *Permutation cascades with normalized cells*, Information and Control, Vol.23, pp. 344-356.
29. A. Bhattacharya, G.K. Maity, and A.K. Ghosh (2017), *Optical quadruple Toffoli and Fredkin gate using SLM and Savart plate*, Computational Intelligence, Communications, and Business Analytics Conference.
30. G.J. Milburn (1989), *Quantum optical Fredkin gate*, Phys. Rev. Letters, Vol.62, pp. 2124.
31. J.L. O'Brien (2003), *Demonstration of an all-optical quantum controlled-NOT gate*, Nature, Vol.426, pp. 264-267.
32. J.H. Lopes, W.C. Soares, B.L. Bernardo, D.P. Caetano, and A. Canabarro (2019), *Linear optical CNOT gate with orbital angular momentum and polarization*, Quantum Information Processing, Vol.18, pp. 1-10.
33. R.B. Patel, J. Ho, F. Ferreyrol, T.C. Ralph, and G.J. Pryde (2016), *A quantum Fredkin gate*, Science Advances, Vol.2, pp. e1501531.
34. F. Wang, S. Ru, Y. Wang, M. An, P. Zhang, and F. Li (2021), *Experimental demonstration of a quantum controlled-SWAP gate with multiple degrees of freedom of a single photon*, Quantum Science and Technology, Vol.6, pp. 035005.
35. S.K. Garai (2014), *A novel method of developing all optical frequency encoded Fredkin gates*, Optics Communications, Vol.313, pp. 441-447.
36. M. Perkowski, M. Lukac, D. Shah, annd M Kameyama (2011), *Synthesis of quantum circuits in linear nearest neighbor model using positive Davio lattices*, Facta Universitatis-Series: Electronics and Energetics.
37. M. Whitney, N. Isailovic, Y. Patel, and J. Kubiatowicz (2007), *Automated generation of layout and control for quantum circuits*, International Conference on Computing Frontiers.
38. M. Lukac, S. Nursultan, G. Krylov, and O. Keszöcze (2020), *Geometric Refactoring of Quantum and Reversible Circuits: Quantum Layout*, Euromicro Conference on Digital System Design.
39. B. Tan and J. Cong (2020), *Optimal layout synthesis for quantum computing*, International Conference on Computer-Aided Design.
40. Y.M. Di and H.R. Wei (2011), *Elementary gates for ternary quantum logic circuit*, quant-ph/1105.5485.
41. Y. Wang, Z. Hu, B.C. Sanders, S. Kais (2020), *Qudits and high-dimensional quantum computing*, Frontiers in Physics, Vol.8, pp. 589504.
42. D. Mandal, S. Mandal, and S.K. Garai (2015), *Alternative approach of developing all-optical Fredkin and Toffoli gates*, Optics & Laser Technology, Vol.72, pp. 33-41.
43. D.Y. Feinstein and M.A. Thornton (2012), *Using the Asynchronous Paradigm for Reversible Sequential Circuit Implementation*, IEEE International Symposium on Multiple-Valued Logic.
44. P.D. Picton (1994), *Fredkin gates as a basis for comparison of different logic design solutions*, IEE Colloquium on Synthesis and Optimisation of Logic Systems.
45. B. Wang and L.M. Duan (2007), *Implementation scheme of controlled SWAP gates for quantum fingerprinting and photonic quantum computation*, Phys. Rev, A, Vol.75, pp. 050304.
46. J. Shamir, H.J. Caulfield, W. Micelli, and R.J. Seymour (1986), *Optical computing and the Fredkin gates*, Applied Optics, Vol.25, pp. 1604-1607.

47. R. Cuykendall and D. McMillin (1987), *Control-specific optical Fredkin circuits*, Applied Optics, Vol.26, pp. 1959-1963.
48. M.M. Mirsalehi, J. Shamir, and H.J. Caulfield (1987), *Residue arithmetic processing utilizing optical Fredkin gate arrays*, Applied Optics, Vol.26, pp. 3940-3946.
49. A.J. Poustie and K.J. Blow (2000), *Demonstration of an all-optical Fredkin gate*, Optics Communications, Vol.174, pp. 317-320.
50. S. Kotiyal, H. Thapliyal, and N. Ranganathan (2010), *Design of a ternary barrel shifter using multiple-valued reversible logic*, IEEE International Conference on Nanotechnology.
51. V. Deibuk, I. Turchenko, and V. Shults (2015), *Optimized design of the universal ternary gates for quantum/reversible computing*, IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems.
52. F.S. Khan and M. Perkowski (2006), *Synthesis of multi-qudit hybrid and d-valued quantum logic circuits by decomposition*, Theoretical Computer Science, Vol.367, pp. 336-346.
53. A. Muthukrishnan and C.R. Stroud Jr. (2000), *Multivalued logic gates for quantum computation*, Phys. Rev. A, Vol.62, pp. 052309.
54. J.L. O'Brien (2007), *Optical quantum computing*, Science, Vol.318, pp. 1567-1570.
55. T.C. Ralph, K.J. Resch, and A. Gilchrist (2007), *Efficient Toffoli gates Using Qudits*, Phys. Rev. A, Vol.75, pp. 022313.
56. J. Pedersen, *Groups of Small Order*, mathweb.ucsd.edu/ atparris/small_groups.html. Accessed 3 Dec. 2024.
57. S. Ghosh, S. Bhunia, and K. Roy (2005), *Shannon expansion based supply-gated logic for improved power and testability*, IEEE Asian Test Symposium.
58. B.J. Falkowski and M. Perkowski (1990), *Walsh Type Transforms for Completely and Incompletely Specified Multiple-Valued Input Binary Functions*, International Symposium on Multiple-Valued Logic.
59. B.J. Falkowski, I. Schafer, and M. Perkowski (1992), *Effective computer methods for the calculation of Rademacher-Walsh spectrum for completely and incompletely specified Boolean functions*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.11, pp. 1207-1226.
60. B.J. Falkowski (1994), *Properties and ways of calculation of multi-polarity generalized Walsh transforms*, IEEE Transactions on Circuits and Systems II, Vol.41, pp. 380-391.
61. M.A. Thornton, R. Drechsler, and D.M. Miller (2012), *Spectral Techniques in VLSI CAD*, Springer Science & Business Media.
62. K.W. Henderson (1964), *Some notes on the Walsh functions*, IEEE Transactions on Electronic Computers, Vol.1, pp. 50-52.
63. G.E. Martin (2001), *Counting: The Art of Enumerative Combinatorics*, Springer.