1-15-2020

A Polarity-Based Approach for Optimization of Multivalued Quantum Multiplexers with Arbitrary Single-Qubit Target Gates

Kevin Jin
University of California, Berkeley

Tahsin Soffat
Massachusetts Institute of Technology

Justin Morgan
Portland State University

Marek Perkowski
Portland State University, marek.perkowski@pdx.edu

Citation Details

Jin, Kevin; Soffat, Tahsin; Morgan, Justin; and Perkowski, Marek. (2020) A Polarity-Based Approach for Optimization of Multivalued Quantum Multiplexers with Arbitrary Single-Qubit Target Gates. Journal of Applied Logics — IfCoLog Journal of Logics and their Applications, vol. 7, no. 1, Jan. 15 2020.

A Polarity-based Approach for Optimization of Multivalued Quantum Multiplexers with Arbitrary Single-qubit Target Gates

Kevin Jin
University of California, Berkeley
kevindujin@berkeley.edu

Tahsin Saffat
Massachusetts Institute of Technology
tahsin.saffat@gmail.com

Justin Morgan
Portland State University
jumorgan@pdx.edu

Marek Perkowski
Portland State University
h8mp@pdx.edu

Abstract

Previous work has provided methods for decomposing unitary matrices to series of quantum multiplexers, but the multiplexer circuits created in this way may be highly non-minimal. This paper presents a new approach for optimizing quantum multiplexers with arbitrary single-qubit quantum target functions and ternary controls. For multivalued quantum multiplexers, we define standard forms and two types of new forms: Fixed Polarity Quantum Forms (FPQFs) and Kronecker Quantum Forms (KQFs). Drawing inspiration from the usage of butterfly diagrams, we devise a method to exhaustively construct new forms. In contrast to previous butterfly-based methods, which are used with classical Boolean functions, these new forms are used to optimize quantum circuits with arbitrary target unitary matrices. Experimental results on the new forms applied to various target gates such as NOT, V, V^+, Hadamard, and Pauli rotations, demonstrate that these new forms greatly reduce the gate costs of ternary quantum multiplexers.

Vol. 7 No. 1 2020
Journal of Applied Logics — IfCoLog Journal of Logics and their Applications

1 Introduction

Previous works in the field of quantum compilation, such as [1, 22], have generated methods for decomposition of arbitrary unitary matrices into a series of quantum multiplexers, but the multiplexers created by these methods may be highly non-minimal. Several papers are related to the synthesis of binary quantum circuits with controlled V and V^+ gates [17, 18, 20, 21, 22, 23, 24, 25, 26, 27]. Although quantum computers now allow for multivalued logic, very little has been published on circuit synthesis with ternary or higher-order multivalued multiplexers. Our method addresses the case where the gates of the multiplexers are controlled by many ternary inputs, but target only a single binary qubit. For this case, our results are significantly less expensive, or even exactly minimal. This paper, by inventing the new concept of polarity-based multiplexers with arbitrary targets, gives a new methodology for optimization of ternary-input, binary-output quantum circuits. Because there are no benchmarks for these kinds of quantum circuits, we use randomly generated benchmark data and some examples from previous papers. We also briefly introduce the concept of ternary-controlled multiplexers with ternary target gates (ternary-input, ternary-output), but our numerical results are restricted to ternary-input, binary-output multiplexers.

Section 2 gives necessary background for our paper, introducing previous research and concepts. Section 3 introduces new multi-valued multiplexer forms, Fixed Polarity Quantum Forms (FPQFs), and Kronecker Quantum Forms (KQFs), which generalize the ideas from binary Reed Muller forms to forms that control arbitrary single-qubit gates. This section also includes an explanation of why our butterfly decompositions work for transforming standard form multiplexers to FPQFs and KQFs. Section 4 discusses the program used to compute matrix transformations and FPQF/KQF forms and also introduces how costs were calculated for these new types of generalized multiplexers. Section 5 analyses and discusses the results generated by our program. Because the best improvements for general multiplexers were found for multiplexers that control V, V^+, and NOT gates, we analyze these results as a special case. Such circuits are created as intermediate results in the methods from previous research discussed in Section 2. Section 6 draws conclusions and summarizes the paper.

2 Background

2.1 General background on quantum multiplexers and their place in quantum circuit synthesis

The **quantum multiplexer** is an important concept in quantum circuit synthesis. In previous literature, it was related to multi-qubit control of target gates, which are theoretically arbitrary. However, most of the earlier work on quantum multiplexers was related to multi-controlled-NOT gates, such as Toffoli. For instance, this was done for the synthesis of ESOP circuits as well as some special cases of ESOP, such as Fixed Polarity Reed-Muller (FPRM) and Kronecker Reed-Muller (KRM). Very little has been published on optimizing similar ternary-controlled forms and circuits; a rare example is [28]. While some methods decompose arbitrary unitary matrices to smaller binary-controlled quantum multiplexers [1, 2], only [22] covers the ternary-controlled case. In our paper, we are not interested in this initial decomposition stage, nor are we interested in the specific optimization of ternary counterparts of classical Reed-Muller-like forms. Rather, we introduce a new concept of butterfly-like diagrams to optimize ternary-controlled quantum multiplexers with arbitrary target single-qubit gates; this is in contrast to multiplexers in previous work [14, 15], where multi-controlled, binary-controlled gates (such as Toffoli) only control NOT gates as target.

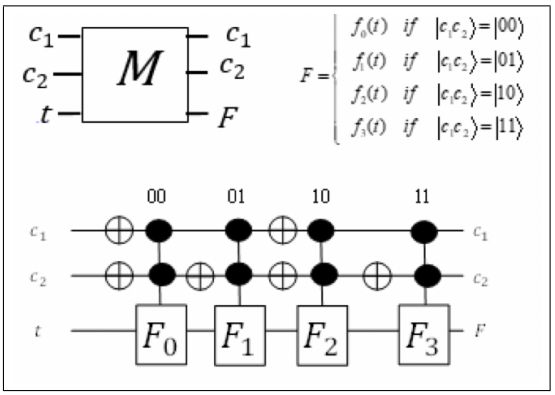

Figure 1: An arbitrary binary standard form quantum multiplexer with two controls, represented both mathematically and as a circuit diagram. The controls are listed above each function.

The concept of a more general quantum multiplexer was first introduced by Shende et. al. in [3], where they propose the quantum multiplexer circuit block for usage in recursive decompositions of arbitrary unitary matrices. Other work by Shende et. al. in [4] focused on optimization of two-qubit unitary operators, but not on optimization of larger circuits such as quantum multiplexers. In [2], Vartiainen et. al. demonstrate a method for optimization of arbitrary multi-qubit gates, but do not provide a method specifically for optimization of the less general multi-qubit multiplexers. Tucci developed the Qubiter program in [1] to decompose

arbitrary unitary matrices to series of quantum multiplexers, but a later work by Hutsell [9] has shown that the output from Qubiter is highly non-minimal, leaving plenty of space for optimization. There are high degrees of similarities between lookup-tables (LUTs) and multiplexers, but the multiplexers introduced by us have arbitrary target gates, in contrast with LUTs, which only have NOTs as target gates. Work by Soeken et. al. [14, 15] mapped networks of LUTs to reversible gates; our work creates large LUTs, controlling arbitrary targets, to realize the entire function. Work by Gidney [13] reduced the cost of adder circuits by reducing the number of T Gates required, while our method allows for arbitrary target gates, although we did not use T Gates.

Yet another application of quantum multiplexers is not to control inverters only, but rather gates that are square roots of NOTs, such as V and V^+, and also other higher-order roots of NOT and their Hermitians [17, 18, 20, 21, 22, 23, 24, 25, 26, 27]. The papers in this area create circuits with multi-controlled gates that can be optimized. For example, a Controlled-V gate composed with a Controlled-V^+ gate creates a two-qubit Identity, which means that these two gates can be removed from the circuit.

In this paper, we propose a method that will take into account all the aforementioned applications of quantum multiplexers.

2.2 Multivalued logic

A q-valued quantum dit is a qudit with q different basis states, which we will call $|0\rangle, |1\rangle, \ldots, |q-1\rangle$. A set of m q-valued qudits can occupy q^m different basis states, and their collective state is a superposition of these basis states.

The single qudit gates for q-valued logic can be represented by arbitrary permutative $q \times q$ matrices. For instance, in ternary logic, these quantum gates are $(+1), (+2), (01), (02)$, and (12), where:

$$(+1) = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}; \qquad (+2) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

The other ternary gates are not important to our paper. Cyclic inverters are an important type of multivalued gates. These are generalizations of the NOT for multi valued quantum logic. We define the cyclic inverter r_k as the gate that takes a qubit in the state $|a\rangle$ to the state $|a+k\rangle$, where the addition is taken modulo q. In the case of ternary, $(+1)$ and $(+2)$ are both cyclic inverters.

2.3 Binary quantum multiplexers and controlled gates

For didactic reasons, let us first define the binary quantum multiplexer. A **binary quantum multiplexer** is a block of gates that will, based on a set of binary-valued control variables (qubits), use a Boolean product of the control literals to select an arbitrary unitary binary-valued quantum function (called a **target function**, or a **target gate**) that acts on a single target variable (qubit): see Figure 1. Note that in the diagram, the black circles denote âĂIJcontrolsâĂİ. That is, if the signal on the line is $|1\rangle$ when it reaches the control, then the control is enabled. If all controls of a particular target gate are enabled, then the gate will activate. Thus, when taken together, the controls of a target function form its **control function** (the previously mentioned product of control literals). For example, the function F_0 is controlled by both $\bar{c_1}$ and $\bar{c_2}$, which means that the control function is $\bar{c_1}\bar{c_2}$, and the target function is active if and only if $c_1 = 0$ and $c_2 = 0$. Target functions can be any single-qubit quantum functions. Thus, they are represented by unitary 2×2 matrices.

Suppose we have a multiplexer M with m control variables, which we can denote as $c_1, c_2, \ldots, c_m$. We can define any of the possible input states to the multiplexer in terms of the values on the control variables: the **input state** i can be represented as a binary string, where each digit i_k is equal to the value of its corresponding control variable c_k. For example, if we have c_1, c_2, c_3 which have values of $0, 1, 1$ respectively, i would be $011 = 3$. Note that there are only 2^m possible input states since we are assuming that the control variables are always in basis quantum states. However, we can be dealing with superposition states in the target qubit.

For the multiplexer M, let F_i be the arbitrary single-qubit quantum function that will act on the target variable if and only if the input state is i: we can say that the function F_i is a **target (controlled) function** that is **controlled by** i. For example, for the function $F_3 = F_{011}$ controlled by the three variables c_1, c_2, c_3, we say that F_3 is controlled by $i = 3 = 011$; this is equivalent to saying that F_3 is controlled by the Boolean product $\bar{c_1}c_2c_3$ since only the input state $i = 3$ satisfies this control function. It is clear that any (binary) multiplexer with m control variables can be represented as 2^m controlled functions, one for each input state. Thus, we can uniquely represent the multiplexer as an ordered set:

$$M = \{F_0, F_1, \ldots, F_{2^m-1}\}_C$$

where C is the ordered set of control variables. We choose to represent the multiplexer in this fashion for mathematical convenience in the âĂIJProofâĂİ section below.

Note that a multiplexer of this form is a direct realization of a Minterm Sum of

Products form: each minterm of the control variables controls its own target function. For example, for a multiplexer with two control variables c_1, c_2, the minterms of the controls $\bar{c_1}\bar{c_2}, \bar{c_1}c_2, c_1\bar{c_2}, c_1c_2$ each appear once and each control a separate function F_0, F_1, F_2, F_3, respectively (once again, see Figure 1). Henceforth, this form of multiplexer that mirrors the Minterm Sum of Products form will be referred to as the **standard form**.

2.4 Multivalued quantum multiplexers

A multivalued quantum multiplexer is a gate that will, based on a set of q-valued control variables, use a classical q-valued function to select an arbitrary binary target function that acts on a set of target variables. These target functions can be any quantum functions, acting on any number of target variables. Thus, they are represented by unitary matrices, and the entire quantum multiplexer can be represented by a âĂIJunitary block matrixâĂİ. Figure 2 gives a general ternary multiplexer.

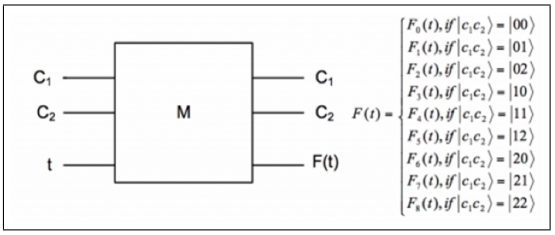

Figure 2: An arbitrary ternary standard form quantum multiplexer with two ternary-valued controls.

Suppose we have a multiplexer M with m q-valued control variables, which we will denote $c_1, c_2, \ldots, c_m$. We can represent an input state as a base q string of numbers, where each digit i_k is equal to the value of its corresponding control variable c_k. Note that there are only q^m possible inputs (input states) since we are assuming that the control variables are always in basis states. However, we will be dealing with superposition states of the target variables.

For the multiplexer M, let F_i be the unitary matrix that will act on the target variables if the input is i. Also, let C be the set of control variables. Then we can uniquely represent the multiplexer as an ordered set:

$$M = \{F_0, F_1, \ldots, F_{q^m-1}\}_C$$

In this paper, the multivalued multiplexers that we predominately deal with will have ternary-valued control variables and binary-valued target functions.

3 Fixed polarity quantum forms (FPQF) and Kronecker quantum forms (KQF)

3.1 Introduction to multivalued FPQF and KQF

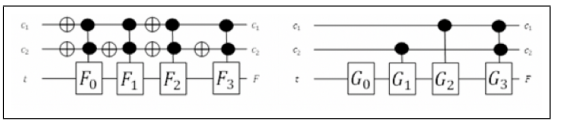

Figure 3: Circuit diagrams for binary multiplexers. The left diagram shows a standard form multiplexer; the right diagram shows a FPQF multiplexer in polarity 11.

One may be familiar with the Fixed Polarity Reed Muller form (FPRM), a polarized, canonical form of a binary Boolean expression where each variable appears solely in either its complemented (negative polarity) or uncomplemented (positive polarity) form. For a function on m variables, there are 2^m different FPRM forms since each of the variables can either be complemented or uncomplemented (there are 2 choices for each of the m variables). FPRM is related to Positive Davio and Negative Davio expansions. There is also the Kronecker Reed Muller form (KRM), a canonical form of a binary Boolean expression where each variable appears in either positive, negative, or mixed polarity. There are 3 choices (positive, negative, or mixed) for each variable, so there are 3^m different KRMs for a function on m variables. FPRM is a subset of KRM since KRM is related to Positive Davio and Negative Davio expansions, along with the Shannon expansion. If one is familiar with these Boolean forms, it may be easier to understand our new forms, but knowledge of FPRM and KRM is not critical to understanding the new concepts that we introduce.

For quantum multiplexers, given a set of control variables, $c_1, c_2, \ldots, c_m$, we can introduce the concept of polarity in a similar fashion as it appears in FPRM or KRM. We can do so by creating a multiplexer where each control variable is in a fixed polarity (either positive or negative polarity); such a multiplexer will henceforth be referred to as a **Fixed Polarity Quantum Form (FPQF)**. Similarly, we can create a multiplexer where control variables are in either fixed or mixed polarity; such a multiplexer will be referred to as a **Kronecker Quantum Form (KQF)**. When a multivalued variable is in fixed polarity, the control that is activated by a check against $|0\rangle$ becomes implicit and is no longer present; for example, for a ternary control variable in fixed polarity, controls will only be placed to check for the values of $|1\rangle$ and $|2\rangle$, but not $|0\rangle$ ($|0\rangle$ will be implicit, and any target functions that originally relied on a check against $|0\rangle$ will instead disregard that control variable completely). Figure 3 contrasts the binary-valued standard form with an FPQF form. If we wish to create an FPQF or KQF multiplexer that realizes exactly the

same functions as a particular standard form multiplexer for all input states while fixing the polarities of some (or all) of the control variables, it is clear that the functions $[G_0, G_1, \ldots, G_{q^m-1}]$ of this FPQF or KQF multiplexer will be different from the functions $[F_0, F_1, \ldots, F_{q^m-1}]$ of the standard form multiplexer since the control functions have changed: see Figure 5 for an example of the way that the control functions change. In an FPQF multiplexer, it is possible for multiple target functions to be active for a given input state; for example, for the input state 10 to a binary multiplexer, both G_0 and G_2 are active.

Henceforth, we will denote target functions as F_i if they are the target functions in a standard form multiplexer, and we will denote them as G_i if they are the target functions of an FPQF/KQF multiplexer. First, it is useful to formally define how the target functions of a FPQF multiplexer G_i operate.

In an FPQF multiplexer, G_i is a unitary matrix acting on the target variable and controlled by all input states j such that $j_k = i_k$ if $i_k \neq 0$ (where j_k and i_k are the kth digits of j and i, respectively). That is, G_i is only controlled by the control variable c_k if the kth digit of i is not 0. We have previously mentioned that multiple target functions may be active for a particular input state; conversely, a target function can be activated by multiple input states. For example, on a binary-valued multiplexer with two controls, $G_1 = G_{01}$ is controlled by both the input states 01 and 11. That is, G_1 is activated if the input state is 01 *or* 11. Similar to our notation for a standard form multiplexer, we denote an FPQF multiplexer as $M = \{G_0, G_1, \ldots, G_{q^m-1}\}_{C,p}$ where p is the polarity of the FPQF multiplexer as represented by a base q string. For q-valued logic, there are q^m possible FPQF forms for a given set of functions F_i, so we need a fast way to compute all possible polarities of $[G_0, G_1, \ldots, G_{q^m-1}]$ from $[F_0, F_1, \ldots, F_{q^m-1}]$. Note that the circuit structures for all FPQF polarities are highly similar; while Figure 4 shows an FPQF multiplexer in polarity 22, other polarities can be realized simply by applying a cyclic inverter to the beginnings of control lines that we would like to place in an alternative polarity, then modifying the target functions G_i.

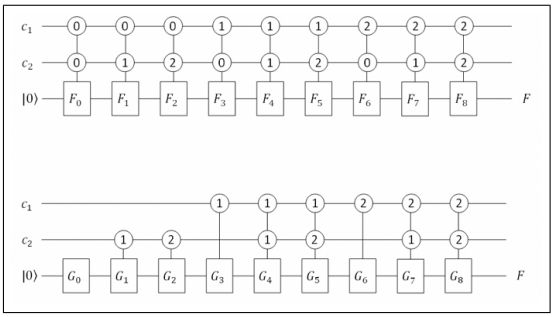

Figure 4: Circuit diagrams for ternary multiplexers. The upper diagram shows a standard form multiplexer; the lower diagram shows a FPQF multiplexer in polarity 22. (polarity 22 means that both variables are in polarity 2)

In a KQF multiplexer, whether or not a gate G_i is active is dependent on the polarity. If the kth variable is in a fixed polarity, G_i is controlled by all input states j such that $j_k = i_k$ if $i_k \neq 0$. That is, unless i_k is 0, G_i is controlled by the control variable c_k. If the kth variable is in a mixed polarity, G_i is controlled by all input states j such that $j_k = i_k$. Alternate fixed polarities can be realized simply by applying a cyclic inverter to the beginnings of controls lines that we would like to place in a different polarity, then modifying the target functions G_i accordingly. Figure 4 gives two example circuit diagrams for ternary multiplexers. The notation for controls is slightly different for ternary multiplexers. The control circles contain numbers. If the signal of the line is equal to the contained number when it reaches the control, then the control is enabled; otherwise, it is not.

There is a clear mathematical relationship between the target functions F_i of the standard form multiplexer and the target functions G_i of the FPQF multiplexer. Below we provide an example of how this relationship can be calculated.

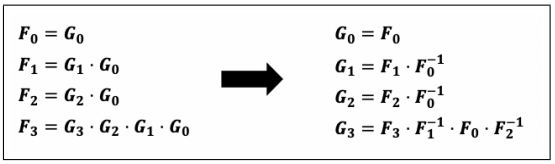

Figure 5: Mathematical relationships between the target functions of a binary standard form multiplexer F_i and the target functions of an FPQF multiplexer G_i in polarity 11. This example case involves multiplexers with two control variables (and thus four target functions).

Remember that a gate F_i or G_i is active if all its controls are enabled. Consider an example with two binary control variables, where we want the standard form and FPQF multiplexers to realize exactly the same target functions for all different control inputs. For input state 00, F_0 is active in the standard form multiplexer and G_0 is active in the FPQF multiplexer. Thus, $F_0 = G_0$. For input state 01, F_1 is active in the standard form multiplexer, but two gates are active in the FPQF multiplexer: G_0, G_1 (Figure 3 may help to visualize this). Thus, $F_1 = G_1 \cdot G_0$, and therefore $G_1 = F_1 \cdot {G_0}^{-1} = F_1 \cdot {F_0}^{-1}$. Note that the $\cdot$ operation is the matrix multiplication operation applied on the target functions when represented as unitary matrices. Similar observations can be made for input states 10 and 11; see Figure 5 for all the mathematical formulas that we derive. We have shown that the functions G_i can be directly calculated from the functions F_i for an example with two controls. Next, we demonstrate how to compose a transformation that can perform these calculations, which will allow us to generalize the process of calculating G_i from F_i.

For a given multiplexer $M = \{F_0, F_1, \ldots, F_{q^m-1}\}_C$ and a given KQF form $M = \{G_0, G_1, \ldots, G_{q^m-1}\}_{C,p}$, define $T_{m,p}$ as the transformation over U, the space of valid $m \times m$ unitary matrices, that takes the vector $[F_0, F_1, \ldots, F_{q^m-1}]$ to the vector

$[G_0, G_1, \ldots, G_{q^m-1}]$ for a certain polarity p (remember that polarities of a multiplexer are represented as base q strings, where the i^{th} digit denotes the polarity for the i^{th} control variable, in the same way as we previously introduced for FPRM polarities in [29]). We now provide a formal explanation for decomposing $T_{m,p}$ into a series of simpler transformations, which we can then represent with KQF butterfly diagrams similar to how it is used in classical KRM butterflies [5, 10].

3.2 The Kronecker product

Figure 6 illustrates the concept of the Kronecker product. The Kronecker product can be applied to any two matrices, each of arbitrary dimension.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \otimes \begin{bmatrix} x & y \\ z & w \end{bmatrix} = \begin{bmatrix} ax & ay & bx & by \\ az & aw & bz & bw \\ cx & cy & dx & dy \\ cz & cw & dz & dw \end{bmatrix}$$

Figure 6: Example of Kronecker product ($\otimes$) on two 2×2 matrices

3.3 Formal explanation of multivalued butterfly decomposition

This explanation demonstrates that we are able to decompose the transformation that takes a standard form multiplexer to a KQF multiplexer into a series of butterfly diagrams.

We first discuss a notation used in the following proof. Let us define an arbitrary transformation B, which acts on a subset of all the functions of a multiplexer. We define the butterfly transformation $B_{n,i}$ as a transformation that acts on a set of q^n functions represented as unitary matrices, $[F_0, F_1, \ldots, F_{q^n-1}]$, such that every set of functions $F_{r0}, F_{r1}, \ldots$, where the base q representations of $r0, r1, \ldots$ differ only at the ith digit, is acted upon by the transformation B. For example, if B takes $[a, b]$ to $[b, a \cdot b]$ for a **binary** case, then $B_{2,1}$ is the transformation performed on $2^2 = 4$ functions, where B is acted upon pairs of functions whose binary representations of index only differ at the 1^{st} digit. The transformation $B_{2,1}$ would take the functions $[F_0, F_1, F_2, F_3]$ to $[F_1, F_0 \cdot F_1, F_3, F_2 \cdot F_3]$. This is because B was applied to the pairs $[F_0, F_1] = [F_{00}, F_{01}]$ and $[F_2, F_3] = [F_{10}, F_{11}]$ since the indices of each pair of functions differ only at the 1^{st} digit. As an additional example, if B takes $[a, b, c]$ to $[b, c \cdot b^{-1}, a \cdot b^{-1}]$ for a ternary case, then $B_{2,1}$ is the transformation performed on $3^2 = 9$ functions, where B is acted upon the triplets of functions whose ternary representations of index only differ at the 1^{st} digit. The transformation $B_{2,1}$ would take the functions $[F_0, F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8]$ to $[F_1, F_2 \cdot {F_1}^{-1}, F_0 \cdot {F_1}^{-1}, F_4, F_5 \cdot {F_4}^{-1}, F_3 \cdot {F_4}^{-1}, F_7, F_8 \cdot {F_7}^{-1}, F_6 \cdot {F_7}^{-1}]$ since B is

applied to the triplets $[F_0, F_1, F_2]$, $[F_3, F_4, F_5]$, and $[F_6, F_7, F_8]$. See Figure 9 for an illustration of this particular ternary transformation. Note that $B_{n,i}$ is the algebraic representation of a column of butterflies on q^n objects where each butterfly kernel is stretched by q^i; thus, we call $B_{n,i}$ a butterfly transformation. A more in-depth explanation of FPRM and KRM butterflies (which are useful, but not essential, to understand our FPQF and KQF butterflies) can be found in [29]. We now show how a KQF transformation for q-valued controls can be decomposed to butterfly transformations: $A_0, A_1, \ldots, A_{q-1}$.

First, we define the transformations A_p, for $0 \leq p \leq q-1$. We define A_p as a degree q transformation that takes $[a_0, a_1, \ldots, a_{q-1}]$ to $[b_0, b_1, \ldots, b_{q-1}]$ such that $b_0 = a_{p+1}$ and $b_k = a_{k+p+1} \cdot {a_{p+1}}^{-1}$ for $k \neq 0$. For example, for ternary (degree $q = 3$), A_0 would take $[a_0, a_1, a_2]$ to $[b_0, b_1, b_2] = [a_1, a_2 \cdot {a_1}^{-1}, a_0 \cdot {a_1}^{-1}]$, A_1 would take $[a_0, a_1, a_2]$ to $[b_0, b_1, b_2] = [a_2, a_0 \cdot {a_2}^{-1}, a_1 \cdot {a_2}^{-1}]$, and A_2 would take $[a_0, a_1, a_2]$ to $[b_0, b_1, b_2] = [a_0, a_1 \cdot {a_0}^{-1}, a_2 \cdot {a_0}^{-1}]$. We claim that $T_{m,p} = A_{p_m} \otimes A_{p_{m-1}} \otimes \ldots \otimes A_{p_1}$. We prove this by induction on m.

3.3.1 Base Case: We first establish the claim for $m = 1$

We start by calculating $T_{1,p}$. We have $M = \{F_0, F_1, \ldots, F_{q-1}\}_C = \{G_0, G_1, \ldots, G_{q-1}\}_{C,p}$. In order for this to be true, the multiplexers must output the same values for all inputs of the control variable. By definition, for the multiplexer $\{G_0, G_1, \ldots, G_{q-1}\}_{C,p}$, the function G_k for $k \neq 0$ is selected by k and the polarity shifts the control variable by $q-1-p$, so G_k is selected when the control variable is $(k-(q-1-p)) mod\ q = (k+p+1-q) mod q = k+p+1$. Thus, we have $G_0 = F_{p+1}$. For $k \neq 0$, we have $G_k \cdot G_0 = F_{k+p+1}$, so $G_k = F_{k+p+1} \cdot {G_0}^{-1} = F_{k+p+1} \cdot {F_{p+1}}^{-1}$. Thus, the transformation A_p over U takes $[F_0, F_1, \ldots, F_{q-1}]$ to $[G_0, G_1, \ldots, G_{q-1}]$. Thus, we have $T_{1,p} = A_p$ as desired.

3.3.2 Induction: We establish the claim for m, assuming it holds for $m-1$.

For the induction step it suffices to show that $T_{m,p} = T_{m-1,p'} \otimes T_{1,p_1}$, where $p' = p - p_1 q^{m-1}$ (p' is the polarity p without its leading digit, p_1).

Given a multiplexer $M = \{F_0, F_1, \ldots, F_{q^m-1}\}_C$ with control variables $c_1, c_2, \ldots, c_m$ and t target variables, it can be reinterpreted as a multiplexer with one control variable, c_1, and $t+m-1$ target variables. If we let C' be the set of control variables without c_1, then $M = \{M_0, M_1, \ldots, M_{q-1}\}_{c_1}$, where each of M_k is a multiplexer of $m-1$ control variables that can be expressed as $M_k = \{F_{kq^{m-1}}, F_{kq^{m-1}+1}, \ldots, F_{(k+1)q^{m-1}-1}\}'_C$. See Figure 7 for an illustration of this.

Now, in order to show that $T_{m,p} = T_{m-1,p'} \otimes T_{1,p_1}$, we will realize M in two steps. First, we implement M as a multiplexer with one control variable in KQF form. That is:

$$M = \{M_0', M_1', \ldots, M_{q-1}'\}_{c_1,p_1}$$

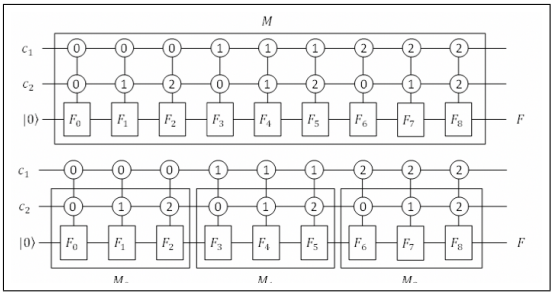

Figure 7: Example illustration of the reinterpretation of a multiplexer as a series of smaller multiplexers.

We will then show that each of M_k' is a multiplexer, so we will decompose each of the M_k' into KQF form with polarity p'.

First, we show that each of M_k' is a multiplexer. Note that the composition of two multiplexers is another multiplexer with target functions that are the composition of the original multiplexersâĂŹ target functions. And the inverse of a multiplexer is another multiplexer whose target functions are the inverses of the original multiplexerâĂŹs target functions. Thus, since the transformation T_{1,p_1} takes $[M_0, M_1, \ldots, M_{q-1}]$ to $[M_0', M_1', \ldots, M_{q-1}']$ each of the functions M_k' is a multiplexer. Now, let $M_k' = \{F_{kq^{m-1}}', F_{kq^{m-1}+1}', \ldots, F_{(k+1)q^{m-1}-1}'\}_{C'}$ Then, we can easily compute $[F_0', F_1', \ldots, F_{q^m-1}']$ from $[F_0, F_1, \ldots, F_{q^m-1}]$. Once again, because of the way multiplexers compose, the transformation T_{1,p_1} also takes $[F_k, F_{k+q^{m-1}}, F_{k+2q^{m-1}}, \ldots, F_{k+(q-1)q^{m-1}}]$ to $[F_k', F_{k+q^{m-1}}', F_{k+2q^{m-1}}', \ldots, F_{k+(q-1)q^{m-1}}']$ for all $0 \leq k \leq q^{m-1} - 1$.

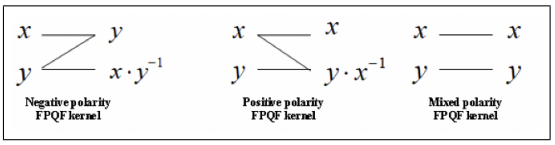

Figure 8: Butterfly kernels for binary FPQF and KQF butterfly diagrams. Only the first two types of kernels are present in FPQF diagrams.

Finally, we can realize $M = \{G_0, G_1, \ldots, G_{q^m-1}\}_{C,p}$ from $M = \{M_0', M_1', \ldots, M_{q-1}'\}_{c_1,p_1}$ by representing each of the multiplexers M_k' in KQF form. Thus, we have:

$$M_k' = \{F_{kq^{m-1}}', F_{kq^{m-1}+1}', \ldots, F_{(k+1)q^{m-1}-1}'\}_{C'}$$

$$= \{G_{kq^{m-1}}, G_{kq^{m-1}+1}, \ldots, G_{(k+1)q^{m-1}-1}\}_{C',p'}$$

Thus, the transformation $T_{m-1,p'}$ takes $[F'_{kq^{m-1}}, F'_{kq^{m-1}+1}, \ldots, F'_{(k+1)q^{m-1}-1}]$ to $[G_{kq^{m-1}}, G_{kq^{m-1}+1}, \ldots, G_{(k+1)q^{m-1}-1}]$. Thus, by the definition of the Kronecker product, $T_{m-1,p'} \otimes T_{1,p_1}$ takes $[F_0, F_1, \ldots, F_{q^m-1}]$ to $[G_0, G_1, \ldots, G_{q^m-1}]$ and we have $T_{m,p} = T_{m-1,p'} \otimes T_{1,p_1}$ as desired.

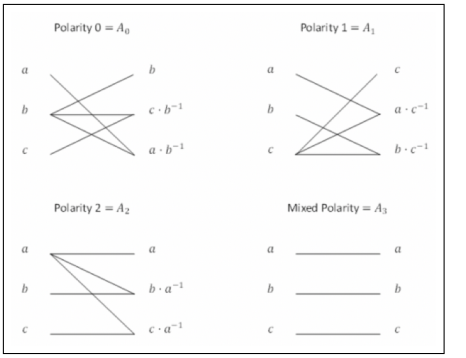

Figure 9: Butterfly kernels for ternary FPQF and KQF butterfly diagrams. Only the first three types of kernels are present in FPQF diagrams.

4 Discussion of the program used to compute FPQF and KQFs

4.1 FPQF and KQF butterflies

Here, we introduce the FPQF and KQF butterfly kernels that are used in FPQF and KQF butterflies. These are illustrated in Figure 8 for binary, and Figure 9 for ternary.

They are based on the previously given explanation for composition. In Figure 9, the four butterflies correspond to A_0, A_1, and A_2, along with a special case A_3 for mixed polarity in KQF, which is a transformation that has no effect on the target functions. The appearance of the binary kernels in Figure 8 may seem arbitrary, but they are in fact based on the appearance of butterfly kernels for FPRM butterfly diagrams and other Fast Fourier Transform butterflies.

The construction of FPQF butterfly diagrams is similar to the construction of FPRM butterfly diagrams, as discussed in [29]. However, note that the inputs to the diagram are no longer the minterms of a Boolean function, and the outputs are no longer spectral coefficients; rather, the inputs are the target functions F_i of the standard form multiplexer, and the outputs are the target functions G_i of the polarized multiplexer. See Figure 10 for an example of the FPQF butterfly diagrams for all polarities of FPQF on a binary multiplexer with three controls . Figure 11 gives an example FPQF butterfly diagram for a single polarity of a ternary multiplexer with two controls , and Figure 12 gives further examples. Figure 12 also demonstrates that the target functions have no effect on the shape of the butterfly kernels; only the polarity has an effect.

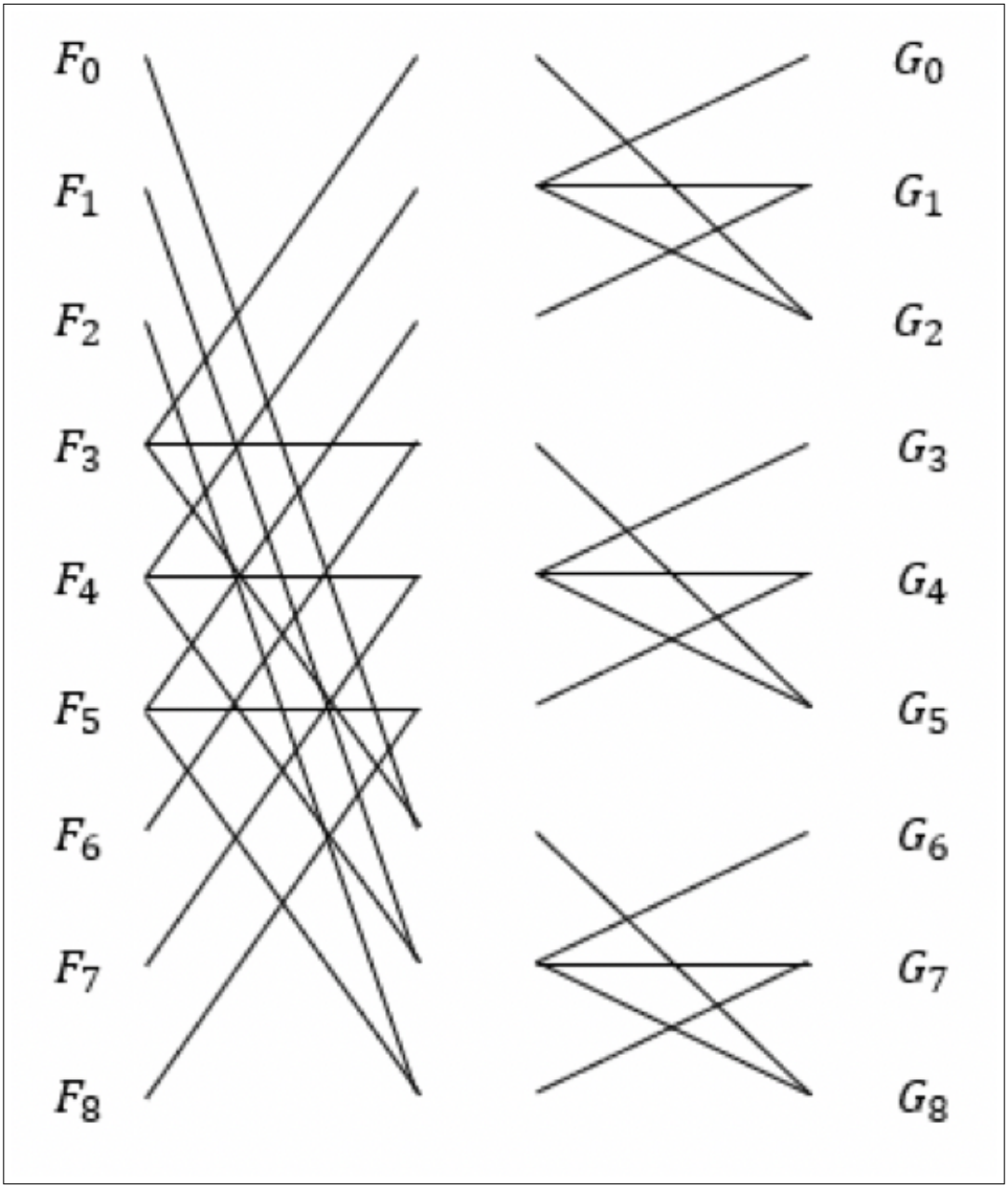

Figure 11: Sample KQF butterfly diagram for a standard form multiplexer with two ternary control variables. This butterfly converts a ternary standard form multiplexer into a KQF form with polarity 00.

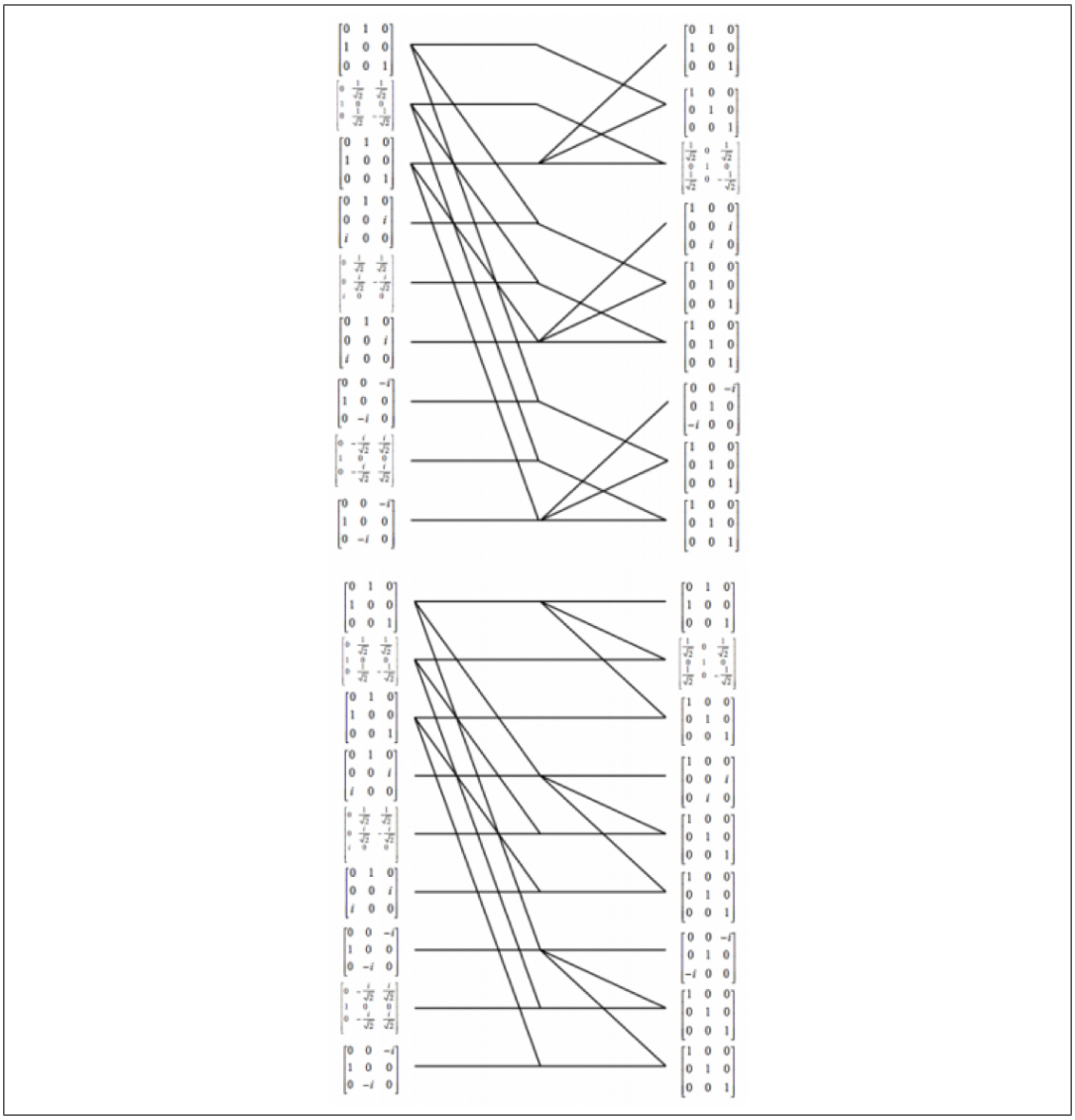

Figure 12: Two arbitrary examples of butterfly diagrams. These diagrams are for multiplexers with ternary-input, ternary-output; in contrast, our program functions for ternary-input, binary-output.

4.2 Quantum cost of FPQF/KQF polarities

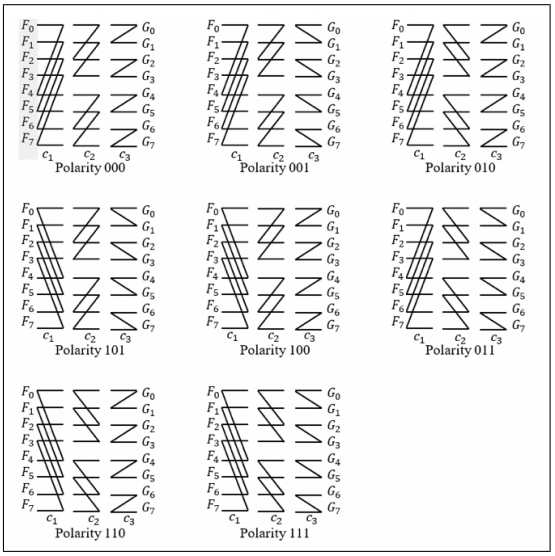

Figure 10: All polarities of FPQF butterflies for a standard form binary multiplexer with three control variables. Note that the sequences of functions G_i generated by each FPQF butterfly are different.

Having developed a method to calculate all possible FPQF and KQF forms of a multivalued standard form multiplexer (that is, by constructing and evaluating all possible FPQF or KQF butterfly diagrams), we now introduce how the cost of quantum multiplexers will be calculated for our ternary multiplexer program. Similar to [8], we define the cost of a quantum circuit in terms of the number of uncontrolled and single-controlled gates required to realize it; we can find this by summing the costs of all the controlled gates in the multiplexer circuit. The concept is being discussed with regards to ternary-valued multiplexers; however, formulations for the costs of multi-controlled ternary gates do not exist yet. Thus, we approximate cost by using currently existing costs for multi-controlled binary gates. Maslov et.al. [7] has previously established the following cost functions for binary Toffoli gates with m controls (see Table 1). We will assume that we are able to use a single Ancilla bit for our entire circuit, so the equation $32m - 96$ is relevant to us. For reasons that are not important to the understanding of this paper (and which are discussed in [8]), it is therefore theoretically possible to create any multi-controlled gate through the same process as the construction of a Toffoli: see *Lemma 6.1* in [8]. Thus, the cost functions that have been developed by [8] for Toffoli gates can also be applied to formulate the cost of multi-controlled binary gates of any type.

For approximation, we will assume that the cost of multi-controlled ternary gates with n controls is directly proportional to the cost of multi-controlled binary gates with n controls. Additionally, we assume that the cost of cyclic inverters is similar to the cost of a binary inverter; in other words, we assume that$(+1)$ and $(+2)$ gates cost roughly as much as a NOT gate. We base this assumption on results from previous work by Wang et. al. [28], where multi-controlled Toffoli-like gates were realized

Multi-controlled gate size $(m+1)$	Number of Ancilla bits	Gate Cost
1	0	1
2	0	1
3	0	5
4	0	13
5	0	29
6	1	52
7	1	84
8	1	116
9	1	154
10	1	192
$(m+1) > 10$	1	$32m - 96$

Table 1: Costs of multi-controlled gates with m controls. These gates use one or fewer Ancilla bits, which can be reused. Reproduced from [8]

for ternary logic with costs that were proportional to multi-controlled binary Toffoli gates.

4.3 Algorithm of the program

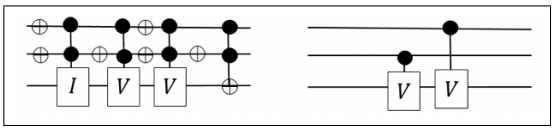

Figure 13: A standard form binary multiplexer (left) and its FPQF equivalent in polarity 11 (right). Note the drastic simplification of the circuit.

A program was written to automate the process of generating KQF butterflies and calculating costs of the resulting circuits. The program is written in Java because of its strong Object Oriented Programming (OOP) support; the code uses many classes (such as Multiplexer, Function, and more), and thus, JavaâĂŹs strong support of OOP is a large benefit. When executed, the program asks for the target functions of the standard form multiplexer in the form of unitary matrices as input (which are inputted in ascending natural order based off the index of the controls), and outputs the least expensive polarity of FPQF.

Alternatively, the program can also parse the names of common functions (e.g. âĂIJPXâĂİ for Pauli X, or âĂIJHâĂİ for Hadamard) to unitary matrices. The program uses a Depth First Search algorithm to generate all possible KQF forms by recursively applying layers of butterflies; that is, for each layer of butterflies,

the program first applies a layer of polarity 0 butterflies to the multiplexer and recursively calls itself; after the call returns, the program backtracks by one layer and applies a layer of polarity 1 butterflies to the multiplexer, then recursively calls itself, etc. Butterflies are implemented as a series of matrix transformations, as seen in Figure 11. The polarity 0 butterfly receives three unitary matrices, a, b, and c. It then outputs c as the first output, $a \cdot c^{-1}$ as the second output, and $b \cdot c^{-1}$ as the third output. These outputs can be encoded as the combinations of matrix inversions and matrix multiplications. The other butterfly polarities can be coded similarly. Once all KQF forms are generated, cost can be calculated as previously discussed: for each controlled function, determine the number of controls n needed for that function. If $n < 10$, the program references the costs derived by [8]. Otherwise, the code uses the equation $32m - 96$ to determine the cost of the gate.

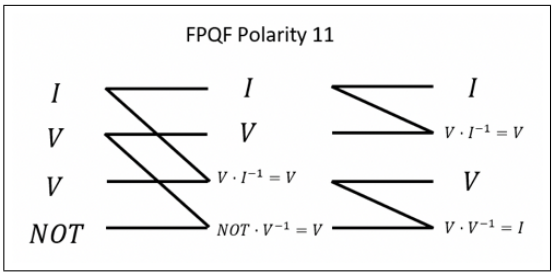

Figure 14: The butterfly diagram used to convert the standard form multiplexer to FPQF polarity 11 in Figure 13.

By finding the costs of all KQF polarities, the program can exhaustively determine which polarities are the cheapest, thus finding the exact minimum cost for an FPQF or KQF realization of a multiplexer.

5 Analysis and results

An additional program was written (also in Java) to generate large random test cases; initially, the test multiplexers were created by randomly pulling target functions from a set of common quantum functions: Pauli Rotations, Hadamard, NOT, V, V Hermitian, and Identity. A set of well-known quantum circuit cases were also created manually, such as a binary case that involves Identity, V, V, and NOT, as seen in Figure 13. The corresponding butterfly diagram for the case in Figure 13 can be found in Figure 14. Another example is provided in Figure 15 of a standard form ternary multiplexer and its drastically cheaper FPQF counterpart. On these well-

Number of Controls	Original Cost	Best Polarity	Worst Polarity	Average Polarity	Best Cost Reduction
3	322	148	322	210	54%
4	2100	1007	2100	1276	52%
5	11713	5248	11713	6727	55%
6	52785	24722	52785	31798	53%
7	219258	108331	219258	138359	51%
8	874045	442109	874045	562295	49%
8	869705	442573	869705	562481	49%
8	870170	442521	870170	562351	49%

Table 2: Data on ternary cases generated with Paulis, Hadamards, NOTs, V/V Hermitians, and Identities. The program generated all KQF polarities.

known cases, the program correctly generated all polarities, including the optimal solutions.

5.1 Randomly generated cases: Pauli X/Y/Z, Hadamard and H Hermitian, NOT, V and V Hermitian, Identity

Cases with more variables take exponentially longer time to run, so it became impractical to run the program on test cases with more than 9 control variables. In the best case, the KQF forms that were generated had costs as little as 50% of the cost of the standard form multiplexer, and the cost difference between the average polarity and the optimal polarity was significant: see Table 2 for results on several examples. Note that the best case cost reduction decreases slightly as the size of the multiplexer increases, which suggests that the method becomes less effective as multiplexer size increases.

It is interesting to note that the optimal polarities (not shown in the tables) were all FPQF forms; Shannon expansions were not a part of the optimal solution for any of the test cases. This suggests that for this type of randomly generated data, having a Shannon expansion does not improve the cost of the result.

5.2 Randomly generated cases: NOT, V and V Hermitian

This method was also tested on quantum multiplexers randomly generated from a much smaller pool of target gates: NOT, V, and V Hermitian only. We are interested in using this set of gates as targets because there are several algorithms to synthesize quantum reversible circuits with this set of gates [8]. In addition, this

Number of Controls	Original Cost	Best Polarity	Worst Polarity	Average Polarity	Best Cost Reduction
3	378	133	378	180	65%
4	2430	737	2430	1029	70%
5	12879	3710	12879	5210	71%
6	61965	17260	61965	24438	72%
7	255879	77828	255879	105731	70%
8	1016955	324077	1016955	428118	68%
8	1016955	325687	1016955	427870	68%
8	1016955	326620	1016955	427705	68%
9	3798819	1277506	3798819	1649713	66%
9	3798819	1280032	3798819	1649279	66%

Table 3: Data on ternary cases generated with NOTs, Vs, and V Hermitians. The program generated all KQF polarities.

set of gates is very conducive for canceling to Identity, so we expect best case cost reduction to be even greater for this set of data. It can be seen in Table 3 that the cost reduction is significantly greater (at 60-70%), and as before, the significant cost difference between the best case and average case costs means that it is meaningful to search for the best polarity instead of only using a random one. Once again, the effectiveness of the method decreased as size increased. Note once more that randomly generated functions are the most difficult cases.

Once more, the KQF method did not offer improvements on the best case cost; all of the optimal butterflies were FPQF forms.

6 Conclusion

In this paper, we first define standard form multiplexers and two new types of multiplexer forms: FPQF and KQF forms. Next, we provide a method to convert a standard form multiplexer to an FPQF or KQF form using a polarity transformation, and we give a proof for decomposing these transformations into smaller transformations that can be represented with butterfly diagrams in an analogous way to well-known FPRM and KRM butterfly diagrams. Note that unlike work on ESOP minimization [6, 11, 12, 19], our work is an extension of Reed Muller to quantum circuits where the target gate can be any arbitrary target function instead of only NOT, like in classical Reed Muller. Then, we test this method for randomly generated standard form ternary multiplexers that use target functions from a small

pool of common quantum functions, finding that cost reductions are as high as 70%, but decrease as size (and equivalently, complexity) of the standard multiplexer increases. We also find that there are significant cost differences between the best case polarity and the average case polarity, which justify the long runtime required to exhaustively apply Depth First Search to find the maximally optimized KQF form. Additionally, we discovered that on randomly generated data, KQF offers no benefit on best case cost reduction (the best case answers are always FPQF forms) while having a longer run time than a program that searched among FPQF forms only. This suggests that on randomly generated multiplexers, there is no benefit to using KQF; only FPQF should be used to optimize these types of multiplexers.

The FPQF and KQF methods can also be extended to any multivalued quantum multiplexers, not just ternary or binary: it is expected that the effectiveness and cost reductions of these methods will be very high for multivalued logics with $q > 3$ as well. However, the time costs of the current algorithm are exponentially great due to the use of complete search, which suggests low scalability potential for circuits with high variable counts. Furthermore, the steadily decreasing trend of effectiveness of this method on larger randomly generated multiplexers suggests that more research should be conducted to find alternate methods for optimizing larger standard form multiplexers since it appears that the FPQF method will not net many savings at extremely high sizes. This is consistent with findings in [11] and [12], which suggest that Reed Muller-based methods, such as FPRM, KRM, and even GRM [12], lead to non-minimal results when minimizing Boolean functions that contain minterms with high Hamming distances. Finally, all our methods can be extended to incompletely specified functions, generalizing the method from [16].

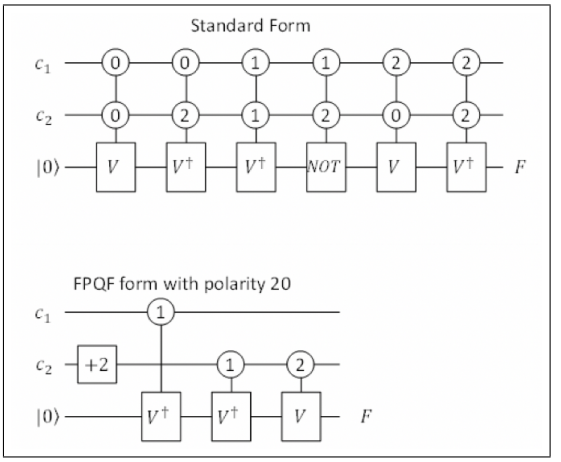

Figure 15: A standard form ternary multiplexer and its FPQF equivalent in polarity 20.

References

[1] R. R. Tucci, "A Rudimentary Quantum Compiler (2cnd Ed.)," arXiv, 2008.

[2] J. J. Vartiainen, M. MÃűttÃűnen and M. M. Salomaa, "Efficient decomposition of quantum gates," *Physical Review Letters*, vol. 92, no. 17, p. 177902, 2004.

[3] V. V. Shende, S. S. Bullock and I. L. Markov, "Synthesis of Quantum Logic Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, pp. 1000-1010, 2006.

[4] V. V. Shende, I. L. Markov and S. S. Bullock, "Minimal Universal Two-Qubit CNOT-based Circuits," *Physical Review A*, vol. 69, no. 6, p. 062321, 2004.

[5] M. Davio, J.-P. Deschamps and A. Thayse, Discrete and Switching Functions, St-Saphorin: Georgi Publishing Company, 1978.

[6] M. Perkowski and D. Shah, Design of Regular Reversible Quantum Circuits, Portland: Portland State University, 2010.

[7] D. Maslov and G. W. Dueck, "Improved Quantum Cost for n-bit Toffoli Gates," *Electronic Letters*, vol. 39, no. 25, pp. 1790-1791, 2004.

[8] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin and H. Weinfurter, "Elementary gates for quantum computation," *Physical Review A*, vol. 52, no. 5, pp. 3457-3467, 1995.

[9] S. R. Hutsell, "An Eigenanalysis and Synthesis of Unitary Operators used in Quantum Computing Algorithms," Portland State University, PHD dissertation, Portland, 2009.

[10] D. H. Green, "Families of Reed-Muller canonical forms," *International Journal of Electronics*, vol. 70, no. 2, pp. 259-280, 1991.

[11] A. Mishchenko and M. Perkowski, "Fast Heuristic Minimization of Exclusive-Sums-of-Products," *5th International Reed-Muller Workshop*, 2001.

[12] L. Csanky, M. A. Perkowski and I. Schafer, "Canonical restricted mixed-polarity exclusive-OR sums of products and the efficient algorithm for their minimisation," *IEE Proceedings E - Computers and Digital Techniques*, vol. 140, no. 1, pp. 69-77, 1993.

[13] C. Gidney, "Halving the cost of quantum addition," *Quantum*, vol. 2, p. 74, 2018.

[14] M. Soeken, M. Roetteler, N. Wiebe and G. D. Micheli, "Hierarchical Reversible Logic Synthesis Using LUTs," in *54th Annual Design Automation Conference*, Austin, 2017.

[15] G. Meuli, M. Soeken, M. Roetteler, N. Wiebe and G. D. Micheli, "A best-fit mapping algorithm to facilitate ESOP-decomposition in clifford+T quantum network synthesis," in *23rd Asia and South Pacific Design Automation Conference*, Jeju, 2018.

[16] D. Debnath and T. Sasao, "Exact Minimization of FPRMs for Incompletely Specified Functions by Using MTBDDs," *IEICE Transactions*, Vols. E88-A, no. 12, pp. 3332-3341, 2005.

[17] W. N. N. Hung, G. Yang, X. Song and J. Yang, "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*, pp. 1652-1663, 2006.

[18] H. Wu, M. A. Perkowski, X. Zeng and N. Zhuang, "Generalized partially-mixed-polarity Reed-Muller expansion and its fast computation," *IEEE Transactions on Computers*, vol. 45, no. 9, pp. 1084-1088, 1996.

[19] R. Drechsler, A. Finder and R. Wille, "Improving ESOP-Based Synthesis of Reversible Logic Using Evolutionary Algorithms," in *Applications of Evolutionary Computation*, Torino, 2011.

[20] E. Tsai and M. Perkowski, "Synthesis of Permutative Quantum Circuits with Toffoli and TISC Gates," in *The International Symposium on Multiple-Valued Logic*, Tuusula, 2012.

[21] F. S. Khan, Quantum Multiplexers, Parrondo Games, and Proper Quantization, Portland: Portland State University, 2009.

[22] F. S. Khan and M. Perkowski, "Synthesis of multi-qudit Hybrid and d-valued Quantum Logic Circuits by Decomposition," *Theoretical Computer Science*, vol. 367, no. 3, pp. 336-346, 2006.

[23] S. A. Bleiler and F. S. Khan, "Properly quantized history-dependent Parrondo games, Markov processes, and multiplexing circuits," *Physics Letters A*, vol. 375, no. 19, pp. 1930-1943, 2011.

[24] M. H. A. Khan, "Primitive quantum gate realizations of multiple-controlled Toffoli gates," in *16th Int'l Conf. Computer and Information Technology*, Khulna, 2014.

[25] Z. Li, M. Perkowski, X. Song and H.-W. Chen, "Realization of a new permutative gate library using controlled-kth-root-of-NOT quantum gates for exact minimization of quantum circuits," *International Journal of Quantum Information*, 2014.

[26] Z. Li, X. Song, S. Chen and M. Perkowski, "Quantum Circuit Synthesis using a New Quantum Logic Gate Library of NCV Quantum Gates," *International Journal of Theoretical Physics*, 2016.

[27] M. Lukac and M. Perkowski, "Quantum Behaviors: Synthesis and Measurement," in *RM 2007*, Oslo, 2007.

[28] Y. Wang and M. Perkowski, "Improved Complexity of Quantum Oracles for Ternary Grover Algorithm for Graph Coloring," in *IEEE 41st International Symposium on Multiple-Valued Logic*, Tuusula, 2011.

[29] K. Jin, T. Saffat and M. Perkowski, "A fixed polarity approach for optimization of general binary quantum multiplexers," in *International Workshop on Quantum Compilation*, San Diego, 2018.

Received 30 April 2019