# QMDD Minimization using Sifting for Variable Reordering

D. Michael Miller[1], David Y. Feinstein[2] and Mitchell A. Thornton[2]

[1]*Department of Computer Science, University of Victoria, Canada*
*E-mail: mmiller@cs.uvic.ca*
[2]*Department of Computer Science and Engineering, Southern Methodist University, USA*
*E-mail: dfeinste, mitch@engr.smu.edu*

This paper considers variable reordering for *quantum multiple-valued decision diagrams* (QMDDs) used to represent the matrices describing reversible/quantum gates and circuits. An efficient method for adjacent variable interchange is presented and this method is employed to implement a vertex reduction procedure for QMDDs using sifting. Experimental results are presented showing the effectiveness of the proposed technique.

*Keywords:* Quantum Multiple-valued Decision Diagrams, Quantum Logic, Reversible Logic, Sifting

## 1 INTRODUCTION

A reversible/quantum circuit is a cascade of reversible/quantum gates. The behaviour of each gate can be described as a matrix and the function performed by the circuit is described by the product of the individual gate matrices. However, the matrix for an $n$-line reversible/quantum gate/circuit in $r$-valued logic has dimension $r^n \times r^n$ so the computation using standard matrix techniques quickly becomes impractical.

The *quantum multiple-valued decision diagram* (QMDD) data structure presented in [10,11] was specifically designed to address this problem. A QMDD represents the matrix corresponding to a gate or circuit as a directed acyclic graph. Efficient methods for constructing QMDDs for individual gates and for performing matrix multiplication directly with QMDDs are presented in [10].

1

As is the case for other decision diagram representations such as the ordered binary decision diagram [2], the number of vertices in a QMDD depends on the variable ordering selected. This paper addresses variable ordering for QMDDs [9]. We present a method for adjacent variable interchange and describe a modification to the QMDD structure required to make this a local operation for all QMDDs. We then present a heuristic vertex reduction algorithm for QMDDs based on Rudell's binary decision diagram "sifting" technique [13].

Section 2 presents the basic concepts of binary and MVL reversible and quantum gates and circuits with particular emphasis on the matrix representation. Section 3 addresses adjacent variable interchange for QMDDs. Section 4 shows how sifting can be applied to QMDDs. Experimental results are presented in section 5 and the paper concludes with observations and suggestions for further research in Section 6.

## 2  PRELIMINARIES

### 2.1  Reversible logic and quantum circuits

We present the basic concepts of reversible and quantum circuits necessary for this paper. More extensive background is available in the literature (*e.g.* [5,12]).

*Definition 1.*  A gate/circuit is logically *reversible* if it maps each input pattern to a unique output pattern.

Binary reversible gates and circuits have garnered considerable interest due to Landauer's principle which states that the erasure of information dissipates energy. Bennett [1] showed that for a binary circuit to not consume energy, it must be composed of reversible gates. The concept of reversibility has been extended to MVL circuits [8]. Quantum logic gates and circuits are inherently both logically and physically reversible [12]. In general, the behaviour of reversible and quantum gates and circuits can be described by complex-valued matrices and are modeled as bijective functions.

Figure 1 shows a binary reversible circuit with 3 lines and 5 gates. The symbol ⊕ denotes the *NOT* operation. For each gate, the *NOT* operates on the *target* line if every *control* line (lines with a black circle) has the value 1. Otherwise the target line is unchanged. Control and unconnected lines pass through the gate unchanged. A gate with no controls is a *NOT* gate. One with a single control is termed a *controlled-NOT* and those with more than one control are *Toffoli* gates [12].

Multiple-valued reversible circuits have been considered in [8]. The structure illustrated in Fig. 1 is generalized so that the target line is operated on by a negation or cycle operation depending on the values of the control lines. The non-zero values indicated in the control line connections specify the nonzero
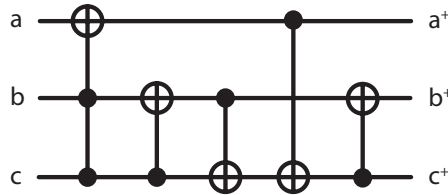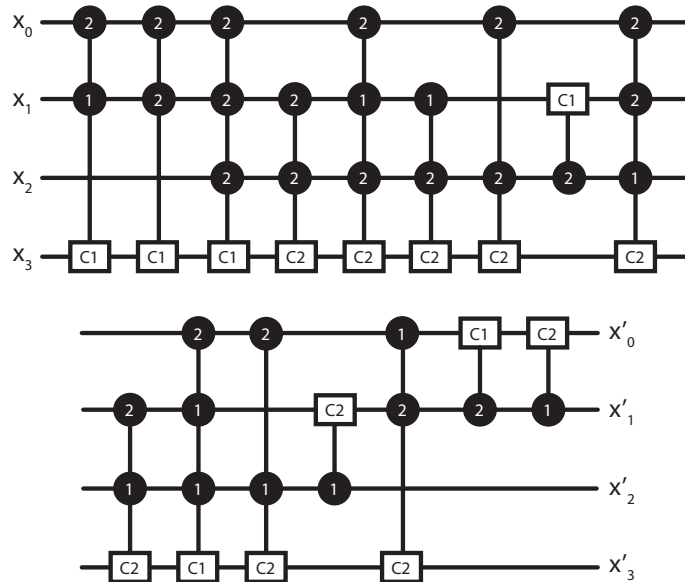
FIGURE 1
A binary reversible circuit



FIGURE 2
A ternary reversible full adder

value required in order to trigger the operation. Figure 2 shows a reversible circuit from [8] that operates as described and realizes a ternary full adder.

Quantum logic gates [12] operate in a similar fashion with the values on designated control lines determining if a particular quantum logic transformation is to be applied to the target line.

### 2.2 Matrix representation of reversible/quantum gates and circuits

The operations performed on the target line for the gates considered in this paper are given by the $r \times r$ matrices in Table 1. *NOT* is the normal binary complement shown as $\oplus$ in Fig. 1. $V$ and $V^+$ are quantum operations sometimes referred to as "square root of NOT" gates since $V \times V = V^+ \times V^+ = NOT$. Note that $V$ is unitary, as are all operation matrices for quantum gates, and

| Binary ($r = 2$) Matrices | | Ternary ($r = 3$) Matrices | |
|---|---|---|---|
| *NOT* | $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ | *NEG* | $\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$ |
| $V$ | $\begin{pmatrix} \frac{1+i}{2} & \frac{1-i}{2} \\ \frac{1-i}{2} & \frac{1+i}{2} \end{pmatrix}$ | *C1* | $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$ |
| $V^+$ | $\begin{pmatrix} \frac{1-i}{2} & \frac{1+i}{2} \\ \frac{1+i}{2} & \frac{1-i}{2} \end{pmatrix}$ | *C2* | $\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ |

TABLE 1
Gate Operation Matrices

| c | b | a | $c^+$ | $b^+$ | $a^+$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

TABLE 2
Representation of the Toffoli Gate $T(a; b, c)$ in a 3-line Circuit

$V^+$ is the conjugate transpose of $V$, hence $V^+ = V^{-1}$. *NEG* is ternary negation, and $C1$ and $C2$ are the two ternary unary cycle operations [8].

The matrices in Table 1 define the operation on the target line. The matrix definition for an $r$-valued gate in an $n$-line reversible or quantum circuit has dimension $r^n \times r^n$ taking into account the operation on the target line and the control and unconnected lines. For example, the leftmost gate in Fig. 1 has the truth table and matrix specifications shown in Table 2.

Table 3 shows the matrix specification for a $V$ type quantum gate with target line $a$ and control line $c$. Line $b$ is not connected for this gate.

A reversible/quantum circuit is a cascade (from input to output) of gates $g_0, g_1, g_2, \ldots$ Each gate $g_i$ has a matrix representation $M_i$ where the dimension depends on the radix and the number of lines in the circuit. The matrix defining the transformation performed by the overall circuit comprised of $k$ gates in a cascade is given by $M_{k-1} \times \cdots \times M_2 \times M_1 \times M_0$. The challenge is that the size

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \dfrac{1+i}{2} & \dfrac{1-i}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & \dfrac{1-i}{2} & \dfrac{1+i}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \dfrac{1+i}{2} & \dfrac{1-i}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & \dfrac{1-i}{2} & \dfrac{1+i}{2}
\end{bmatrix}
$$

TABLE 3
Representation of the Quantum Gate $V(a; c)$ in a Circuit with Lines $a, b, c$

of these matrices and the computation required for matrix multiplication by traditional techniques is prohibitive for all but a small number of circuit lines. For example, for a ternary circuit with 10 lines each matrix has dimension 59,049 by 59,049.

### 2.3  Quantum multiple-valued decision diagrams

The use of binary decision diagrams for the representation and manipulation of matrices was discussed in [3,4]. Quantum multiple-valued decision diagrams (QMDD) were introduced in [10,11] as a means to represent and manipulate the matrices required for reversible/quantum gates and circuits. Here we present a brief description of QMDDs and assume the reader is familiar with fundamental decision diagram techniques [16].

As noted above, a reversible/quantum circuit with $n$ lines has a transformation matrix of dimension $r^n \times r^n$ where $r$ is the radix. Such transformation matrices quickly explode in size. However, they do exhibit a great degree of regularity.

A matrix of dimension $r^n \times r^n$ can be partitioned as:

$$
M = \begin{bmatrix}
M_0 & M_1 & \cdots & M_{r-1} \\
M_r & M_{r+1} & \cdots & M_{2r-1} \\
\vdots & \vdots & \ddots & \vdots \\
M_{r^2-r} & M_{r^2-r+1} & \cdots & M_{r^2-1}
\end{bmatrix}
\tag{1}
$$

where each $M_i$ is a submatrix of dimension $r^{n-1} \times r^{n-1}$. Each of the $M_i$ can be similarly partitioned and the process repeated until scalars are reached. This repeated partitioning leads to the fundamental QMDD structure.

*Definition 2.* We shall refer to a matrix partition of the type shown in eqn. (1) as an $r^2$-*partitioning*.

*Definition 3.* A *quantum multiple-valued decision diagram* (QMDD) [10] is a directed acyclic graph with the following properties:

- There is a single *terminal* vertex annotated with value 1. The terminal vertex has no outgoing edges.

- There are some number of *non-terminal* vertices each labeled by an $r^2$-valued selection variable. Each non-terminal vertex has $r^2$ outgoing edges designated $e_0, e_1, \ldots, e_{r^2-1}$.

- One vertex is the *start* vertex and has a single incoming edge that itself has no source vertex.

- Every edge in the QMDD, including the one leading to the start vertex, has an associated complex-valued *weight*. An edge with weight of 0 must point to the terminal vertex. This is required to ensure uniqueness of the representation of each matrix.

- The selection variables are *ordered* (assume with no loss of generality the ordering $x_0 \prec x_1 \prec \cdots \prec x_{n-1}$) and the QMDD satisfies the following two rules:

  — Each selection variable appears at most once on each path from the start vertex to the terminal vertex.

  — An edge from a non-terminal vertex labeled $x_i$ points to a non-terminal vertex labeled $x_j$, $j < i$ or to the terminal vertex. Hence $x_0$ is closest to the terminal and $x_{n-1}$ labels the start vertex.

- No non-terminal vertex is *redundant*, *i.e.* no non-terminal vertex has its $r^2$ outgoing edges all with the same weights and pointing to a common vertex.

- Each non-terminal vertex is normalized (see details of the normalization process in the next subsection).

- Non-terminal vertices are *unique*, *i.e.* no two non-terminal vertices labeled by the same $x_i$ can have the same set of outgoing edges (destinations and weights).

## 2.4  Vertex normalization
The initial definition of QMDDs [10,11] used the following normalization rule:

*Definition 4.* A QMDD vertex is *normalized* if its outgoing edges are such that there is a $j$ such that the edge $e_j, 0 \le j \le r^2 - 1$, has weight 1 and $e_i, \forall i, 0 \le i < j$ has weight 0.

When this normalization rule is used, each vertex is normalized when it is constructed by finding the nonzero weight on the lowest index edge (one must exist or the vertex is redundant), dividing all edge weights by the weight identified, and attaching the identified weight to the edge leading to the vertex.

Given this definition of normalization, it is possible to show that the QMDD representation for any given matrix is unique.

**Theorem 1.** *An $r^n \times r^n$ complex valued matrix M has a unique (up to variable reordering or relabeling) QMDD representation.*

*Proof:* The proof is by induction on $n$.

$n = 0$: In this case, $M$ is a single element. The QMDD representation consists of the terminal vertex which is also the start vertex and no nonterminal vertices. The weight on the edge leading to the terminal (start) vertex is the value in $M$. This is clearly a unique representation.

$n > 0$: Assume the result holds for all $r^{n-1} \times r^{n-1}$ matrices. Consider thee $r^2$-partitioning of a matrix $M$ of dimension $r^n \times r^n$. Since each submatrix $M_i$ has dimension $r^{n-1} \times r^{n-1}$, by the inductive hypotheses the QMDD for each $M_i$ is unique. Let $sv$, which is labeled $x_{n-1}$, denote the start vertex for the QMDD representing $M$. Initially, equate each outgoing edge $e_k, 0 \leq k \leq r^n - 1$ from $sv$ to the edge pointing to the start vertex in the QMDD representation of $M_i$, and give the edge leading to $sv$ weight 1.

Normalizing $sv$ as defined above, begins by finding the smallest $k$ such that the weight $w_k$ on edge $e_k$ from $sv$ is nonzero. If $w_k = 1$, $sv$ is already normalized. If $w_k \neq 0$, divide all the nonzero weights on outgoing edges from $sv$ by $w_k$ and set the weight on the edge leading to $sv$ to $w_k$.

Since the QMDD for each $M_i$ is unique up to variable reordering or relabelling and the normalization process ensures $sv$ and its associated edge weights are unique, the QMDD for $M$ is unique up to variable reordering or relabelling.                                                                                    □

We have found that vertex normalization as given in Definition 4 does allow for adjacent variable interchange to be performed as an operation local to the variables being interchanged for QMDDs representing 0-1 matrices such as those representing reversible binary and multiple-valued circuits. However, it does not for QMDDs representing some matrices encountered for quantum circuits. The problem arises since variable reordering changes the order in which the elements of the matrix are considered and normalization as defined in Definition 4 can identify different normalization divisors depending on the variable order.

The solution is to use the following alternative normalization definition:

*Definition 5.* A QMDD vertex is *normalized* if its outgoing edges are such that the largest weight on any edge out of the vertex is 1.

Note there is no natural ordering for complex numbers. In this work we consider the complex number $r_a e^{i\theta_a}$ to be greater than $r_b e^{i\theta_b}$ if $r_a > r_b$ or in the case when $r_a = r_b$, $\theta_a < \theta_b$.

When Definition 5 is used, the process for normalizing a vertex is as described above except the maximum weight on the edges from the vertex is used as the divisor and incoming edge weight. This approach is independent of variable order, and allows for adjacent variable interchange as a local operation since the maximum value in a matrix is independent of the order in which the matrix elements are considered. The result of Theorem 1 is still applicable with this modified normalization rule and QMDDs formed with the modified normalization rule thus provide a canonical representation for all matrices encountered for reversible and quantum circuits.

### 2.5  Skipped variables

In implementing a sifting procedure for QMDDs, it is necessary to formulate an efficient adjacent variable interchange procedure. Such a procedure is more complex, as are QMDD-based matrix addition and multiplication, if intermediate variables are "skipped". This section provides a proof that intermediate variables in a QMDD representing a binary or multiple-valued reversible circuit are never skipped unless the outgoing edge points directly to the terminal vertex and has weight 0. We conjecture that this is also the case for quantum circuits.

*Definition 6.* Given the ordering $x_0 \prec x_1 \prec \cdots \prec x_{n-1}$, an edge from a vertex labeled $x_i$, $i > 0$, *skips* a variable if it points to the terminal vertex or it points to a vertex labeled $x_j$, $j < i - 1$.

**Theorem 2.** *A QMDD for a matrix representing a binary or multiple-valued reversible circuit has no edges that skip variables except for edges that point to the terminal vertex and have weight 0.*

*Proof:* It is clear from the definition of QMDD, that an edge with nonzero weight that skips a variable means the corresponding matrix has a sub-matrix of equal non-zero entries of dimension $r^k \times r^k$ for some $k > 0$. However, the matrix representing a binary or multiple-valued reversible function is a permutation matrix since reversible functions are bijections and thus has only 0 and 1 entries with a single 1 in each row and column. It follows that the QMDD for such a circuit can have no edges with nonzero weight that skip variables.                                                                    □

**Conjecture 2.1.** *Theorem 2 also holds for a QMDD corresponding to a circuit composed of quantum gates.*

The proof of Theorem 2 is based on the fact that reversible binary or multiple-valued logic gates/circuits are represented by permutation matrices. In the quantum case, the matrix describing a gate/circuit is an $r^n \times r^n$ unitary matrix with complex-valued elements [5,12]. The matrix is constructed from

an $r \times r$ unitary gate operation matrix $U$ (see examples in Table 1) using the Kronecker product and a similar complexity matrix composition operator. Detail of the construction procedure for the matrix describing a gate can be found in [10].

The following lemmas are a basis for approaching a proof of Conjecture 2.1.

**Lemma 1.** *Given the ordering $x_0 \prec x_1 \prec \cdots \prec x_{n-1}$, an edge from a QMDD nonterminal vertex labeled $x_{i+1}$ representing the matrix $M$ skips variable $x_i$ if, and only if, at least one $M_j$ in its $r^2$-partitioning itself partitions into $r^2$ identical sub-matrices.*

*Proof:* Suppose there is an $M_j$ that partitions into $r^2$ identical sub-matrices. Consider the QMDD representation of $M_j$. It does not have a start vertex labeled by $x_i$ since that vertex would have all its outgoing edges identical which makes it redundant. The correct representation for $M_j$, is an edge pointing to the common submatrix and $x_i$ is therefore skipped.

Conversely, consider the representation of $M$ with start vertex $sv$ labelled $x_{i+1}$. Assume edge $e_j$ from $sv$ skips variable $x_i$ which means it leads to a vertex representing a matrix $M_j$ whose value is independent of the value of $x_i$ the highest candidate selection variable. Clearly, this only happens if $M_j$ is itself composed of $r^2$ identical submatrices. □

**Lemma 2.** *The $r^n \times r^n$ unitary matrix for a quantum gate constructed from an $r \times r$ unitary operation matrix $U$ has the property that every row/column consists of either (a) a single 1 with all other entries 0, or (b) a distribution of the elements of a row/column of $U$ with the other elements 0.*

*Proof:* The proof follows directly from the procedure for constructing the matrix describing the gate (see [10]). □

The actual distribution of values described in Lemma 2 depends on which are the target, the control lines and the unconnected lines for the gate in question.

Empirical evidence shows Conjecture 2.1 is very likely true and we are currently working on a formal proof. The challenge lies in characterizing the complete situation to cover arbitrary quantum gates for binary and multiple-valued logic. It is important to note that QMDD are fully applicable, including the variable reordering procedure discussed below, even if Conjecture 2.1 is found to be false. The advantage of proving it to be true will be an even further improvement to the QMDD implementation.

## 3 INTERCHANGING ADJACENT VARIABLES

We consider the case of interchanging variables $\alpha$ and $\beta$ where the former is immediately above the latter (closer to the start vertex) in the QMDD. Recall that each vertex has $r^2$ outgoing edges. The key is, as mentioned

above, to perform the interchange as a local transformation. The technique presented is based on the technique developed by Miller and Drechsler [7] for multiple-valued decision diagrams.

Consider a vertex $\gamma$ labelled by variable $\alpha$. We construct a square matrix $\mathbf{T}$ of dimension $r^2 \times r^2$.

For $i = 0, 1, \ldots, r^2 - 1$,

- if the $i^{th}$ edge from $\gamma$ leads to a vertex $\delta$ labelled by variable $\beta$, then for $j = 0, 1, \ldots, r^2 - 1$, $\mathbf{T}_{ij}$ is set to point to the vertex pointed to by the $j^{th}$ edge of $\delta$ with the edge weight being the product of the edge weights on the $i^{th}$ edge from $\gamma$ and the $j^{th}$ edge from $\delta$;

- if the $i^{th}$ edge from $\gamma$ leads to a vertex $\delta$ not labelled $\beta$, then $\mathbf{T}_{ij}$ is set to the $i^{th}$ edge from $\gamma$ for $j = 0, 1, \ldots, r^2 - 1$.

Once $\mathbf{T}$ is constructed as above, the level interchange is made by *relabelling* $\gamma$ with $\beta$, and setting the $j^{th}$ edge from $\gamma$, $j = 0, 1, \ldots, r^2 - 1$ to point to a vertex labelled $\alpha$ whose $i$-th edge, $i = 0, 1, \ldots, r^2$-1, points to the vertex pointed to by $\mathbf{T}_{ij}$. During this construction the vertices are normalized as described in Definition 5. It is easily seen that following this construction, vertex $\gamma$, now labelled $\beta$, is the top vertex of a QMDD representing the same matrix it did when originally labelled $\alpha$.

The complete level interchange is accomplished by performing the above for all vertices originally labelled $\alpha$. These are readily identified as we use a separate unique table [16] for each variable. The idea of relabelling these vertices, as opposed to creating new vertices, is critical as it means that edges leading to them, and the vertices from whence those edges originate, are unaffected by the interchange. When a vertex is relabelled it must be removed from one unique table and entered into the unique table corresponding to its new variable label but this is a relatively simple operation given the data structures used for QMDD [11].

The vertices originally labeled $\beta$ are affected as edges to them are removed. The use of reference count garbage collection [11] accounts for when a vertex can be deleted (actually reused) or must be retained.

Note that no vertex above or below the two levels being interchanged is affected except for changing the reference counts of vertices immediately below. The result is that adjacent variable interchange is a local operation.

## 4  SIFTING QMDDS

Given the above method for adjacent variable interchange, variable reordering for QMDDs is readily implemented using an approach based on Rudell's sifting approach [13] developed for binary decision diagrams.

In general terms, our sifting method proceeds as follows:

**QMDD Sifting Procedure:**

i) Select a variable $\alpha$ that labels the most vertices in the QMDD. In the event of a tie, choose the variable closest to the terminal vertex.

ii) Sift $\alpha$ to the bottom (closest to the terminal vertex) of the QMDD by a sequence of adjacent variable interchanges.

iii) Sift $\alpha$ to the top of the QMDD by a sequence of adjacent variable interchanges.

iv) During steps (ii) and (iii) a record is kept of the position of $\alpha$ that yields the smallest vertex count in the QMDD, so now sift $\alpha$ back down to that position.

v) Repeat steps (i) to (iv) until each variable has been sifted into its *best* position noting that once a variable is selected for sifting, it is not selected a second time.

Note that the size of the QMDD after each variable interchange required in step (iv) is determined by checking reference counts for the vertices for the two variables being interchanged. It is not necessary to traverse the entire QMDD. There are $n!$ possible orderings of $n$ variables. The sifting method examines on the order of $n^2$ orderings, and determines the ordering among this subset that results in the smallest QMDD.

## 5 EXPERIMENTAL RESULTS

The QMDD package is implemented in C. The results reported here were run on a laptop computer with a 1.73 GHz Intel Pentium M processor and 1GB of RAM running LINUX on a 256MB virtual machine under VMware 5.5. We used LINUX in order to compare our implementation to QuIDDPro 3.0(beta) [15] which is available as an executable only. We used the gcc 4.0.0 C compiler with level 4 optimization to compile the QMDD package.

### 5.1 Binary examples

Results for a number of binary functions from Maslov's [6] benchmark web site are reported in Table 4. For each circuit, we give the following information:

- type – nct: circuit uses *NOT*, controlled-*NOT* and Toffoli gates; qc: circuit uses controlled-not, $V$ and $V^+$ gates.

- lines – number of lines in the circuit,

- gates – number of gates in the circuit,

- number of vertices before sifting,

- time to build the QMDD – CPU msec. using the standard library time.h routines,

- number of vertices after sifting,

- time to sift QMDD – CPU msec. using the standard library time.h routines,

- percentage vertex count reduction by sifting,

- maximum number of vertices encountered during sifting – this is an indicator of how large the QMDD might be but is not necessarily the maximum since sifting does not consider all variable orderings.

The results show that the effect of sifting varies significantly from example to example. A low improvement can be a result of having started from what is already a good ordering, the fact that the sifting heuristic does not visit all possible variable orderings, or, the function's QMDD representation is insensitive to variable ordering.

The results for the "hidden-weight-bit" examples hbw4 – hbw12 are interesting. They show the size of the QMDD can grow exponentially with the number of lines in the circuit. The benefit gained by sifting also increases with the number of lines.

Table 4 also shows the results of using QuIDDPro Version 3.0(beta) [15] on the same computer. On average, for the circuits shown, the number of vertices for the QuIDDPro representation is 2.06 times the number for the QMDD representation prior to sifting. This is as expected since a nonterminal QuIDDPro vertex has two outgoing edges while a nonterminal QMDD vertex has four outgoing edges for binary functions. What is interesting is how much the ratio can differ from 2. The largest sized decision diagrams for the circuits shown is for cyc17_3 where the ratio is 2.47. QuIDDPro uses the highly efficient CUDD decision diagram package [14] and also offers considerably more functionality than the current QMDD implementation. QuIDDPro is designed for binary reversible and quantum gates and circuits.

### 5.2 Ternary Examples

There are as yet no established benchmarks for multiple-valued reversible and quantum circuits available in the literature. This is largely because CAD tools for designing and simulating such circuits are not generally well developed. Indeed it is hoped that QMDD will be helpful in this regard.

Table 5 contains some ternary examples. The first is the reversible ternary adder from [10] shown in Fig. 2. The initial QMDD is relatively small (23 vertices) but even in this case sifting results in notable reduction of the number of vertices.

The S circuits are highly regular. An S circuit with $n$ lines has $n-1$ gates where gate $g_i$ is a C1 gate with target $x_i$ and a single 1-control $x_{i+1}$. As expected, given this regular and quite simple structure, the QMDDs are small (the number of vertices is twice the number of lines in the circuit) and can

| name | type | lines | gates | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5mod5 | nct | 6 | 17 | 28 | 0 | 16 | 8 | 43.9% | 28 | 45 | 77 |
| 6symd2 | nct | 10 | 20 | 247 | 7 | 170 | 42 | 31.2% | 478 | 299 | 117 |
| 9symd2 | nct | 12 | 28 | 229 | 7 | 184 | 50 | 19.7% | 558 | 445 | 194 |
| c2 | nct | 35 | 116 | 150 | 30 | 136 | 289 | 9.3% | 504 | 348 | 1546 |
| c2 | qc | 35 | 305 | 150 | 220 | 136 | 241 | 9.3% | 504 | 348 | 10245 |
| c3-17 | nct | 3 | 6 | 10 | 0 | 10 | 4 | 0.0% | 11 | 21 | 22 |
| c410184 | nct | 14 | 46 | 39 | 0 | 33 | 43 | 15.4% | 86 | 86 | 274 |
| c410184 | qc | 14 | 74 | 39 | 0 | 33 | 52 | 15.4% | 86 | 86 | 492 |
| cyc17-3 | nct | 20 | 48 | 236 | 10 | 42 | 131 | 82.2% | 418 | 584 | 880 |
| ham3 | nct | 3 | 5 | 10 | 0 | 10 | 0 | 0.0% | 10 | 21 | 21 |
| ham15 | nct | 15 | 132 | 4522 | 140 | 2638 | 488 | 42.7% | 13878 | 7547 | 2051 |
| hwb4 | nct | 4 | 11 | 22 | 0 | 20 | 4 | 9.1% | 22 | 45 | 43 |
| hwb4 | qc | 4 | 21 | 22 | 0 | 20 | 8 | 9.1% | 22 | 45 | 83 |
| hwb7 | nct | 7 | 289 | 179 | 30 | 155 | 16 | 13.4% | 181 | 351 | 2117 |
| hwb8 | nct | 8 | 614 | 343 | 120 | 280 | 28 | 18.4% | 351 | 684 | 6254 |
| hwb9 | nct | 9 | 1541 | 683 | 640 | 520 | 40 | 23.9% | 690 | 1349 | 24566 |
| hwb10 | nct | 10 | 3595 | 1331 | 2920 | 960 | 59 | 27.9% | 1347 | 2650 | 98704 |
| hwb11 | nct | 11 | 9314 | 2639 | 14290 | 1730 | 130 | 34.4% | 2685 | 5223 | 478555 |
| hwb12 | nct | 12 | 18393 | 5167 | 53350 | 3185 | 265 | 38.4% | 5254 | 10283 | 1722050 |
| rd84d1 | nct | 15 | 28 | 3588 | 27 | 396 | 117 | 89.0% | 4415 | 1252 | 255 |

TABLE 4
Experimental Results – binary circuits.
(a) vertices before sifting (b) time to build QMDD (msec) (c) vertices after sifting (d) time to sift QMDD (msec) (e) vertex count reduction by sifting (f) max. vertices during sifting (g) QuIDDPro vertices (h) time to build BDD (msec)

| name | lines | gates | a | b | c | d | e | f |
|------|------:|------:|------:|------:|------:|------:|------:|------:|
| adder | 4 | 16 | 23 | 0 | 15 | 4 | 34.8% | 39 |
| S25 | 25 | 24 | 50 | 11 | 50 | 312 | 0.0% | 184 |
| S50 | 50 | 49 | 100 | 58 | 100 | 1370 | 0.0% | 384 |
| S75 | 75 | 74 | 150 | 168 | 150 | 3112 | 0.0% | 584 |
| S100 | 100 | 99 | 200 | 370 | 200 | 5550 | 0.0% | 784 |
| R5-25 | 5 | 25 | 132 | 11 | 124 | 15 | 6.1% | 185 |
| R5-50 | 5 | 50 | 204 | 35 | 200 | 23 | 2.0% | 205 |
| R5-75 | 5 | 75 | 196 | 58 | 193 | 19 | 1.5% | 208 |
| R5-100 | 5 | 100 | 203 | 85 | 198 | 23 | 2.5% | 213 |
| R10-25 | 10 | 25 | 1308 | 74 | 442 | 105 | 66.2% | 1308 |
| R10-50 | 10 | 50 | 9670 | 687 | 6839 | 1081 | 29.3% | 12151 |
| R10-75 | 10 | 75 | 41170 | 6776 | 34991 | 7901 | 15.0% | 45826 |
| R10-100 | 10 | 100 | 51133 | 12361 | 46556 | 10646 | 9.0% | 52785 |

TABLE 5
Experimental Results – ternary circuits.

(a) vertices before sifting (b) time to build QMDD (msec) (c) vertices after sifting (d) time to sift
QMDD (msec) (e) vertex count reduction by sifting (f) max. vertices during sifting

be shown to have a very regular structure. Sifting results in no improvement, but, and this is a disadvantage of the heuristic, considerable computation is required.

Each R*n*-*m* circuit has *n* lines and *m* pseudo-randomly generated gates. Each gate is randomly chosen to be C1 or C2 with a randomly chosen target and a single randomly chosen control. The control is randomly chosen to be a 1 or 2-control. The improvement by sifting is as expected quite variable.

These examples indicate that QMDD construction and sifting are reasonably practical for quite large binary and ternary problems. It is a concern that the cost of sifting seems quite high for large ternary examples. We are looking for ways to improve the implementation, but it may well simply be that we are dealing with representations of very large and very complex matrices in those cases.

The QMDD package is applicable for higher radix problems as well. To put this in better context, we note that constructing a QMDD for an $r-$valued, $n$-line circuit with $m$ gates is equivalent to constructing $m$ $r^n \times r^n$ matrices and performing $m-1$ matrix multiplications but this approach does so in a tractable manner.

## 6  CONCLUSIONS AND FUTURE WORK

This paper has considered variable reordering in QMDDs and has described a sifting technique for determining good variable orderings. The experimental results show the method can be quite effective but also that it can be computationally expensive sometimes with little benefit.

We are working on criteria that may help in determining when it is likely to be useful to apply sifting to a QMDD. We are studying the special structure of QMDDs not evident in general decision diagrams such as the frequency of edges of weight 0 pointing to the terminal vertex and the regular structure of a QMDD resulting from the regular structure of the matrices being represented. We are also exploring how sifting can be used to transform QMDD to variable orderings that will better illuminate the structure of the matrices for synthesis.

## ACKNOWLEDGEMENTS

## REFERENCES

[1] C.H. Bennett. (1973). Logical reversibility of computation. *IBM J. Research and Development*, 17-6:525–532.

[2] R.E. Bryant. (1986). Graph-based algorithms for Boolean function manipulation. In *IEEE Transactions on Computers*, volume C35-8, pages 677–691.

[3] E.M. Clarke K.L. McMillan X. Zhao M. Fujita and J. Yang. (1993). Spectral applications for large Boolean functions with applications to technology mapping. In *Proc. ACM/IEEE Design Automation Conference*, pages 54–60.

[4] P.C. McGeer M. Fujita and J.C.-Y. Yang. (1997). Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. *Formal Methods in System Design*, 10(2/3):149–169.

[5] D.C. Marinescu and G.M. Marinescu. (2005). *Approaching Quantum Computing*. Pearson Prentice Hall.

[6] D. Maslov, (2007). Reversible logic synthesis benchmarks page. http://www.cs.uvic.ca/∼dmaslov/.

[7] D.M. Miller and R. Drechsler. (2003). Augmented sifting of multiple-valued decision diagrams. In *Proc. 2003 Int. Symposium on Multiple-Valued Logic*, pages 275–282.

[8] D.M. Miller, G. Dueck, and D. Maslov. (2004). A synthesis method for MVL reversible logic. In *Proc. 2004 Int. Symposium on Multiple-Valued Logic*, pages 74–80.

[9] D.M. Miller, D.Y. Feinstein, and M.A. Thornton. (2007). Variable reordering and sifting for QMDD. In *IEEE International Symposium on Multiple-Valued Logic*. to appear.

[10] D.M. Miller and M.A. Thornton. (2006). QMDD: A decision diagram structure for reversible and quantum circuits. In *IEEE International Symposium on Multiple-Valued Logic*. CD-ROM.

[11] D.M. Miller, M.A. Thornton, and D. Goodman. (2006). A decision diagram package for reversible and quantum circuits. In *Proc. IEEE World Congress on Computational Intelligence*. CD-ROM.

[12] M.A. Nielsen and I.L. Chuang. (2000). *Quantum Computation and Quantum Information*. Cambridge University Press.

[13] R. Rudell. (1993). Dynamic variable ordering for ordered binary decision diagrams. In
     *Proc. International Conference on Computer-Aided Design*, pages 42–47.

[14] F. Somenzi, (2007). The CUDD package version 2.4.0. http://vlsi.colorado.edu/∼fabio/.

[15] G.F. Viamontes, I.L. Markov, and J.P. Hayes, (2006). QuIDDPro: High-performance
     quantum circuit simulation. vlsicad.eecs.umich.edu/Quantum/qp/.

[16] S.N. Yanushkevitch, D.M. Miller, V.P. Shmerko, and R.S. Stankovic. (2006). *Decision
     Diagram Techniques for Micro- and Nanoelectronic Design*.CRC Taylor and Francis.