Decision Diagram Techniques for Reversible and Quantum Circuits

Michael Miller
Department of Computer Science
University of Victoria
Victoria, BC CANADA
mmiller@uvic.ca

Universität Bremen (July 2008 – March 2009)
Outline

• Motivation
• Reversible and Quantum Gates & Circuits
• Related Work
• Decision Diagrams
  – QuIDD
  – XQDD
  – QMDD
• Circuit Equivalence
• Ongoing work
Why quantum circuits?

- Potential for low power and greatly increased density.
- Quantum circuits offer a new computational paradigm.
- Will potentially change what is computable?
Why Decision Diagrams?

• DD offer consistent representations for binary and multiple-valued reversible and quantum circuits.

• Applicable to
  – Specification
  – Simulation
  – Circuit equivalence checking
  – Synthesis?
Reversible and Quantum Circuits

• A gate or circuit is **reversible** iff each input assignment is mapped to a unique output assignment.

• A reversible circuit is a cascade of reversible gates with no fanout and no feedback.

• Quantum gates and circuits are all reversible since the operation of any quantum gate is defined by a unitary matrix (hence always has an inverse).
A complex-valued square matrix $A$ is **unitary** iff $A^H A = I$.

Unitary matrices are **closed under multiplication**, raising to an integer power and inversion.

**Inverse**: $U^{-1} = U^H$.

**Permutation matrices** are a special case of unitary matrices – hence reversible gates and circuits are functionally a special case of quantum gates and circuits.
Some Reversible Gates

**NOT**

\[
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
\]

**CNOT**

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]
### Toffoli Gate

#### Circuit Diagram

```
  o---a---o
     |     |
     |    ^
     v    |
  o-----b---o
        |
        |
  o-----c---o
```

#### Truth Table

<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>a b c</td>
<td>a(^<em>) b(^</em>) c(^*)</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1 0 1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 1 1</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1 1 0</td>
</tr>
</tbody>
</table>

#### Boolean Matrix

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]
Fredkin

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(a^*)</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1 1 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 0 1</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>
Interpreting the Matrices

\[
    \begin{bmatrix}
        \alpha_0 \\
        \alpha_1 \\
        \alpha_2 \\
        \alpha_3 \\
        \alpha_4 \\
        \alpha_5 \\
        \alpha_6 \\
        \alpha_7 \\
    \end{bmatrix}
    =
    \begin{bmatrix}
        1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
        0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
        0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
        0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
        0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
        0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
        0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
        0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
    \end{bmatrix}
    \begin{bmatrix}
        \alpha_0 \\
        \alpha_1 \\
        \alpha_2 \\
        \alpha_3 \\
        \alpha_4 \\
        \alpha_5 \\
        \alpha_6 \\
        \alpha_7 \\
    \end{bmatrix}
\]

The output will be in state 6 if the input is in state 7.
Qubits

\[ |\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle \]

\[ |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \]

\[ |\psi\rangle = \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} \]

\[ |\Psi\rangle = \alpha_{00} |00\rangle + \alpha_{01} |01\rangle + \alpha_{10} |10\rangle + \alpha_{11} |11\rangle = \begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix} \]
Example

\[ M = M_3 \times M_2 \times M_1 \times M_0 \]
Interpreting the Result

The adder is a target function embedded in a reversible function.
Quantum Gates

**Controlled-V**

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \frac{1}{2}+\frac{1}{2}i & \frac{1}{2}-\frac{1}{2}i \\
0 & 0 & \frac{1}{2}-\frac{1}{2}i & \frac{1}{2}+\frac{1}{2}i
\end{bmatrix}
\]

**Controlled-V\(^+\)**

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \frac{1}{2}-\frac{1}{2}i & \frac{1}{2}+\frac{1}{2}i \\
0 & 0 & \frac{1}{2}+\frac{1}{2}i & \frac{1}{2}-\frac{1}{2}i
\end{bmatrix}
\]
Decision Diagrams


Quantum Information Decision Diagrams

• QuIDD
  – Developed at University of Michigan

  – Build on top of CUDD

  – Has a very good user interface modelled on MATLAB.
Leaves are indices into a table of complex numbers.
• **XQDD**
  – Developed at National Taiwan University

XQDD
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

QuIDD

XQDD
Quantum Multiple-Valued Decision Diagrams (QMDD)

• Miller and Xiao (UVic), Thornton, Goodman, Feinstein (SMU)

• Introduced at 2006 International Symposium on Multiple-Valued Logic, Singapore.

QMDD Basic Concept

Consider the binary case:

\[
M = \begin{bmatrix}
M_0 & M_1 \\
M_2 & M_3
\end{bmatrix}
\]

Use a single terminal node with value 1.

Add a complex valued multiplier to every edge.
Normalization

**Definition:** A QMDD vertex is *normalized* if the weight on edge $j$ is 1 where the weight on every edge $i$, $i<j$, is 0.

Normalization makes the representation of a matrix unique up to variable order.

Department of Computer Science
University of Victoria
The Multiple-valued Case

\[ M = \begin{bmatrix}
M_0 & M_1 & \cdots & M_{r-1} \\
M_r & M_{r+1} & \cdots & M_{2r-2} \\
\vdots & \vdots & \ddots & \vdots \\
M_{r^2-r} & M_{r^2-r+1} & \cdots & M_{r^2-1}
\end{bmatrix} \]
Adding Two QMDD

Based on Bryant's basic apply operation for BDDs.

\[
A = \begin{bmatrix}
A_0 & A_1 \\
A_2 & A_3
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
B_0 & B_1 \\
B_2 & B_3
\end{bmatrix}
\]
Multiplying two QMDD

\[
A = \begin{bmatrix}
A_0 & A_1 \\
A_2 & A_3
\end{bmatrix}
\quad B = \begin{bmatrix}
B_0 & B_1 \\
B_2 & B_3
\end{bmatrix}
\]

\[
A \times B = \begin{bmatrix}
A_0 B_0 + A_1 B_2 & A_0 B_1 + A_1 B_3 \\
A_2 B_0 + A_3 B_2 & A_2 B_1 + A_3 B_3
\end{bmatrix}
\]
Kronecker (Tensor) Product

• The two matrices do not have to have the same dimension – but in this application they will be square and of size a power of 2.

\[
A \otimes B = \begin{bmatrix}
A_0 & A_1 \\
A_2 & A_3
\end{bmatrix} \otimes B = \begin{bmatrix}
A_0 \otimes B & A_1 \otimes B \\
A_2 \otimes B & A_3 \otimes B
\end{bmatrix}
\]
Zero and Identity Matrices

- A zero matrix is represented by an edge pointing to the terminal vertex with weight 0.
- An identity matrix is represented by

![Diagram showing a zero and identity matrix]

- The simplifications to computational routines are straightforward.
Each edge weighted 0 is a zero matrix the dimension of which depends on the number of variables skipped.
Implementation Techniques

• Several standard decision diagram techniques are used to make QMDD computation efficient.

• Additional techniques to improve efficiency have also been introduced.
Unique Table

Make the number of buckets a power of two.

Use a separate unique table for every variable.
DD Computed Table

- We use a very simple computed table

- Each table entry is a single computation with no chaining
Garbage Collection

• Necessary for large examples.
• Uses simple reference count technique where the reference count is the number of edges pointing to a vertex.
• Garbage collection is triggered when the number of vertices reaches a preset limit.
• Vertices with 0 reference count are moved to an available space list for recycling.
### Complex Value Table

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>0.5 + 0.5i</td>
<td>0.5 + 0.5i</td>
</tr>
<tr>
<td>3</td>
<td>1.0 + 1.0i</td>
<td>1.0 + 1.0i</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Complex Operation Lookup Tables

- A separate lookup table is built dynamically for each complex operation – addition, multiplication and division.
Gate Computed Table

• A second computed table is used to avoid duplicate gate computation

• Once again there is no chaining.
Circuit Equivalence Checking

Given two circuits designed to implement the same target function $F$ and thus having common primary input and primary output sets, but possibly different ancillary input and garbage output sets, show that the two circuits exhibit the same functional response at the primary outputs for all assignments to the primary input for which $F$ is defined, regardless of the ordering and naming of the circuit inputs and outputs.
A Restricted Case

• The circuit is binary at its inputs and outputs.
• The line names are consistent for the primary inputs and outputs.

• Note: This places no restriction on the gates in the circuit – they can be reversible or quantum.
Example
Another Example
Circuit Normalization

QMDD variable order from top to bottom

ancillary inputs (order is not important)
primary inputs (ordered)

reversible circuit

garbage outputs (order is not important)
primary outputs (ordered)
Comparing Two Circuits

- Circuits do not have to have the same number of lines.
- Names must match.
  - QMDD variable order from top to bottom.
  - Reversible circuit structure with inputs and outputs.

Department of Computer Science
University of Victoria
Setting an Ancillary Input to 0

\[
\begin{bmatrix}
  g_0 \\
  g_1
\end{bmatrix} =
\begin{bmatrix}
  M_0 & M_1 \\
  M_2 & M_3
\end{bmatrix}
\begin{bmatrix}
  f_0 \\
  f_1
\end{bmatrix}
\]

\[
\begin{bmatrix}
  ) \\
  g_1
\end{bmatrix} =
\begin{bmatrix}
  M_0 & \phi \\
  M_2 & \phi
\end{bmatrix}
\begin{bmatrix}
  f_0 \\
  f_1
\end{bmatrix}
\]

Since the ancillary inputs are at the top of the QMDD they can be dealt with by a simple recursive descent procedure.
Setting an Ancillary Input to 1

\[
\begin{bmatrix}
  g_0 \\
  g_1
\end{bmatrix} =
\begin{bmatrix}
  M_0 & M_1 \\
  M_2 & M_3
\end{bmatrix}
\begin{bmatrix}
  f_0 \\
  f_1
\end{bmatrix}
\]

\[
\begin{bmatrix}
  g_0 \\
  g_1
\end{bmatrix} =
\begin{bmatrix}
  M_0 & M_1 \\
  M_2 & M_3
\end{bmatrix}
\begin{bmatrix}
  f_0 \\
  f_1
\end{bmatrix}
\]

Note we don’t have to do this as we only Work with the matrix.
Collapsing a Garbage Output

\[
\begin{bmatrix}
  g_0 \\
g_1
\end{bmatrix}
= 
\begin{bmatrix}
  M_0 & M_1 \\
  M_2 & M_3
\end{bmatrix}
\begin{bmatrix}
  f_0 \\
f_1
\end{bmatrix}
\]

\[
\begin{bmatrix}
  g_0 \\
g_1
\end{bmatrix}
= 
\begin{bmatrix}
  M_0 + M_2 & M_1 + M_3 \\
  \phi & \phi
\end{bmatrix}
\begin{bmatrix}
  f_0 \\
f_1
\end{bmatrix}
\]

**IMPORTANT**: Recall that \( M \) is a permutation matrix.
A Simple Example

```plaintext
# test circuit 2
.version 1.0
.numvars 3
.variables c b a
.inputs  c b a
.outputs g g f
.constants 0--
.garbage 11-
.begin
t2 b,c
t2 c,a
.end
```

![Diagram of test circuit 2]
C A B
0--
1-1
Swapping lines 1 0
top edge weight 1
1 1 0| C[ 1:1 2:1 3:1 4:1 ] 1553296
1 1 1| A[ 5:1 6:1 T:0 T:0 ] 1553008
1 1 2| A[ T:0 T:0 7:1 8:1 ] 1553080
1 1 3| A[ T:0 T:0 8:1 7:1 ] 1553152
1 1 4| A[ 6:1 5:1 T:0 T:0 ] 1553224
2 2 BD 5| B[ T:1 T:0 T:0 T:0 ] 1551208
2 2 6| B[ T:0 T:0 T:1 T:0 ] 1552504
2 2 7| B[ T:0 T:1 T:0 T:0 ] 1552648
2 2 BD 8| B[ T:0 T:0 T:0 T:1 ] 1551280

1........ 0 0
..1..... 1 2
.....1.. 2 5
.......1 3 7
.......1. 4 6
.....1... 5 4
...1.... 6 3
..1..... 7 1
Function after assigning constants

top edge weight 1
1 0  0| C[ 1:1 NULL 2:1 NULL ] 1553368
1 0  1| A[ 3:1 4:1 T:0 T:0 ] 1553008
1 0  2| A[ T:0 T:0 5:1 6:1 ] 1553152
1 0 BD 3| B[ T:1 T:0 T:0 T:0 ] 1551208
1 0 BD 4| B[ T:0 T:0 T:1 T:0 ] 1552504
1 0 BD 5| B[ T:0 T:0 T:0 T:1 ] 1551280
1 0  6| B[ T:0 T:1 T:0 T:0 ] 1552648

1...----
..1.----
....----
....----
....----
....----
....----
....----
....----
....----
.1...----
.1.----
Function after collapsing garbage

top edge weight 1
1 0 0 | A [ 1:1  2:1 NULL NULL ] 1553584
1 0 BDI 1 | B [ T:1  T:0  T:0  T:1 ] 1551352
1 0 2 | B [ T:0  T:1  T:1  T:0 ] 1552288

\[
\begin{bmatrix}
\alpha_0 + \alpha_3 \\
\alpha_1 + \alpha_2
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
\alpha_0 \\
\alpha_1 \\
\alpha_2 \\
\alpha_3
\end{bmatrix}
\]
Please enter the name of the circuit file ==> ReVLib/sym9-a.txt

number of variables 12
Variables: S2 S3 S4 X1 X2 X3 X4 X5 X6 X7 X8 X9
Ancillary: 000--------
Garbage: 111-11111111
Swapping lines 8 0
    47.00 msec
Top edge address 1978128 weight 1
    0.00 msec

Please enter the name of the circuit file ==> ReVLib/sym9-b.txt

number of variables 10
Variables: J X1 X2 X3 X4 X5 X6 X7 X8 X9
Ancillary: 0---------
Garbage: 1-11111111
Swapping lines 8 0
    62.00 msec
Top edge address 1978128 weight 1
    0.00 msec

Program completed normally
Don’t-cares in the Target Function

• A **don’t-care** in the target function is the situation where for a particular primary input assignment, we don’t-care what the output of one or more of the primary outputs is.

• A don’t-care is **total** if it applies to all primary outputs and **partial** if it applies to only some of the primary outputs.
<table>
<thead>
<tr>
<th>E</th>
<th>D</th>
<th>C</th>
<th>B</th>
<th>A</th>
<th>E</th>
<th>D</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Partial don’t-cares**

**Total don’t-cares**

**NOTE** that output E is a garbage output since it is a don’t-care for all input assignments.
A Method for Total Don’t-cares

• Let $M$ denote the matrix describing the behaviour of a circuit after assigning any ancillary inputs and collapsing any garbage outputs.

• The matrix $M$ has
  – A row for each assignment to the target function’s primary outputs.
  – A column for each assignment to the target function’s primary inputs for the section of the truth table identified by the assignment to any ancillary inputs.
• If an output assignment is a total don’t-care, this means we want to ignore the corresponding input assignment.

• This can be done by multiplying $M$ by a matrix $D$ which is a diagonal matrix with 1’s on the diagonal in positions corresponding to care assignments and 0’s on the diagonal in positions corresponding to don’t-care assignments.
Example: mod-10 counter

```
# mod 10 counter
.version 1.0
.numvars 4
.variables d c b a
.inputs d c b a
.outputs d c b a
.constants ----
garbage ----
.begin
0001
0010
0011
0100
0101
0110
0111
1000
1001
0000
----
----
----
----
----
.end
```

```
# mod 10 counter
.version 1.0
.numvars 4
.variables d c b a
.inputs d c b a
.outputs d c b a
.constants ----
garbage ----
.begin
0001
0010
0011
0100
0101
0110
0111
1000
1001
0000
1010
1011
1100
1101
1110
1111
.end
```

```
.version bidirectional synthesis
.numvars 4
.variables D C B A
.inputs D C B A
.outputs D C B A
.constants ----
garbage ----
.begin
T2  D B
T4  B C D A
T4  A B D C
T4  A B C D
T2  D B
T3  A D B
T3  A B C
T2  A B
T3  B D A
T1  A
# There are 10 gates.
# Quantum cost 58
.end
```
\[ M = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \]

\[ D = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \]

\[ M \times D = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \]
An Alternative Circuit

# mod 10 counter
.version 1.0
.numvars 4
.variables d c b a
.inputs d c b a
.outputs d c b a
.constants ----
.garbage ----
.begin
0001
0010
0011
0100
0101
0110
0111
1000
1001
1000
----
----
----
1011
1100
1101
1110
1111
1010
.end

.version 1.0
.numvars 4
.variables d c b a
.inputs d c b a
.outputs d c b a
.constants ----
.garbage ----
.begin
0001
0010
0011
0100
0101
0110
0111
1000
1001
0000
1011
1100
1101
1110
1111
1010
.end

.version reverse solution
.numvars 4
.variables D C B A
.inputs D C B A
.outputs D C B A
.constants ----
.garbage ----
.begin
T3 A B C
T4 A C D B
T3 A D B
T4 A B C D
T3 A B D
T2 A B
T1 A
# There are 7 gates.
# Quantum cost 43
.end
Partial Don’t-cares

Pre-multiply by a matrix to combine rows to get a single function.
## Experimental Results

<table>
<thead>
<tr>
<th>Benchmark*</th>
<th>n</th>
<th>gates</th>
<th>Equal CPU (sec.)</th>
<th>Not Equal CPU (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>rd53_133</td>
<td>7</td>
<td>12</td>
<td>0.005</td>
<td>0.007</td>
</tr>
<tr>
<td>ckt1_cycle_151</td>
<td>9</td>
<td>1487</td>
<td>1.700</td>
<td>2.153</td>
</tr>
<tr>
<td>ckt3_cycle_155</td>
<td>10</td>
<td>26468</td>
<td>116.345</td>
<td>122.039</td>
</tr>
<tr>
<td>ckt5_cycle_158</td>
<td>9</td>
<td>10276</td>
<td>13.853</td>
<td>14.758</td>
</tr>
<tr>
<td>ckt6_cycle_160</td>
<td>15</td>
<td>10740</td>
<td>192.769</td>
<td>344.479</td>
</tr>
<tr>
<td>decod24-v2_43</td>
<td>4</td>
<td>6</td>
<td>0.000</td>
<td>0.046</td>
</tr>
<tr>
<td>hwb6_56</td>
<td>6</td>
<td>126</td>
<td>0.078</td>
<td>0.140</td>
</tr>
<tr>
<td>hwb9_119</td>
<td>9</td>
<td>1544</td>
<td>1.997</td>
<td>2.137</td>
</tr>
<tr>
<td>sym9_146</td>
<td>12</td>
<td>28</td>
<td>0.030</td>
<td>0.047</td>
</tr>
</tbody>
</table>

*Source: [www.revlib.org](http://www.revlib.org)
Quantum Circuit Equivalence

- Global phase
- Relative phase


- Collapsing garbage output and isolating outputs for partial don’t-cares is different.
Ongoing Work

• Improving the implementation of QMDD.
• A proper user interface.
• Circuit equivalence:
  – The multiple-valued case: implemented but needs to be fully tested.
  – Implement and test the ‘true’ quantum case.
  – Identifying points of difference for two circuits.
  – Comparison to SAT-based techniques.
• Synthesis algorithms employing decision diagram techniques.