File format for Representation of Reversible Functions and Reversible Circuits

Robert Wille[1], Daniel Große[1], Lisa Teuber[1],
Gerhard W. Dueck[2], and Rolf Drechsler[1]

[1]Institute of Computer Science, University of Bremen
28359 Bremen, Germany
{rwille,grosse,teuber,drechsle}@informatik.uni-bremen.de

[2]Faculty of Computer Science, University of New Brunswick
Fredericton, NB, Canada
gdueck@unb.ca

Last revision: 20^{th} May 2008 (Version 1.0)[1]

Abstract. This document describes the proposed format for the specification of reversible functions and reversible circuits. The format allows a destinct representation of reversible functions and reversible circuits, respectively, and is used at RevLib (www.revlib.org) for exchanging benchmarks and results.

1 File Formats

This section describes the standardized file format used at RevLib for the specification of reversible functions and circuits, respectively. Both formats are defined in ASCII files consisting of two parts: the header and the specification.

1.1 Header

The header contains information about the characteristics of the instance, i.e.:

- *Version:* In order to support further enhancements, any changes will be associated with a version number. The version of a respective file is given by the line starting with `.version`. The current version of this revison is `1.0`.
- *Comments:* Comments provide human-readable information about the respective instance, e.g. a short description, authors, or similar. A comment line starts with a `#` character. All comment lines will be ignored by programs.
- *Number of Variables:* The line starting with `.varnum` signifies the number n of variables (lines) of the respective function (circuit). The number has to be a positive integer.

[1] Updates of this documentation will be published at RevLib (www.revlib.org).

- *Variables:* The line starting with `.variables` signifies the ordered list of identifiers for the respective variables (lines). An identifier has to be a sequence of letters, digits and underscores. In total, n different identifiers have to be defined in a `.variables` line (separated by spaces).
- *Inputs/Outputs:* The line starting with `.inputs` (`.outputs`) signifies the ordered list of names of the respective inputs (outputs). A name has to be a sequence of letters, digits and underscores. In total, n different names have to be defined in a `.inputs` (`outputs`) line (separated by spaces). Defining the inputs (outputs) is optional. By default the names of inputs (outputs) is equal to the respective identifier of the variable.
- *Constant Inputs:* Constant inputs can be definied by a line starting with `.constants` followed by a string containing the values for the input constants ('`1`' for constant one, '`0`' for constant zero and '`-`' if the respective input line is not constant) in the order they appear without any spaces. Defining constant inputs is optional. By default all inputs are not constants (i.e. for each input the value is '`-`').
- *Garbage Outputs:* Garbage outputs can be definied by a line starting with `.garbage` followed by a string indicating whether an output is garbage or not ('`1`' if the output is garbage and '`-`' if the output is not garbage) in the order they appear without any spaces. Defining garbage outputs is optional. By default all outputs are not garbage (i.e. for each output the value is '`-`').

Following the header either the function or the circuit have to be specified. Both specifications start with a line containing the string `.begin` and end with a line containing the string `.end`.

1.2 Function Specification

A function is specified by all its truth table entries. The ith line ($0 \leq i < 2^n$) of the specification gives the respective output values (`1` for one, `0` for zero and `-` for don't care) of the *ith* line of the truth table by a string in the order they appear without any spaces.

Example 1. Figure 1 (a) shows a reversible function including one constant input as well as garbage outputs and don't care outputs given as truth table. The respective specification of this function in the proposed format is shown in Figure 1 (b).

1.3 Circuit Specification

A reversible circuit is specified by all its gates. The specification lists the gates in the order they appear in the design. In the current version we distinguish five different gates (see Figure 2):

1	a	b	f	g_1	g_2
0	0	0	-	-	-
0	0	1	-	-	-
0	1	0	-	-	-
0	1	1	-	-	-
1	0	0	0	-	-
1	0	1	0	-	-
1	1	0	0	-	-
1	1	1	1	-	-

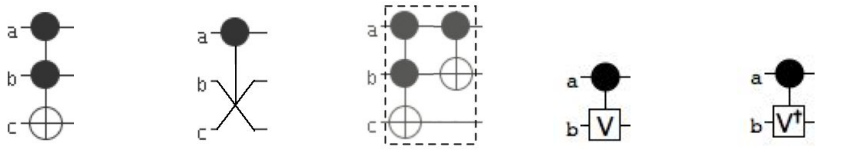
(a) Function

```
 1 # Embedded AND function
 2 .version 1.0
 3 .numvars 3
 4 .variables x y z
 5 .inputs 1 a b
 6 .outputs f g1 g2
 7 .constants 1--
 8 .begin
 9 ---
10 ---
11 ---
12 ---
13 ---
14 0--
15 0--
16 0--
17 1--
18 .end
```

(b) Function Format

Fig. 1. Format for Function Specification

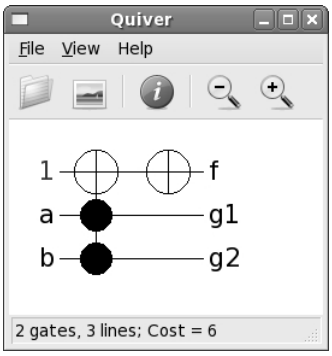
- *Multiple Control Toffoli gates (MCT)*
  A (multiple control) Toffoli gate is sifgnified by the character `t` and an integer indicating the size of the gate followed by a list of variables for the respective lines such that the the target line is at the end of the list. Note that Toffoli gates include (controled) NOT gates as well.
  Example: `t3 a b c`
- *Multiple Control Fredkin gate (MCF)*
  A (multiple control) Fredkin gate is signified by the character `f` and an integer indicating the size of the gate followed by a list of variables for the respective lines such that the the targets of the gates are at the end of the list.
  Example: `f3 a b c`
- *Peres gate (P)*
  A Peres gate is signified by the character `p` and an integer indicating the size of the gate followed by a list of variables for the respective lines such that the the targets of the gates are at the end of the list.
  Example: `p3 a b c`
- *V gate*
  A V gate – one of the elementary quantum gates (EQ) – is signified by the character `v` and an integer indicating the size of the gate (i.e. 2) followed by a list of variables for the respective lines such that the the target line is at the end of the list.
  Example: `v2 a b`
- *V+ gate*
  A V+ gate – one of the elementary quantum gates (EQ) – is signified by the characters `v+` and an integer indicating the size of the gate (i.e. 2) followed by a list of variables for the respective lines such that the the target line is at the end of the list.



(a) Toffoli gate (b) Fredkin gate (c) Peres gate (d) V gate (e) V+ gate

Fig. 2. Reversible gates

```
 1 # AND circuit
 2 .version 1.0
 3 .numvars 3
 4 .variables x y z
 5 .inputs 1 a b
 6 .outputs f g1 g2
 7 .constants 1--
 8 .begin
 9 t3 y z x
10 t1 x
11 .end
```



(a) Circuit Format (b) Circuit

Fig. 3. Format for Circuits

Example: `v+2 a b`

Example 2. Figure 3 (b) shows a Toffoli gate network realization of the function considered in Example 1 (drawn by Quiver). The respective specification of this circuit in the proposed format is shown in Figure 3 (a).