ReLoS

(known as RMRLS in the literature)

–

Reversible Logic Synthesizer

Pallav Gupta
James Donald

August 27, 2007
Version 0.2

Abstract

ReLoS is an integrated tool for synthesis, simulation, and test of reversible logic circuits.

1 Synopsis

relos [**-i** *file*] [**-h**|–**help**] [**-v**|–**version**]

2 Description

ReLoS can be started in two modes: *interactive* or *batch*

interactive mode starts if no command line options are supplied.

batch mode starts if the **-i** option is used to execute a series of commands from a file.

3 Options

-i *file* Execute commands specified in file.

-h|–**help** Show help message.

-v|–**version** Show version information.

4 Project Organization

`relos/doc` Documentation for *ReLoS*.

`relos/include` Links to all header files used in *ReLoS*.

`relos/src` Source code for *ReLoS*.

`relos/tools` Useful scripts for *ReLos*.

`relos/benchmarks` Benchmark examples.

`relos/INSTALL` Installation instructions for *ReLoS*.

`relos/AUTHORS` People who have contributed to *ReLoS*.

`relos/THANKS` People whose code/tools I have used in *ReLoS*.

`relos/ChangeLog` Main changes between revisions of *ReLoS*.

5 Requirements

GLib Version >= 2.2+ (http://www.gtk.org)

Perl Version >= 5.0004_03+ (http://www.cpan.org)

Argtable Version >= 2.0+ (http://argtable.sourceforge.net)

Platforms Red Hat Linux 8.0+

6 Version

Version: 0.2 of August 27, 2007.

7 Reporting Bugs

Report bugs to `pallav.gupta@villanova.edu`

8 Copyright and License

Misc If you use this tool in academic research, please send me an email and cite the following papers:
- P. Gupta, A. Agrawal, and N. K. Jha. An Algorithm for Synthesis of Reversible Logic Circuits. IEEE Trans. Computer-Aided Design.
- J. Donald, P. Gupta, and N. K. Jha. Reversible Logic Synthesis with Fredkin and Peres Gates. Journal on Emerging Technologies in Computing Systems.

9 Authors

Pallav Gupta
Villanova University

Email: `pallav.gupta@villanova.edu`
WWW: `http://pandim.ece.villanova.edu/`

James Donald
Princeton University, NVIDIA Corporation

10 Basic Commands

The following basic commands are currently supported in *ReLoS*.

list [**-h**|**–help**] List all commands available in *ReLoS*.

help [**-h**|**–help**] ***command*** Display usage information on a command.

quit [**-h**|**–help**] Terminate session.

time [**-h**|**–help**] Display elapsed time between two successive *time* calls.

history [**-h**|**–help**] [***num***] Display list of successfully executed commands.

source [**-h**|**–help**] [**-x**] ***file*** Execute commands given in file.
Note: Any line starting with a # is treated as a comment.

print_stats [**-h**|**–help**] Prints simple statistics about the reversible network.

11 Reversible Logic Netlists

The following commands are used to read/write reversible logic netlists.

read_netlist [**-h**|**–help**] ***file*** Read a netlist to create an internal reversible logic network.

write_netlist [**-h**|**–help**] [***file***] Write the reversible logic network in netlist format to file or STDOUT.

print [**-h**|**–help**] Shows a textual representation of a reversible logic network. The equations displayed are not Boolean equations.

read_pprm [**-h**|**–help**] ***file*** Read a positive-polarity reed-muller (PPRM) spec. Synthesis can only proceed if a valid pprm spec exists.

write_pprm ***file*** Writes the positive-polarity reed-muller (PPRM) spec in the network to file or STDOUT.

minterms_to_pprm [**-h**|**–help**] [***outfile***] ***-v var -m minterm*** Generates the PPRM coefficents from the minterms of a single-output function.

esop_to_pprm [**-h**|**–help**] [***outfile***] ***file*** Generates the PPRM coefficients from the ESOP representation of a multi-output function.

11.1 Netlist Format

The syntax of the netlist is as follows:

```
# This is a comment
.model <name>
.inputs <in1> <in2> ...
.gate <TOF, FRED, PER, RPER> <1, –, x, y> <1, –, x, y> ...
.gate <TOF, FRED, PER, RPER> <1, –, x, y> <1, –, x, y> ...
...
.end
```

TOF : n-bit general TOFOLLI gate.

FRED : n-bit general FREDKIN gate.

PER : n-bit general PERES gate.

RPER : n-bit general REVERSE-PERES gate.

1 : control.

– : no connection.

x : target.

y : second distinguishable target (only for PERES and REVERSE-PERES)

Note: See `relos/benchmarks/netlists` for examples specified in this format.

11.2 PPRM Format

The syntax of the PPRM is as follows:

```
# This is a comment
.model <name>
.inputs <in1> <in2> ...
.z <1, 0> <1, 0> ...
.p <1, –, 0><1, –, 0>... <1, –, 0><1, –, 0>...
.p <1, –, 0><1, –, 0>... <1, –, 0><1, –, 0>...
...
.end
```

z A bit vector indicating which inputs is a constant 0.

p A product term in a PPRM.

11.3 A PPRM Example

Consider the full-adder spec whose PPRM spec is given as follows:

d_{out} = d XOR ab XOR bc XOR ac (carry bit)
c_{out} = a XOR b XOR c (sum bit)
b_{out} = a XOR b (propagate bit)
a_{out} = a (garbage bit)

The PPRM spec is then:

.model adder
.inputs d c b a
.z 1 0 0 0 (d is constant 0)
.p 0001 0111 (a is in c_{out}, b_{out}, a_{out})
.p 0010 0110 (b is in c_{out}, b_{out})
.p 0100 0100 (c is in c_{out})
.p 0011 1000 (ab is in d_{out})
.p 0101 1000 (ac is in d_{out})
.p 0110 1000 (bc is in d_{out})
.end

Note: See `relos/benchmarks/pprm` for examples specified in this format.

11.4 PPRM Generation of CSOP Functions

To generate the PPRM specification of a single-output function specificed in canonical sum of products form, use the *minterms_to_pprm* command. To generate the PPRM spec of $f(x_1, x_2, ..., x_n) = m(0, 2, 3, 7)$, say:
minterms_to_pprm -v 3 -m 0 -m 2 -m 3 -m 7

000 1
100 1
110 1
001 1
101 1

Thus, the PPRM is $f = 1$ XOR x_3 XOR x_2x_3 XOR x_1 XOR x_1x_3.

11.5 PPRM Generation of Arbitrary Reversible Functions

To generate the PPRM of a multi-output function, use the *esop_to_pprm* command. A tool called EXORCISM is used which takes an aribtrary reversible function in PLA or BLIF format and converts it into a XOR sum of products (ESOP). By using the relationship, x bar = x XOR 1, it is possible to convert the ESOP into PPRM.

Note: It is the user's responsibility to ensure that the function being converted into PPRM format is a reversible function.

12 Netlist Simulation

The following commands are used in simulation of reversible logic networks.

simulate [**-o** ***outfile***] [**-h**|**–help**] ***file*** Simulate the reversible logic network with vectors supplied in file.

Note: See `relos/benchmarks/vectors` for examples on how to supply the simulation vectors. The output (and only certain bits of the output) are valid only when the garbage bits on the input have the proper assignment. It is assumed the user is familiar with this.

TODO: It would be nice to specify which bits are garbage bits in the netlist format specification and then, those bits should not get output during simulation.

UPDATE: J. Donald has thought out a scheme to do this. It involves allowing "free" substitutions that do not increase the gate count and thus bring more potential solutions to the front of the priority queue. As I've reasoned it out so far, however, the input function would have to be the reverse of the original input function.

13 Network Synthesis

The following commands are used in synthesis of reversible logic networks.

syn [**-q**|**–quit**] [**-h**|**–help**] [**-g**|**–greedy**] [**-e**|**–exhaustive**] [**-d**|**–depth** ***int***] [**-t**|**–time** ***int***] [**-s**|**–swap**] [**-f**|**–fredkin**] [**-p**|**–peres**] [**-b**|**–bigperes**] [**-x**|**–extra**] [**-y**|**–misc**] [**-m**|**–miniterms**] [**-c**|**–quantum**] [**-l**|**–limit** ***int***] Synthesize a reversible logic network given a valid PPRM specification.

Many new options were added in version 0.2. **-s**|**–swap**, **-f**|**–fredkin**, **-p**|**–peres** enable various gate types other than Toffoli gates. **-b**|**–bigperes** permits n-bit Perse and n-bit reverse-Peres gates.

-c|**–quantum** tells the algorithm to optimize for quantum cost rather than gate count, and can only work if a cost table is loaded. **-l**|**–limit** prevents the synthesis from using any gates beyond a certain size.

-x|**–extra** enables the additional NOT substitution for any variable as described in Section 4.D of the journal paper, although this extra substitution apparently was not actually implemented in the original release version. **-y**|**–misc** enables a lightweight decision to base Toffoli candidate factors even on terms that do include the target variable. **-m**|**–miniterms** enables the miniterms substitution that creates many branches based on subsets of candidate factors, and so far is the only known way to synthesize a minimum cost Fredkin gate. *Note*:

Only the last of these three options has any effect on Fredkin gate substitutions.

Note: The algorithm currently implemented may not be to always synthesize a reversible logic network. Clearly, a lot of research needs to be done to be able to handle arbitrary specifications.

14 Quantum Cost Calculation

The following command is used in reading a quantum cost table from which the quantum cost of a reversible logic network is calculated. This table should be kept in sync with the table at `http://www.cs.uvic.ca/~maslov/definitions.html`

read_cost_table [-h|–help] ***file*** Read the quantum cost of Toffoli gates from the specified file.

In cost calculation, the minimal cost of an n-Toffoli gate is used (if several implementation of a gate are known) provided that the sum of size of the gate and garbage does not exceed the number of lines in the circuit. Take a look at `relos/benchmarks/quantum_cost.txt` for the syntax of a sample file. The first column indicates gate size, the second column indicates garbage, the third column indicates cost. The cost of an n-Fredkin gate is just the cost of an n-Toffoli gate +2. An exception to this is the 3-Fredkin gate, which has a cost of 5.

The standard 3-bit Peres gate has a quantum cost of 4. If n-Peres gates are specified, Peres gates and reverse-Peres gates of a size greater than 3 are each measured with the pessimistic cost of an n-Toffoli plus an $(n-1)$-Toffoli.

Note: It is the user's responsibility to ensure correct syntax. Three columns with each row containing three integers. No other tokens are allowed.

15 Useful Scripts

The `relos/tools` directory contains some useful scripts.

1. gen_pprmcir
2. gen_shift
3. PPRMbenchmarkgen.pl
4. PPRMcirgen.pl
5. PPRMsyn.pl
6. parse_log

gen_pprmcir - This program will generate all reversible functions of three variables. 40,320 files will be generated so run this program in a separate directory.

gen_shift - This program generates *correct* PPRM specifications for the shifter functions.

```
Usage: gen_shift <num_bits>
```

PPRMbenchmarkgen.pl - This scripts generates random PPRM circuits. It is more robust than *PPRMcirgen.pl*.

```
Usage: PPRMbenchmarkgen.pl <options>
 --var   Number of variables (default is 3).
 --depth Max depth of circuit (default is 10).
 --num   Number of circuits to generate (default is 1).
 --nct   Use NOT, CNOT, 2-Toffoli gates.
 --help  Show this message.
```

PPRMcirgen.pl - This script generates random PPRM circuits of n variables.

```
Usage: PPRMcirgen.pl <options>
 --var   Number of variables.
 --num   Number of circuits to generate.
 --dir   Name of directory to put the circuits in.
 --help  Show this message.
```

PPRMsyn.pl - This script synthesizes PPRM circuits and logs the result in *relos.log*.

```
Usage: PPRMsyn.pl <options>
 --dir   Directory containing benchmarks.
 --help  Show this message.
```

parse_log - This program reads a `relos.log` file generated by *PPRMsyn.pl* and reports the distribution of circuit sizes.

```
Usage: parse_log <filename>
```

16 Contributing Your Source Code

If you want to contribute your source code to *ReLoS* and/or make improvement, add new features, etc., please contact the authors. This is so we do not have multiple versions floating around.