Readme File for MALS

Dept. of Electrical Engineering, Princeton University

March 30, 2006

1 ABOUT MALS

MALS is a tool that does majority logic synthesis. It can be used for circuits made of quantum-cellular automata (QCA), tunneling phase logic (TPL), and single-electron tunneling (SET) based nanotechnologies. It is developed on top of SIS and distributed freely for research purpose. PLEASE DO NOT RE-DISTRIBUTE TELS OR MALS in any other manner. It comes as-is with no warranties.

2 INSTALL & RUNNING MALS

2.1 INSTALL

Before installing, you should install student versions of CPLEX and AMPL (www.ampl.com) on your machine and add these two programs in your ${PATH} variable.

The tool has been tested on Red Hat Linux 8.0 and above only. The installation instructions are as follows:

1. Type 'tar zvxf sis-1.2.tar.gz' to uncompress the file.
2. Type 'cd sis-1.2; make' to compile everything
3. Add '${PATH}/sis-1.2/bin' to your ${PATH} variable

Note that you may have to modify the variables in the top level Makefile a little bit to get the tool to compile. If you are having some major problems, send a detailed report to one of the authors and they will try to get back to you.

Here are some problems you may encounter during compiling PROBLEMS:

1. Make sure /bin/make exists. Create a soft link if you have to.

2. In newer versions of linux (Red Hat 8.0 and above), you have to modify /usr/include/bits/dirent.h to get SIS to compile. Here is what the file looks like. Add "int d_namlen;" as shown below. If you don't, you will get a bunch of compile errors.

```
#ifndef _DIRENT_H
# error "Never use <bits/dirent.h> directly; include <dirent.h> instead."
#endif


struct dirent
  {
#ifndef __USE_FILE_OFFSET64
    __ino_t d_ino;
    __off_t d_off;
#else
    __ino64_t d_ino;
    __off64_t d_off;
#endif
    unsigned short int d_reclen;
    unsigned char d_type;
    int d_namlen; /* pallav, for sis to work */
    char d_name[256];           /* We must not include limits.h! */
  };
...
```

2.2 HELP

Run-time help can be obtained by typing "help" or "man" while in sis, which will list all of the available commands (including MALS commands). Help for each command can be obtained by typing "help command_name" or "man command_name" while in sis.

2.3 MALS COMMANDS

MALS provides the following new commands: *majsyn, thbuffer, majcheck, invcheck, majfactor*

usages:

- **majsyn** [-n int1] [-h]

  -n int1: Initial collapse fanin restriction set to int1 (default no collapsing).

  This command does majority logic synthesis on a given Boolean network.

- **majsyn_red_remove** [-n int1] [-h]

  -n int1: Initial collapse fanin restriction set to int1 (default no collapsing).

  This command does majority logic synthesis on a given Boolean network and produces an irredundant majority network.

- **thbuffer** [-h]

  -h: For help information.

  This command does technology mapping for RTD and Single-Electron Boxes based ciruits by inserting buffers to the circuit wherever needed. The circuit's opera tion is based on the four-phase or three-phase clock scheme for RTDs and Single- electron boxes respectively.

- **majsyn** [-h]

  check if the network is a majority network.

- **invcheck** [-h]

  Check how namy inverters exists inside of a given network (not including the primary inputs)

- **majfactor** [-n num1] [-p num2] [-r] [-h]

  Decomposing the nodes for a better majority network.

  -n num1: Assign the integer num1 to fanin restriction, default value is 3.

  -p num2: Consider the num2/100 portion of the candidate kernels (0¡= num2 ¡=100, default is 60).

  -r : Split each node in the network recursively until num1 is reached or no kernels (default is recursively.

  -h: For help information.

  This command uses new objective function to do factorization. It can be used as a preprocess procedure before doing majority logic synthesis.

2.4 INPUT to MALS

Input to MALS should be a combinational circuit. For example, the circuit can be written in *.blif* format. Preprocessing should be done before doing majoirty network synthesis. A script can be used to do the preprocessing, which uses the command targeted at programmable gates arrays (PGA'S): "x1_split -n 3". It decomposes the network such that no node in the network has more than three fanins.

2.5 EXAMPLE

First run SIS by typing command "sis". In SIS, first read in a nework by using some SIS commands, such as "read_blif xxx.blif". Preprocessing must be done before calling any MALS commands. After preprocessing, type MALS command "majsyn" to generate a valid majority network. The command "thbuffer" can then be called to perform technology mapping on the SET-based circuits, which has three-phase clocking.

One example is given below:

First given a preprocessing script: run_maj, which does optimization and decomposition on the network. The script file is written as following:

```
#### 3 - feasible network
#### algebraic decomposition
collapse
sweep
eliminate5
simplify -mnocomp -d
resub -a -d*
gkx -abt30
resub -a -d*
sweep
gcx -bt30
resub -a -d*
sweep
gkx -abt10
resub -a -d*
sweep
gcx -bt10
resub -a -d*
sweep
gkx -ab
resub -a -d*
sweep
gcx -b
resub -a -d*
sweep
eliminate0
xl_split -n3
```

The example is running as following:

```
[sispurnea mlex]$ sis
UC Berkeley, SIS 1.3 (compiled 13-Jul-04 at 3:15 PM)
sis> read_blif majority.blif /* read in the network */
sis> print /* print out the network */
f = h'
h = a'b'd' + a'c'd' + a'd'e' + b'c'd' + b'd'e' + c'd'e'
sis> source run_maj /* preprocessing the network */
sis> majsyn /* do majority network synthesis */
sis> print /* print out the majority network */
f = [29] + [4]
[2] = bc + be + ce
[4] = [5] + d
[5] = [20]e
[20] = bc
[29] = [2]a
sis> print_stats /* print out the status of the majority network */
traffic_cl pi = 5 po = 1 nodes = 6 latches = 0 lits(sop) = 16
sis> majcheck /* check if the resulting network is a valid majority network */
Passed check
sis> invcheck
/* check if there is any inverters inside of the network, not considering inverters
on the primary inputs */
NO inverters inside network "traffic_cl".
sis> thbuffer /* do technology mapping by inserting buffers */
sis> print /* print out the network */
f = [35] + [4]
[2] = bc + be + ce
[4] = [33] + [5]
[5] = [20][34]
[20] = bc
[29] = [2][31]
[31] = a
[32] = d
[33] = [32]
[34] = e
[35] = [29]
sis> print_stats /* print out the status of the network */
traffic_cl pi = 5 po = 1 nodes = 11 latches = 0 lits(sop) = 21
sis>
```

3 SCRIPTS TO RUN MALS

The preprocessing script can be changed anyway as you like. However, you have to make sure the input network to the command 'majsyn' is a 3-feasible network. This can be done add the command "*xl_split* $-n$ 3" at the end of the

script file, or using some other factorization command, such as "*majfactor -n 3 -r*". An example script file *maj2.syn* is given for reference only.

4 MORE INFORMATION

For more information, refer to our paper:

Rui Zhang, Pallav Gupta, and Niraj K. Jha, "Synthesis of Majority and Minority Networks and Its applications to QCA, TPL and SET Based Nanotechnologies," in *Proc. Int. Conf. VLSI Design*, pp. 229-234, Jan. 2005.

Please reference the above paper in work which has significant use of MALS.

5 CONTACTS

All bugs and problems should be mailed to pgupta@princeton.edu or rzhang@princeton.edu. Please send a detailed description of the bugs and problems.