NAME

misII - Multiple-level Combinational Logic Optimization Program

SYNOPSIS

misII [options] [file]

DESCRIPTION

MIS is an algorithmic multi-level logic synthesis and minimization program. MIS starts from a description of a combinational logic macro-cell and produces an optimized set of logic equations which preserves the input-output behavior of the macro-cell. The program includes algorithms for minimizing the area required to implement the logic equations, and a technology mapping step to map a network into a user specified cell library.

MIS can be run in interactive mode accepting commands from the user, or in batch mode reading commands from a file or from the command line. If no options are given on the command line, MIS will enter interactive mode. Otherwise, MIS will enter a batch mode. When running in batch mode, MIS reads its input from the file given on the command line, or from standard input if no filename is given; output is directed to standard output, unless -o is used to specify an output filename.

When MIS starts-up, it performs an initial source of the file ~octtools/lib/misII/lib/.misrc. Typically this defines a standard set of aliases for various commands. The last command in this file is a 'source ~/.misrc' to allow the user to define his own aliases at startup.

OPTIONS

-c cmdline

Run MIS in batch mode, and execute cmdline. Multiple commands are separated with semicolons.

-f script

Run MIS in the batch mode, and execute commands from the file script.

-t type

Specifies the type of the input when running in batch mode. The legal input types are: Berkeley PLA Format (-t pla), Berkeley Logic Interchange Format (-t blif), eqntott(1CAD)-format equation input (-t eqn), Oct logic view (-t oct), and suppress input (-t none). The default input type is blif.

-T type

Specifies the type of the output when running in batch mode. The legal output types are: Berkeley PLA Format (-T pla), Berkeley Logic Interchange Format (-T blif), bdnet(1CAD)-format net-list (-T bdnet), Oct Logic View (-T oct), and suppress output (-T none). The default output type is blif.

-o file

Specifies the output file when running in batch mode. For Oct output, this is a string of the form cell:view. The default for the output is the standard output.

-s

Suppress sourcing the commands from the standard startup script (~octtools/lib/misII/lib/.misrc).

-x

For batch mode operation, suppress reading an initial network, and suppress writing an output network. Equivalent to -t none -T none.

COMMAND SUMMARY

The commands are summarized below according to their function (network manipulation, input-output, network status, command interpreter, and miscellaneous).

Network Manipulation commands

act_map	do a mapping into "actel" blocks of pld gates
add_inverter	add inverters to the network to make all gates negative
atpg	perform combinational atpg using SAT approach
buffer_opt	inserts buffering trees for high fanout gates
collapse	collapse a network or a set of nodes
cspf_simplify	two-level minimization of each node using CSPF's
decomp	decompose a node into a set of nodes
eliminate	eliminates nodes whose value falls below a threshold
espresso	collapse network and minimize with a two-level minimizer
factor	determine a factored form for a node
gcx	extract common cubes from the network
gkx	extract common multiple-cube divisors from the network
invert	invert a node, and toggle the phase of all of its fanouts
map	technology mapping to find an implementation for the network
phase	phase assignment to minimize number of inverters
resub	perform resubstitution of a node into other nodes in the network
red_removal	perform redundancy removal via atpg
simplify	two-level minimization of each node
speed_up	speed up of nodes in the network
sweep	remove all inverters and buffers from the network
tech_decomp	decompose a network for technology mapping
wd	re-express a node using another node
xl_cover	global cover of nodes by "xilinx" blocks of pld gates
xl_k_decomp	Karp-Roth decomposition for mapping into "xilinx" gates
xl_merge	merge "xilinx" blocks
xl_partition	local cover of nodes by "xilinx" blocks of pld gates
xl_split	decompose a network (using routing complexity as cost)

Input-Output Commands:

read_blif	read a network in BLIF format
read_eqn	read a network in eqntott(1CAD) format
read_oct	read a network from an Oct Logic view
read_pla	read a network in PLA format
read_library	read a library description file
plot	do a plot of the network (only for X11)
set_delay	set delay parameters for primary inputs and outputs
write_bdnet	write the current (mapped) network in bdnet(1CAD) format
write_blif	write the current network in BLIF format
write_eqn	write the current network in eqntott(1CAD) equation format
write_lif	write the current network in LIF format
write_oct	write the current network into an Oct Logic view
write_pla	write the current network in PLA(5CAD) format

Network Status Commands:

constraints	print the delay constraints for a set of nodes
print	print logic function associated with a node
print_altname	print the short (and long) names for anode
print_delay	timing simulate a network and print results
print_factor	print the factored form associated with a node
print_gate	print the library information associated with a gate
print_io	print the fanin and fanout of a node (or the network)
print_kernel	print the kernels (and subkernels) of a set of functions
print_level	print the levels of a set of nodes
print_library	list the gates in the current library
print_stats	print statistics on a set of nodes
print_value	print the value of a set of nodes

Command Interpreter:

alias	provide an alias for a command
chng_name	switch between short and long forms for node names
echo	merely echo the arguments
fanout_alg	select a fanout optimization algorithm (to be used by map)
fanout_param	set some parameters for fanput algorithm (to be used by map)
help	provide on-line information on commands
quit	exit MIS
reset_name	rename all of the short names in the network
save	save a copy of the current executable
set	set an environment variable
source	execute commands from a file
time	provide a simple elapsed time value

unalias	remove the definition of an alias
undo	undo the result of the last command which changed the network
unset	unset an environment variable
usage	provide a dump of process statistics

MIScellaneous

bdsyn	special command used by bdsyn(1CAD)
simulate	logic simulation of the current network
sim_verify	verify networks equivalent via simulation
verify	verify equivalence of two networks

NODELIST ARGUMENTS

Most commands which take a node also take a list of nodes as an argument. This is referred to as a node-list in the documentation below. This list of nodes includes * to specify all nodes in the network, i() to specify the primary inputs of the network, o() to specify the primary outputs of the network, i(node) to specify the direct fanin of node, and o(node) to specify the direct fanout of node.

STANDARD ALIASES

When MIS starts, it executes commands from a system startup file (usually ~octtools/lib/misII/lib/.misrc). This defines a standard set of aliases, and then sources the file ~/.misrc to allow users to define their own set of aliases. The default alias set includes the following aliases which have proven useful. Note that many of the aliases are intended for compatibility with MIS Version #1.3.

Standard Aliases:

alias	command	description
ai	add_inverter	add inverters to a network to correct the phases
alt	print_altname	print both long and short names for a node
asb	resub -a	algebraic resubstitution
c	chng_name	toggle between long and short names
clp	collapse	collapse network
el	eliminate	eliminate nodes below a threshold
exit	quit	terminate program
gd	decomp -g	good decomposition (i.e., best kernel decomposition)
gf	factor -g	good factoring (i.e., best kernel factoring)
gp	phase -g	good phase assignment (i.e., more expensive)
inv	invert	invert a node keeping network function consistent
nts	print_stats	print network status (including factored form)
p	print	print sum-of-products form of a node
pat	print_delay -a	print node arrival times
pd	print_delay	print delay
pf	print_factor	print factored form of a node
pg	print_gate	print gate information for a node
pgc	print_gate -s	summarize gate information for the network

pio	print_io	print inputs and outputs of a node or the network
pk	print_kernel	print kernels of a node
plt	print_delay -l	print output loading for each node
pn	p -n	print nodes in 'negative' form
prt	print_delay -r	print node required times
ps	print_stats -f	print network status (including factored form)
psf	print_stats	print network status
pst	print_delay -s	print node slack times
pv	print_value	print node values
qd	decomp -q	quick decomposition (i.e., any kernel decomposition)
qf	factor -g	quick factoring (i.e., any kernel factoring)
qp	phase -q	quick phase (i.e., simple greedy algorithm)
re	read_eqn	read equations from a file
rl	read_blif	read blif network from a file
rlib	read_library	read library in genlib format
ro	read_oct	read network from an Oct view
rp	read_pla	read PLA in espresso format
rsn	reset_name	reset all short names starting from 'a'
sim	simulate	logic simulation on a network
sim0	simplify -d	quick minimization of a node (no don't cares)
sim1	simplify -m nocomp -d	complete minimization of a node (no don't cares)
sim2	simplify	single pass minimization with fanin DC-set
sim3	simplify -m nocomp	complete minimization with fanin DC-set
so	source	source a script file
sw	sweep	remove buffers, inverters from a network
td	tech_decomp	decompose network to meet AND/OR constraints
u	undo	undo last command which changed network
wb	write_bdnet	write mapped network in BDNET format
wl	write_blif	write network in blif format
wp	write_pla	write network in Espresso PLA format
wo	write_oct	write network as an Oct view

DETAILED COMMAND DESCRIPTIONS

add_inverter

Add inverters into the network wherever needed to make each signal (including the primary inputs) used only in its negative form. After this command, every literal in a node is in the negative form. This is the appropriate starting point for the technology mapping step.

alias [name [string]

unalias name ...

The alias command, if given no arguments, will print the definition of all current aliases. If given a single argument, it will print the definition of that alias (if any). If given two arguments, then the keyword **name** becomes an alias for the command string **string**, replacing any other alias with the same

name. The unalias command removes the definition of an alias.

atpg [-a] [-r] [-n] [-b] [-t] [-f] [-c] [-i] [-d] [-l]

Perform combinational test generation using random test generation, deterministic test generation, and fault simulation. Random test generation is done by using parallel fault simulation. Deterministic test generation is accomplished by using Boolean satisfiability.

The -l option sets the give up level of the rtg. The current default is 1 which means that the atpg will exit the rtg if a parallel simulation of 32 patterns did not detect any faults in the fault list. A value of 2 would set it to 64 and so on.

The -a option removes active clauses. Active clauses are used to accelerate the deterministic test generator by minimizing the search space of the boolean satisfier. Removing active clauses will dramatically degrade the performance of the dtg.

The -r option causes the atpg not to do random test pattern generation.

The -n option removes non-local implications. Non-local implications are done for both good and bad circuits. The boolean satisfier also does non-local implications by binding the variables and then noting the direct implications.

The -t option traces through the test generation process.

The -T option traces through the Boolean satisfy process.

The -f option causes the atpg not to do fault simulation after each deterministic test generation. It should be noted that as a self-check whenever fault simulation is being done, the fault begin tested for in the dtg is also given to the fault simulator.

The -c option removes critical clauses. Critical clauses forces gates on the active path to have inputs not on the active path to take on values that will allow the fault to be propagated.

The -i option removes additional critical clauses. These critical clauses involve two signals that both belong to the transitive fanout of the fault site.

The -d option adds dominator clauses. The current implementation does not include the improved improved unique sensitization procedure that includes multiple gates.

red_removal

Perform combinational redundancy removal using random test generation, deterministic test generation, and fault simulation. Random test generation is done by using parallel fault simulation. Deterministic test generation is accomplished by using Boolean satisfiability.

Options used are -acni -l 20

bdsyn

A special command exported for use by bdsyn(1). Not for general use.

chng_name

Toggles the network between long-name mode (user supplied names) and short-name mode (automatically generated single- character names).

collapse [n1] [n2]

Collapse nodes in the network. With no arguments, the entire network is collapsed into a single-level of functions (i.e., two-level form). Each output will be expressed in terms of the primary inputs.

Given a single node, that function is collapsed until it is represented entirely in terms of primary

inputs.

Given two arguments, it is assumed that the second node is a fanin of the first node. In this case, this dependency is removed (the first node is expressed without the second node as a fanin).

Please note that this command negates any mapping that may have been done at an earlier time.

Caution should be taken when collapsing network to two levels because the two level representation may be too large. The alternative is to use eliminate (selective collapse). Refer to eliminate for the details.

constraints [node_1] ... [node_n]

Print the values of the various delay constraints for the nodes in the argument list, which must be either inputs or outputs. Also prints the default values of the default delay parameters for the network. Used to check the values set by set_delay.

cspf_simplify [-d] [-i <num>[:<num>] [-m method] [-f filter]

Simplify each node in the network using method with a subset of the observability and satisfiability don't-cares generated at that node. The observability don't care subset is computed using the compatible set of permissible functions (CSPFs). An ordering is given to the nodes of the network and CSPFs for the nodes are computed according to that ordering. Each node is simplified using its CSPF and an appropriate satisfiability don't care subset.

method specifies the algorithm used to minimize the nodes. snocomp (default) invokes a single pass minimization procedure that does not compute the complete offset. nocomp invokes the full minimization procedure (ala ESPRESSO) but does not compute the complete offset. dcsimp invokes single pass tautology based minimizer.

dctype specifies how the don't-care set is generated. This option is not used at this point.filter specifies how the don't-care set is filtered. This option is not used at this point.

decomp [-gqd] [node-list]

Decompose all the nodes in the node-list. If the node-list is not specified, all the nodes in the current network will be decomposed. Decompostion will factor nodes and make the divisor a new node within the network, re-expressing other nodes in terms of this newly introduced node. It is one of the transforms used to break down large functions into smaller pieces, usually at the cost of introducing a few more literals.

If the -q option (the default) is specified, the **quick decomp** algorithm is used which extracts out an **arbitrary** kernel successively. Because of the fast algorithm for generating an arbitrary kernel, decomp -q is very fast compared with the decomp -g. In most cases, the result is very close. This command is recommanded at the early phase of the optimization.

If the -g option is specified, the **good decomp** algorithm is used which successively extracts out the **best kernel** until the function is factor free, and apply the same algorithm to all the kernels just extracted. This operation will give the best **algebraic** decomposition for the nodes. But, since it generates all the kernels at each step, it takes more CPU time. In general, decomp -q should be used in the early stage of the optimization. Only at the end of the optimization, should decomp -g be used.

If the -d option is specified, the disjoint decomposition is preformed. Currently, the disjoint decomposition is limited to the following simple algorithm: It partitions the cubes into sets of cubes having disjoint variable support, creates one node for each partition, and a node (the root of the decomposition) which is the OR of all the partitions.

echo args ...

Echoes the arguments to standard output.

eliminate [-l limit] thresh

Eliminate all the nodes in the network whose value is less than or equal to thresh. The value of a node represents the number of literals saved in the literal count for the network by leaving the node in the network. If the value is less than (or equal to) the threshold, the node will be eliminated by collapsing the node into each of its fanouts. A primary input or a primary output will not be eliminated.

The value of the node is approximated based on the number of times the node is used in the factored form for each of its fanouts. Note that if a node is used only once, its value is always -1.

limit is used to control the maximum number of cubes in any node. The default is 1000. Using a very large limit may result in collapsing the network to two levels. In general, if the circuit is collapsible, the command collapse is more efficient.

espresso

Collapse the network into a PLA, minimize it using **espresso**, and put the result back into the multiple-level **nor-nor** form.

factor [-gq] node-list

Throw away the old factored forms and factor each node in the node-list, and store the factored forms with the nodes. If the -q option is specified, the **quick factor** algorithm is used to factor the node. If the -g option is specified, the **good factor** algorithm is used to factor the node.

fanout_all [-v #] alg_list

Activates selectively one or more fanout algorithms. For a list of fanout algorithms known to the system, use the -v 1 option. The algorithms activated are the ones specified in the list. One algorithm, nnooaallgg, is always active. A two_level is a fast, area efficient algorithm. The best results are obtained with fanout_all bottom_up lt_trees mixed_lt_trees.

Fanout optimization itself is performed using the mmaapp command.

fanout_param [-v #] fanout_alg [property value]

Changes the default parameter values associated with specific fanout algorithms. For a list of these parameters and their values, do not specify the last two arguments.

gcx [-bcdf] [-t threshold]

Extract common cubes from a network, and re-express the network in terms of these cubes, and in the process cutting down on the number of literals needed in the network.

The -b option chooses the best cube at each step when exa mining possible cubes to be extracted; otherwise, the more efficient **ping-pong** algorithm is used to find a good (but not necessarily the best) cube at each step.

The -c option uses the complement of each cube as well as the cube when dividing the new cube into the network.

The -f option uses the number of literals in the factored form for the network as a cost function for determining the best cube to be extracted.

The -t option sets a threshold such that only a cube with a value greater than the threshold will be

extracted. By default, the threshold is 0, so that all possible cube divisors are extracted.

The -d option enables a debugging mode which traces the execution of gcx.

gkx [-1abcdfo] [-t threshold]

Extract multiple-cube common divisors from the network. The -a option generates all kernels of all function in the network when building the kernel-intersection table. By default, only level-0 kernels are used.

The -b option chooses the best kernel intersection as the new factor at each step of the algorithm; this is done by enumerating and considering each possible kernel intersection, and choosing the best. By default, the more efficient **ping-pong** algorithm is used to find a good (but not necessarily the best) kernel intersection.

The -c option uses the new factor and its complement when attempting to introduce the new factor into the network.

The -d option enables debugging information which traces the execution of the kernel extract algorithm.

The -f option uses the number of literals in the factored form for the network as the cost function when determining the value of a kernel intersection; by default, the number of literals in the sum-of-products form for the network is used.

The -o option allows for overlapping factors.

The -t option sets a threshold such that divisors are extracted only while their value exceeds the threshold. By default, the threshold is 0 so that all possible multiple-cube divisors are extracted from the network.

The -1 option performs only a single pass over the network. By default, the kernel extract algorithm is iterated while there are still divisors whose value exceeds the given threshold.

help [-a] [command]

Given no arguments, help prints a list of all commands known to the command interpreter. The -a option provides a list of all debugging commands, which by convention begin with an underscore. If a command name is given, detailed information for that command will be provided.

invert node-list

Invert each node in the node-list. It is used when the complement of a node is to be implemented. Note that it does not change the logic function of the current Boolean network, but will have an effect on the structure of the network.

map [-i] [-f #] [-m #] [-n #] [-f] [-a] [-r] [-s] [-v #]

Perform a technology mapping on the current network. A library must be read using the 'read_library' command before mapping can be performed.

The -m option controls the mode used for the technology mapping. A mode of '0' (the default) minimizes the area of the resulting circuit. A mode of '1' minimizes the delay of the resulting circuit (without regard to the total area). This does not take into account the actual load values in the mapped network. An intermediate value uses as cost function a linear combination of the two, and can be used to explore the area-delay tradeoff. A value of '2' minimizes the delay on an estimate of the critical path obtained from a trivial 2-input nand mapping, and the area elsewhere.

The -f option enables the fanout optimizer. The fanout optimizer inserts inverters at multiple fanout points to reduce the delay through the circuit.

The -a option recovers unnecessary buffers inserted by the fanout optimizer. This is just an area optimization. It does not increase the delay.

The -n option allows the access to the new tree mapper. With area mode performs worse than -m 0. With '1.0', it optimizes for delay only, and with an intermediate value, produces an intermediate result. The minimum delay mode performs better than -m 1. Specifying -n .9 provides a circuit with equal or less delay as -n 1.0 does, but for less area and more cpu time. Unfortunately, the new tree mapper has not been integrated with the fanout optimizer yet, so the effect of both optimizations do not always add up.

The -f option controls the internal fanout handling. The value is interpreted as a four bit mask: [b0,b1,b2,b3], where b0 is the least significant bit of the mask. For example, a value of 3 means that b0 and b1 are set, b2 and b3 unset. 'b2' is not currently used. Setting 'b3' allows the mapper to use gates with internal fanout, like XOR's or MUX'es. Setting 'b0' enables a heuristics used in minimum area mode to approximate the cost of a tree rooted at a multiple fanout point. The heuristics estimates the area of the tree by dividing the actual area by the number of fanouts. If 'b0' is not set, the area of the tree is estimated to be 0. Setting 'b3' is ineffective in area mode if 'b0' is not set. Finally setting 'b1' enables the mapper to introduce overlaps between gates (i.e. to duplicate logic). This wastes a lot of area in delay mode. The default value is 11 in area mode (i.e. b0, b1, b3 set) and 0 in delay mode.

The -i option disables the 'inverter-at-branch-point' heuristic. Intended for experimentation with different mapping heuristics.

If -r is given (**raw mode**), then the network must already be either 1- and 2-input NAND gates, or 1- and 2-input NOR gates, depending on whether a NAND-library, or a NOR-library was specified when the library was originally read. If -r is not given, appropriate commands are inserted to transform the network into the correct format.

The -s option prints brief statistics on the mapping results.

The -v options is for development use, and provides debugging information of varying degrees as the mapping proceeds.

phase_assign [-qgst] [-r n]

Decide for each node whether to implement the node of its complement in order to reduce the total cost of the network. If the network is mapped, the cost is the total area and the network will be kept mapped, otherwise, the cost is the total number of inverters in the network assuming all the inverters are already in the network. At the end, all the necessary inverters are inserted into the network and all the extra inverters are removed from the network.

The default algorithm, which can also be specified by -q option, is a greedy algorithm called **quick phase**. The -g option uses the Kernighan-Lin type algorithm called **good phase**. The -s option chooses the Simulated Annealing algorithm. The -r n chooses a random-greedy algorithm which does a random assignment first and then uses a greedy approach to get to a local minimum, and iterate n times. If -t option is specified, some tracing information is printed as the assignment proceeds.

plot [-n name] [-k]

The plot command creates a window with a abstract representation of the network, at its current level of optimization, labeling all nodes, including primary inputs and outputs. Vectors are used to show relationships between the various nodes.

The -n option gives the graph a name which the user may refer to in future uses of the plot command. If a graph by the given name already exists then the plot will appear in that window, otherwise a new window is created. If a name is not specified then a NULL name, or the name of the data file which was read is defaulted to.

The -k option "kills" (removes) a plot of a given name (using the -n option), or the window with the plot of the current network if the -n option is omitted.

print [-n] [-d] node-list

Print all the nodes in the node-list in sum-of-product form.

If -n option is specified, the nodes are printed in the negative form. (i.e. a' + b' will be printed as (a + b)').

If -d option is specified, the nodes in the external don't care network are printed.

print_altname node-list

Print the alternate name of all the nodes in the node-list. If the current name mode is **short** (mis internal name), the alternate name will be the **long** name (user specified name) and vice-versa.

print_delay [-alrs] [-m model] [-p n] n1 n2 ...

Do a delay trace on the network depending on the specified "model" and print the delay data. Without any arguments the routine will use the "library" model which assumes that the network is mapped and will print the arrival times, required times and the slack for all the nodes in the network.

Specifying an optional node list (n1 n2 ...) will print the delay data only for the specified nodes.

The user can selectively have portions of the delay data printed. The option -a will cause the arrival times to be printed. The option -r will cause the required times to be printed. The option -s will cause the slacks at nodes to be reported. The option -l will cause the load driven by the node to be printed.

The option -p n when specified with one of the options -[alrs] will print out the delay data so that the first n nodes with the most critical values are printed. The critical portion of the delay data is determined by the first of the options -[alrs] specified. Thus specifying -p n -[al] prints the nn nodes with the greatest arrival-time/load. For the -[rs] option the nodes with the smallest required-time/slack are printed.

The delay model can be specified by the -m option followed by one of the following keywords -- "unit" ,"unit-fanout", "library" or "mapped". Specifying "unit" results in delay being computed as 1 unit per node in the network. "unit-fanout" adds an additional delay of 0.2 per fanout. If a library has been read in using the read_library command one can use more accurate models ,"mapped" and "library", by using data stored in the library. Using the model "library" assumes that the network has been mapped. The "mapped" model does not make this assumption and will do a mapping of the nodes on an individual basis to compute a delay model for use during the delay trace.

print_factor node-list

Print all the nodes in the node-list in the **factored** form. If a node has not beed factored, **factor-q** will be used to factor the node.

print_gate [-ps] node-list

Prints the information provided in the library for the list of mapped gates.

The -p option prints the pin information of the gates.

The -s option prints summary of the gates and their area.

print_io [-d] [node-list]

Print both fanin and fanout list for each node in the node-list. Absence of node-list implies all the

nodes in the current network.

if the -d option is specified, the nodes in the external don't care network are considered.

print_kernel [-ass] node-list

Print kernels and corresponding co-kernels of all the nodes in the node-list. If -a option (default) is specified, all kernels, including the function itself if it is a kernel, are printed. If -s option is specified, only the sub-kernels are printed.

print_level [-l] [-c] [-m model] [-t value]

Prints the nodes of the network according to their level. The primary inputs are assigned level 0, and the level of a node is the length of the longest path from it to a primary input. The -l options prints only the number of levels in the network.

If the -c option is specified, only critical nodes are printed according to their level. The delay trace is done according to the -m model option (default is the unit-fanout model) and all the nodes with a slack within a -t value of the smallest slack are considered to be critical.

print_library [function_string]

Print the contents of the current library. If the optional string is given, only the gates with the same logic function as the string will be printed. **function_string** is in the format of read_eqn. For example, print_library "f=!(a*b);" will print all gates with a nand2 logic function.

print_stats [-f] [-d]

Print the current network status, which includes the network name, number of primary inputs (pi), number of primary outputs (po), number of nodes (nodes), and number of literals in the sum-of-product form (lits(sop)).

If -f option is specified, the number of literals in the factored form (lits(fac)) is computed. This could be slow when the factored form for some network takes too long to generate.

If -d option is specified, the statistics of the external don't care network is printed.

print_value [-a] [-d] [-p n] node-list

Print the value of all the nodes in the node-list. The value is currently defined as the number of literals increased if the node were eliminated from the network. Since the value of a node depends on the particular factored form of the node and its fanouts, all the nodes which don't have factored forms will be factored using **factor_q**.

The -a option prints the values in ascending order.

The -d option prints the values in descending order.

The -p takes an argument n, and directs print_value to only print the top n values.

quit

Stop the program. Does not save the current network before exiting.

read_blif [-a] filename

Read in a network from the file filename which is assumed to be in **blif** format. The network name is given by the **.model** statement in the file. If a **.model** is not given, the network name is the filename

with any trailing extension removed.

Usual filename conventions apply: - (or no filename) stands for standard input, and tilde-expansion is performed on the filename.

Normal operation is to replace the current network with a new network. The -a option specifies that the new network should be appended to the current network. Functions are associated between the two networks using the long names of each network. Name conflicts (where two functions attempt to define the same name) generate warning messages and are resolved by renaming the signal from the new network.

The -s option, though accepted, has no effect on read_blif, and is instead reserved for the read_pla command.

read_eqn [-a] [filename]

Read a set of logic equations in the format expected by eqntott(1). Each equation becomes a node in the logic network.

INORDER and OUTORDER can be used to specify the primary inputs and primary outputs for the network. If neither is given, then primary inputs are inferred from signals which are not driven, and primary outputs are inferred from signals which do not have any fanout.

The equations are of the form "<signal> = <expr> ;". For reference, the equation format uses the operators:

()	grouping
!= (or ^)	exclusive-or
==	exclusive-nor
!	complement
& (or *)	boolean-and
| (or +)	boolean-or

As a simple extension to eqntott, juxtaposition of two operands stands for boolean-and, and ' used as a post-fix operator stands for complement. Hence, F = a*!b + c*!d ; and F = a b' + c d' ; represent the same equation.

Note that eqntott and read_eqn treat the intermediate nodes of a network slightly differently. read_eqn will not make an intermediate node a primary output unless it also appears in the OUTORDER list. Also, the resulting network is a multiple-level network with all of the intermediate signals preserved. Finally, eqntott is order-dependent in that it requires signals to be defined before they can be used again; read_eqn relaxes this condition.

The -a option specifies that the new network should be appended to the current network. Functions are associated between the two networks using the long names of each network. Name conflicts (where two functions attempt to define the same name) generate warning messages and are resolved by renaming the signal from the new network.

The -s option, though accepted, has no effect on read_eqn and is instead reserved for the read_pla command.

read_oct cell[:view]

Read in a network from the Oct facet `cell:view:contents'. If `view' is not specified, it defaults to `logic'. The network name is set to `cell:view'. Oct nets without names are given machine generated unique names. All primary inputs and output are named the same as the equivalent Oct formal terminals of the facet.

This operation replaces the current network with the new network.

read_pla [-a] [-s] filename

Read in an espresso-format PLA from the file filename (see espresso(5) for more information). The network name is derived from the filename with any trailing extension removed.

Usual filename conventions apply: - (or no filename) stands for standard input, and tilde-expansion is performed on the filename.

Normal operation is to replace the current network with a two-level network of NOR-gates (and inverters) which implements the PLA. This preserves the multiple-output nature of the PLA. The -s option replaces the current network with a single-level network of complex gates with the same logic functions as the PLA outputs. This makes each PLA output a separate single-output function.

The -a option specifies that the new network should be appended to the current network. Functions are associated between the two networks using the long names of each network. Name conflicts (where two functions attempt to define the same name) generate warning messages and are resolved by renaming the signal from the new network.

read__library [-ainr] filename

Read a MIS-format library for future technology mapping. The -a option appends the library to the current library; otherwise, any previous library is discarded. The -i flag suppress adding extra inverters to all of the primitives. Intend for debugging only. The -n flag requests that a library, if it is generated, be made using NAND gates rather than NOR gates. The -r flag reads a raw library format (given in BLIF) rather than the normal genlib format.

red__removal

Perform combinational redundancy removal using random test generation, deterministic test generation, and fault simulation. Random test generation is done by using parallel fault simulation. Deterministic test generation is accomplished by using Boolean satisfiability.

atpg command options used are -acni -l 20

reset__name [-ls]

Resets either the short-names (starting again from the single letter a) with the -l option, or the MIS-generated long-names (starting again from [0]) with the -s option.

resub [-a] [-b] [-d] [node-list]

Resubstitute each node in the node-list into all the nodes in the network. The resubstitution will try to use both the **positive** and **negative** phase of the node. If node-list is not specified, the resubstitution will be done for every node in the network and this operation will keep looping until no more changes of the network can be made. Note the difference between resub * and resub. The former will apply the resubstitution to each node only once.

The -a (default) option uses algebraic division when substitute one node into another. The division is performed on both the divisor and it's complement.

The -b option uses Boolean division when substituting one node into another. NOTE: Boolean resubstitution has not yet been implemented.

The -d option directs resub not to use the complement of a given in algebraic resubstitutions.

save filename

Save a copy of the current executable to a file which can be restarted. This can be used to freeze the current network or the current library for later optimization. When the executable 'filename' is executed, execution returns to the top-level of the command interpreter.

NOTE: The save command is very operating-system dependent and may not be implemented on your system. If this is the case then the save command is unusable on your system.

set [name] [value]

unset name .....

A variable environment is maintained by the command interpreter. The set command sets a variable to a particular value, and the unset command removes the definition of a variable. Set, if given no arguments, prints the definition of all variables.

Different commands use environment information for different purposes. The command interpreter makes use of the follow ing:

autoexec

Defines a command string to be automatically executed after every command processed by the command interpreter. Useful for things like timing commands, or tracing the progress of optimization.

MISout

Standard output (normally stdout) can be re-directed to a file by setting the variable misout.

MISerr

Standard error (normally stderr) can be re-directed to a file by setting the variable misout.

open_path

open_path (in analogy to the shell-variable PATH) is a list of colon-separated strings giving directories to be searched whenever a file is opened for read. Typically the current directory (.) is first in this list. The standard system library (typically ~cad/lib/misII/lib) is always implicitly appended to the current path. This provides a convenient short-hand mechanism for reaching standard library files. prompt defines the prompt string

set_delay [-a|r|l|d f] [-s f] [-W f] [-a f] [-r f] [-d f] [-l f]

[o1 o2 ... | i1 i2 ...]

Set various delay parameters for the inputs and outputs of a network. Capitalized options set defaults, lower-case options set the parameters for the nodes in nodelist, which is either a list of output nodes or a list of input nodes.

The option -a sets the default arrival time for primary inputs to f. The option -r sets the default required time for primary outputs to f. The -d option sets the default drive on a primary input to f, and the -l option sets the default load on primary outputs to f. The -s option sets the wire load per fanout to f. One can get more fancy and specify the wire loads for a given number of fanouts by specifying the -W option. With the _ith use of the -W option, the load for a gate driving _i outputs is set to the value f. There is no undoing of these settings in the current session.

The -a,, -d,, -l, and -dR options perform specific functions, but only on the node arguments given. The -a (-r) option sets the default arrival (required) time to f if the node list given is a list of primary inputs (outputs). The -d option sets the drive on each node in the list to f; if there is a non-primary-input in the list this is an error. The -l option sets the load on each node in the list to f; if there is a non-primary-output in the list this is an error.

simplify [-d][-i <num>[:<num>]] [-m method] [-f filter] [node-list]

Simplify each node in the node-list using **method** with the don't-care set generated according to **dctype**.

method specifies the algorithm used to minimize the nodes. **snocomp** (default) invokes a single pass minimization procedure that does not compute the complete offset. **nocomp** invokes the full minimization procedure (ala ESPRESSO) but does not compute the complete offset. **dcsimp** invokes single pass tautology based minimizer.

dctype specifies how the don't-care set is generated. default don't care set generated is a subset of the fanin don't care set. -d option specifies that no don't care set is used. -i m:n specifies that the fanin don't cares of nodes within mm levels of transitive fanin and nn levels of transitive fanout of these transitive fanin are to be generated.

filter specifies how the don't-care set is filtered. **exact** (default) uses the exact filter. **disj_sup** used the disjoint support filter.

simulate in1 in2 in3 ...

For the current implementation of the network, given a value ('0' or '1') for each of the primary inputs of the network, simulate prints the value produced at each of the primary outputs. The correspondence of the input values and the primary inputs can be determined by the order in which the primary inputs and outputs are printed using the write_eqn command.

For example, for a three-input AND gate, the command simulate 1 1 0 will produce a 0

sim_verify [-c root] [-n # pats] filename.blif

Verify that two networks are equivalent using random-pattern simulation. That is, generate a random input vector, simulate the logic network, and check that the outputs between the two networks agree. The first network is the current network, and a second network is read from the file 'filename.blif'; it must be a BLIF format file. (This restriction will be fixed when the command interpreter is expanded to handle multiple networks.)

-n gives the number of random patterns to simulate.

-c root generates a C program **root.c** to simulate each network, and verify that the outputs are equivalent. This is automatically compiled into the program **root** and run for the specified number of vectors. This is much faster than the default interpreted mode.

NOTE: the -c option to sim_verify has not yet been implemented.

source [-psx] filename

Read commands from a file. The -p option prints a prompt before reading each command, and the -x option echoes each command before it is executed. The -s silently ignores an attempt to execute commands from a nonexistent file. Eventually source will allow arguments with the standard $1, $2, ... convention.

speed_up [-m model] [-d #] [-w #] [-t #.#] [-i] [-c] [-t] [-vD]

node-list

Speed-up the nodes in the node-list. If no nodes are specified selects the nodes to be speeded-up in order to speed-up the entire network. The best decomposition seen so far is accepted (except with the -c flag).

The -m model option selects the delay model according to which the delay data is computed. The values allowed for model are "unit", "unit-fanout" and "mapped". the "unit" delay model counts the

level of the circuit as its delay. the "unit-fanout" model is intended to capture a technology independent model and it assigns a delay of 1 unit to each gate and 0.2 units to each fanout stem. the "mapped" delay model uses the delay data in the library to compute delay.

The -d # option selects the distance upto which the critical fanins are collapsed in order to do the speed-up. A fast value is 3, a good one is 6.

The -t #.# option determines which nodes are considered critical. The critical nodes are those with a slack within #.# of the most negative slack.

The -w # option selects between the area mode and the pure timing mode. A value of 0 selects pure-timing mode while a value of 1 will conserve as much area as possible.

The -i option specifies that only the initial 2-input NAND decomposition be carried out.

The -c option specifies that one pass be carried out. The new decomposition is always accepted, even if it results in a slower circuit.

The -t option displays the delay as the iterations progress.

The -v,-d option display debugging information.

Note: Invoking speed_up on a mapped network may cause the network to become unmapped.

sweep

Successively eliminate all the single-input nodes and constant nodes (0 or 1) from the currently network. NOTE: Successfully invoking a sweep command on a mapped network can possibly "unmap" the network.

tech_decomp [-a and-limit] [-o or-limit]

Decompose all the nodes in the current network into **and** gates or **or** gates or both depending on whether -a, -o, or both flags are specified. The fanins of **and** gates will be no more than and-limit and that of the **or** gates will be no more than or-limit. and-limit and or-limit, if specified, must be at least two. The default option is -a 2.

time

Prints the processor time used since the last time command, and the total processor time used since misII was started.

alias [name [string]]

unalias name .....

The alias command, if given no arguments, will print the definition of all current aliases. If given a single argument, it will print the definition of that alias (if any). If given two arguments, then the keyword **name** becomes an alias for the command string **string**, replacing any other alias with the same name. The unalias command removes the definition of an alias.

undo

A simple 1-level undo is supported. Reverts the network to its state before the last command which changed the network. Note that interrupting a command (with ^C) which changes the network uses up the 1-level of undo.

set [name] [value]

unset name ...

A variable environment is maintained by the command interpreter. The set command sets a variable to a particular value, and the unset command removes the definition of a variable. Set, if given no arguments, prints the definition of all variables.

Different commands use environment information for different purposes. The command interpreter makes use of the follow ing:

autoexec

Defines a command string to be automatically executed after every command processed by the command interpreter. Useful for things like timing commands, or tracing the progress of optimization.

MISout

Standard output (normally stdout) can be re-directed to a file by setting the variable misout.

MISerr

Standard error (normally stderr) can be re-directed to a file by setting the variable misout.

open_path

open_path (in analogy to the shell-variable PATH) is a list of colon-separated strings giving directories to be searched whenever a file is opened for read. Typically the current directory (.) is first in this list. The standard system library (typically ~cad/lib/misII/lib) is always implicitly appended to the current path. This provides a convenient short-hand mechanism for reaching standard library files.

prompt

defines the prompt string

usage

Prints a formatted-dump of processor-specific usage statistics. For Berkeley Unix, this includes all of the information in the getrusage() structure.

verify [-m method] [-v] file1 [file2]

Verify the Boolean equivalence of two networks. file11 is compared with the current network when file2 is not specified, otherwise, file1 is compared with file2. The input and output variables from two networks are associated by their names.

The -m option specifies the verification method. If 'method' is 'clp' (default), two networks are collapsed and compared as PLA's. If 'method' is 'bdd', the BDD's are contructed for both networks, and compared.

The -v option engages the "verbose" mode of verify, where the user can see the progress that verify is making.

wd [-c] node1 node2

The wd command (which stands for "weak division") is very similiar to resubstitution (resub command), except that instead of operating on the entire network, wd simply re-expresses node1 in terms of node2.

The -c directive further allows re-substitution in terms of the compliment of node2.

write_bdnet [filename]

Write the current network to file filename in the format fo a net-list defined by bdnet(1). This is

allowed only after the network has been mapped into a final implementation technology.

The environment variable OCT-CELL-PATH defines where the cell library is located. If a cell does not have a leading '~' or '/' in its name, then OCT-CELL-PATH is prepended to the filename.

The variable OCT-CELL-VIEW defines the viewname to be used if the cell does not have a ':' in its name to separate the cell name from the view name.

The variables OCT-TECHNOLOGY, OCT-VIEWTYPE, and OCT-EDITSTYLE define the technology, view-type, and edit-style properties for the Oct cell.

write_blif [-s] [-n] [filename]

Write the current network to file filename in the Berkeley Logic Interchange Format (**BLIF**).

The -s option uses the network short-names rather than the network long-names for the BLIF file. This can be used to encrypt the names of a net-list.

The -n option uses the net-list format of **BLIF** when a node has a gate implementation in the library.

write_eqn [-s] [filename]

The write_eqn command prints out the equations from the current network according to format specifications laid out in the documentation for read_eqn. Both primary inputs and outputs are indicated.

The -s option uses the network short-names rather than the network long-names for the output.

If filename is not specified the equations will be outputed to stdout, otherwise they will be written into the given file and may be read by **read_blif** at a later time.

write_lif [filename]

Write the current network to file filename in the Logic Interchange Format (**LIF**) defined for the "International Workshop on Logic Synthesis", Research Triangle Park, North Carolina, May, 1987.

write_oct [-m] cell[:view]

Write the current network to the Oct facet `cell:view:contents'. If `view' is not specified, it will default to `logic'.

If the `-m' flag is specified, the network is merged into an existing network. All of the logic elements and internal nets are ripped up and replaced with the new network. Oct net names are used to determine how to merge in the network, so if the net names at the interface of the logic are not defined the merge will fail.

The environment variable OCT-CELL-PATH defines where the cell library is located. If a cell does not have a leading '~' or '/' in its name, then OCT-CELL-PATH is prepended to the filename.

The variable OCT-CELL-VIEW defines the viewname to be used if the cell does not have a ':' in its name to separate the cell name from the view name.

The variables OCT-TECHNOLOGY, OCT-VIEWTYPE, and OCT-EDITSTYLE define the technology, view-type, and edit-style properties for the Oct facet.

write_pla [filename]

Write the current network to file filename in the Berkeley PLA Format. No optimization is done on the PLA.

xl_cover [-n numberr] [-h heuristic number]

For mapping onto xilinx architecture. The initial network should have all intermediate nodes with fanin less than or equal to number (Default is 5). Uses Mathony's binate covering algorithm to minimize the number of nodes in the network. Uses different heuristics to solve the covering problem. The heuristic number can be specified by -h option. Heuristic number can be 0, 1, 2 or 3:

- 0 (exact),
- 1 (Luciano's heuristic),
- 2 (For large examples),
- 3 (default)(automatically decides between 0 and 2)

xl_k_decomp [-n number] [-p node_name]

Uses Karp_Roth disjoint decomposition to decompose recursively nodes of the network having fanin greater than number to obtain nodes each having fanin of at most number. If -p node__name is specified,, only the node with name "node__name" is decomposed. Else, all the nodes that have fanin greater than "number" are decomposed. in the present implementation, the first bound set is selected and no attempt is made to have the best decomposition.

xl_merge [-f MAX_Fanin] [-c MAX_COMMon_Fanin] [-u MAX_UNION_Fanin]

For mapping onto Xilinx architecture. Selects nodes of the network that can be merged so as to minimize the number of nodes in the network. Solves an integer program, using package Lindo. In the end it lists the pairs of nodes that were merged.

MAX_FANIN is the limit on the fanin of a mergeable node(default = 4)

MAX_COMMON_FANIN is the limit on the common fanins of two mergeable nodes(default = 3)

MAX_UNION_FANIN is the limit on the union of the fanins of two mergeable nodes(default = 5) Does not change the network.

xl_partition [-n value]

Collapses nodes greedily using the number of nets as a criterion.

The -n option takes an argument, value, which sets the limit on fanins of the node.

xl_split [-n value] [-d]

Ensures that every node in the network has number of fanins less than or equal to value. Does so by using kernel extraction and AND-OR decomposition. -d debug facilities turned on.

FILES

~octtools/lib/misII/ex

~octtools/lib/misII/lib/.misrc

~octtools/lib/misII/lib/script

~octtools/lib/misII/lib/*

SEE ALSO

espresso(1CAD), espresso(5CAD), eqntott(1CAD),

~octtools/doc/blif.doc.

AUTHOR

Richard Rudell
Albert Wang
Rick Spickelmier (Oct read and write)