1 User manual

1.1 Input format

1.1.1 Circuit description file

A circuit description file is made up of four parts, one of which is optional. The ordering of these parts is fixed and can not be switched. At the end of the file the keyword *end* must be specified. Throughout the file, blanks, empty lines and *C*- like comments (over one or several lines) can be added anywhere. Keywords can be written in either lower or upper case. No keyword can be used as a name for either a gate or a line. When specifying an input line format, optional parameters will be placed in square brackets ([]). The order in which the input line parameters are presented is fixed and can not be changed.

The first part of the input file is the *clock declaration segment*. It begins with a *declaration line* with the keyword *clocks*. The declaration line is followed by lines with clock names, one name per line. This part is optional, since not all circuits use clocked elements.

The second, third and fourth segments must appear in every input file.

The second segment is the *gate declaration segment*. It begins with a *declaration line* with the keyword *gate*. In each of the following lines a gate (combinational or clocked) is declared, one gate per line. All user names containing more than seven characters will be truncated to include only their seven character prefix. The format of a line varies, depending on the type of the declared gate.

If the gate is of type *AND, OR, NAND, NOR, INV* or *XOR*, the format is

gate_name gate_type output_line input_lines [rise_delay] [fall_delay]

gate_name is the user specified name for the declared gate.

gate_type is the keyword (*and, or, nand, nor, inv, xor*) representing the type of the declared gate.

output_line is the user name for the line connected to the output of the declared name.

input_lines is a list of the user names for the lines connected to the inputs of the declared gate.

The *rise_delay* (*fall_delay*) field is optional and is used only when the *rise* (*fall*) delay of the gate is different than the default value for the gate type. A delay parameter is specified by the keyword *rise* (*fall*) followed by the delay value. If both parameters are specified, the *rise* parameter must come before the *fall* parameter.

If the gate is an *inverter*, the fan-in is one. If the gate is an *exclusive-OR*, we allow only two input lines. All other gates can have up to *max_fan* fan-in, where the default value for *max_fan* is six. If this parameter is changed, the whole system must be recompiled.

To represent branch points a gate of type *BUF* is used. The input format is

gate_name BUF input_line output_lines [rise_delay] [fall_delay]

The fields are the same as above, the difference begin that since there is only one input line and multiple output lines, the name of the lone input line comes before the names of the output lines. The limit on the number of output lines is *max_fan*.

When the gate type is *DFF*, the input format is

gate_name DFF clock_line [reset_line] [preset_line] d_line [q_line] [qbar_line] [rise_delay] [fall_delay] [setup_time] [hold_time]

gate_name is the user name for the gate.

clock_line is the name of the clock driving the gate. It must be a name of a clock that was previously declared in the *clock declaration segment*.

The *reset_line* (*preset_line*) field is optional and used to represent the name of the line connected to the *reset* (*preset*) input of the gate. The field is specified by the keyword *rl* (*pl*) followed by the line name.

d_line is the name of the line connected to the *d* input of the gate.

The *q_line* and *qbar_line* fields are optional and represent the name of the line connected to the *q* or *qbar* output of the gate. They are optional since one of these outputs might not be used, but since one of them is always used, at least on of these fields must be present. The field is specified by the keyword *q* (*qb*) followed by the line name.

The delay parameters *rise* and *fall* are the same as for combinational gates, but for clocked elements we have two additional parameters, the *setup* and *hold* requirements.

When the gate type is *JKFF*, the format is

gate_name JKFF clock_line [reset_line] [preset_line] j_line k_line [q_line] [qbar_line] [rise_delay] [fall_delay] [setup_time] [hold_time]

All fields are the same as for a *D flip flop*, the difference being in the type specified and that in place of a *d* line we specify the *j* and *k* lines.

The third segment is the *primary input segment*. It begins with a *declaration line* with the keywords *primary inputs*. This line is then followed by lines of the format

primary_input_gate primary_input_line

A primary input is modelled as a one output gate. The *primary_input_gate* is the name of the primary input and the *primary_input_line* is the name of the line connected to the primary input.

The fourth and final segment is the *primary output segment*. It begins with a *declaration line* with the keywords *primary outputs*. This line is then followed by lines of the format

primary_output_gate primary_output_line

A primary output is modelled as a one input gate. The *primary_output_gate* is the name of the primary output and the *primary_output_line* is the name of the line connected to the primary output.

This section is followed by the keyword *end* which specifies the end of the circuit description file.

An example circuit description:

```
/* fig 6.7 p. 186 */
gates
b1 buf a a1 a2
b2 buf c c1 c2
b3 buf i i1 i2
g1 nand i a2 b c1
g2 nor h d a1
g3 inv j c2
g4 nor k j e
g5 and f1 h i1
g6 and f2 i2 k
primary inputs
a a
b b
c c
d d
e e
primary outputs
f1 f1
f2 f2
end
```

1.1.2 Control description file

The control description file is used only by the good circuit simulator. It specifies the control information. It consists of three parts, the first of which is essential. The other two are optional. The first part consists of the input waveforms specified as a list of events. For each input the input *line name* is specified followed by the sequence of events within round brackets. Each event is specified as a time-value pair. Events are delimited by a ":". The input event list is followed by an integer which specifies the duration of simulation.

The first optional part consists of *initial conditions* followed by the reserved word *ic*. *Initial conditions* are values that the user can force some lines to have. They also represent the initial states in the case of sequential circuits. Each line is a *line name* followed by the initial value.

The second optional part consists of a list of line names whose values are to be printed in the *trace* mode. This can be specified by a line which begins with the reserve word *print* and is followed by *line names*.

The control description file must have the reserved word *end* at the end of the control description.

An example control description file:

```
s (0 1: 5 2: 10 0)
```

```
r (0 1: 5 2: 10 0)
20
ic
y 0
y1 0
y2 0
ybar 1
ybar1 1
ybar2 1
print y ybar
end
```

1.1.3 Fault list file

The fault list file is used by the test generator as well as the fault simulator.

For the test generator it specifies the faults for which tests are desired and for the fault simulator it specifies the faults to be simulated.

The file is a list of faults, one fault per line. Each line is thus the *line name* followed by the *stuck at value.*

An example fault list file:

```
a 0
a 1
b 1
c 0
c 1
d 1
e 0
g 0
h 0
h 1
```

1.2 System commands

This section lists the commands the system recognizes. For each command the command line format and the resulting output is specified.

```
collapse filename [> output_filename]
```

The command receives the name of a circuit description file as a parameter. It parses the circuit file and finds all non-dominating and dominating equivalent fault classes. The output is the equivalence table that was constructed and a list of all non-dominating classes followed by a list of all dominating classes. For each class we specify the class representative and if there is a dominating fault we specify that fault. The output is sent to the standard output device but can be redirected to a user specified output file.

`gen_tests circuit_file fault_list [step] [trace] [> output_filename]`

The command receives the name of a circuit description file and a name of a file containing a list of faults, one fault per line (the fault list format is *line_name stuck_at_value*, where *line_name* is the user name for the faulty line and *stuck_at_value* is either 0 or 1). If neither of the *step* or *trace* parameters are specified, the output is a list of test vectors for non-redundant faults, and if a fault is redundant, a message stating this fact is generated. If the *step* parameter is used, all actions are single stepped, informing the user of every action that is taking place. Before proceeding with any selected option, the selected option is presented, along with a list of all other options, and the user can either continue processing with the selected option, or change it. In the *trace* mode, all action are stated but there is no user interaction. Only one of these parameters can be used.

`min_set filename [> output_file_name]`

The command receives the name of a circuit description file. It's function is to generate a minimal test set for all non-redundant stuck at faults in the circuit. The generated set is minimal in the sense that if we discard any vec- tor in the set, some non-redundant fault will not be caught. The function parses the circuit and does fault collapsing. It then iterates over the fault list until this list is emptied. For each fault taken off the list the function first tries to generate a test vector. If one is found, fault simulation is also performed and all other caught faults that are on the fault list are taken off. If a test vector can not be generated, the fault is added to the redundant fault list. At the end of the test set generation, all redundant faults are listed.

`parse filename [> output_filename]`

The command receives the name of a circuit description file. It parses the file and creates the global data structures. The output is a printout of the global structures *gate_list[], line_list[], fan_table[][], line_x_reference[]* and *clk_table[]*. It also outputs the initial list of fault classes representatives.

`test_set filename [> output_filename]`

The command receives the name of a circuit description file. It parses the file, performs fault collapsing and generates tests for all non-redundant faults. The output is a list of test vectors for all non-redundant faults and a list of all redundant faults.

```
gsim circuit_file control_file [step] [trace [> output_filename] ]
```

The command receives the name of a circuit description file and the name of a file containing the control information for the simulation. The control_file should contain the primary input events, simulation duration and line values to be printed out in the *trace* mode. If neither the *step* or *trace* parameters are used only each event being simulated is printed out. If the *step* option is used, all actions are single stepped - the simulator halts after every event. Depending upon user selected options, information is displayed after every event. If the *trace* parameter is used, the simulator runs through the simulation, stating events processed and the final results. There is no user interaction in this mode. The redirection option can be used only with the *trace* parameter.

```
fsim circuit_file [fault_list_file] [step]
     [ trace [type_of_test [number_of_vectors]] [> output_filename] ]
```

The command receives the name of a circuit description file. The second argument is the name of a file with the faults to be simulated (the fault list format is *line_name stuck_at_value*, where *line_name* is the user name for the faulty line and *stuck_at_value* is either 0 or 1). If this argument is not specified, all stuck-at faults are simulated. If the *step* parameter is used, the simulator steps through every test vector simulated. The user can check any fault list and see the faults detected. The *step* mode also asks the user for the type of test. If the *trace* parameter is used, the *type_of_test* argument has to follow. This argument could be a file name consisting of the test vectors or the reserved words *exhaust*, *rand* or *wtd*. If the *type_of_test* is specified as *rand* or *wtd* then the *number_of_vectors* parameter should be specified which has to be a 2-digit number. (For e.g. if 4 random vectors are needed, specify 04 as the parameter).

Both *step* and *trace* cannot be used together and one of them has to be specified at all times.