HITEC/PROOFS User Instructions

The HITEC/PROOFS sequential circuit test generation and fault simulation package includes four preprocessing programs, a test generator, a fault simulator, and two shell script files, as described below:

- *do_proofs* - shell script file to generate a TEST.run file with options set for running the fault simulator only. All programs except for *level* look at the *TEST.run* file for their arguments.

- *do_hitec* - shell script file to generate a TEST.run file with options set for running the test generator (which invokes the fault simulator).

- *level* - preprocessing program to parse a circuit in the *.bench* file format; makes *.lev* and *.name* files. The *.lev* file is a levelized circuit description with testability measures, and the *.name* file is a gate-number-to-name translation.

- *faultlist* - preprocessing program to make a complete fault list for the circuit in the *.fault* file with the following format:

  [Gate number] [Input number] [Fault (s-a-0 or s-a-1)].

  Input number 0 is the gate output, and the gate inputs are numbered 1, 2, 3,... The fault can be either 0 or 1.

- *equiv* - preprocessing program to collapse and order the fault list into an *.eqf* file. Faults are ordered in a depth-first search from the primary outputs.

- *dominators* - preprocessing program to find the dominator lines for each node in the circuit and list them in the *.dom* files, which is used by testgen.

- *testgen* - the test generator; test generation results, test vectors, and a list of redundant faults are placed in the files *.grs, .atp*, and *.red*, respectively. The order of primary inputs in the *.atp* file is the same as that in the *.lev* and *.name* files.

- *faultsim* - the fault simulator; fault simulation results and a list of undetected faults are placed in the files *.frs* and *.ufl*, respectively. Detected faults are placed in the *.det* file. When run as a stand-alone program, faultsim requires a test vector file, *.vec*.

The test generator may also be run with dynamic compaction. In this case, a modified version of the above test generator is used, along with the dynamic compactor:

- *testgen_squeeze* - the test generator with dynamic compaction; test generation results, test vectors, and a list of redundant faults are placed in the files *.grs, .atp*, and *.red*, respectively. The order of primary inputs in the *.atp* file is the same as that in the *.lev* and *.name* files.

- *squeeze* - the dynamic compactor; fault simulation results and a list of undetected faults are placed in the files *.frs* and *.ufl*, respectively. Detected faults are placed in the *.det* file.

To run the test generator and fault simulator, first place all executables in a "bin" directory, and include the pathname of that directory in your search path. Then create a TEST.run file with one of the two script files provided. Next, run the preprocessing programs, followed by the fault simulator or test generator. Program invocations for the s298 circuit are as follows:

do_hitec s298 (or *do_proofs s298*)
level s298
faultlist
equiv
dominators (not needed if the test generator is not used)
testgen (or *testgen_squeeze* or *faultsim*)

The *TEST.run* file format is shown below:

simpic port number - port number for faultsim graphics (obsolete)
master host name - name of master host for graphics (obsolete)
bench file - ckt.bench (name of benchmark file -- level input)
lev file - ckt.lev (name of levelized circuit file -- level output)
vec file - ckt.vec (name of vector file -- faultsim input)
fault file - ckt.fault (name of fault file -- faultlist output)
equiv fault file - ckt.eqf (name of equivalent fault file -- equiv output)
undetected fault file - ckt.ufl (name of undetected fault file -- faultsim output)
faultsim host name - used with testgen (name of host for faultsim)
testgen host name - used with testgen (name of host for testgen)
name file - ckt.name (name of "name" file -- level output)
atp file - ckt.atp (test vector file -- testgen output)
faultsim result file - ckt.frs faultsim results
testgen result file - ckt.grs testgen results
sim gen port - port between faultsim and testgen
color - color graphics used if one
faultsim running - faultsim running if one
testgen running - testgen running if one
fault pic running - graphics running if one (obsolete)
read vec - read ckt.vec file before test generation if one
dominator file - ckt.dom (name of dominators file -- dominators output)
redundant fault file - ckt.red (name of file with untestable faults -- testgen output)
debug - debug value for testgen
backtrack limit - maximum number of backtracks allowed in testgen
state backtrack limit - maximum number of state backtracks allowed in testgen
time limit - maximum time allowed to generate a test in pass 1 (unit = 1/100 sec for HP, 1/60 sec for SUN)
scan stat file - name of file for scan statistics

An example circuit description is shown below (ISCAS89 s27 circuit):

4 inputs

```
# 1 outputs
# 3 D-type flipflops
# 2 inverters
# 8 gates (1 ANDs + 1 NANDs + 2 ORs + 4 NORs)

INPUT(G0)
INPUT(G1)
INPUT(G2)
INPUT(G3)
OUTPUT(G17)
G5 = DFF(G10)
G6 = DFF(G11)
G7 = DFF(G13)
G14 = NOT(G0)
G17 = NOT(G11)
G8 = AND(G14, G6)
G15 = OR(G12, G8)
G16 = OR(G3, G8)
G9 = NAND(G16, G15)
G10 = NOR(G14, G11)
G11 = NOR(G5, G9)
G12 = NOR(G1, G7)
G13 = NOR(G2, G12)
```

A test vector file (*.vec*) for the example circuit is shown next. The *.atp* file has the same format.

```
4 <- number of primary inputs
0110 <- forced value for each primary input
1111
0110
1111
1110
0101
0100
1111
0111
0111
0110
0101
1101
0111
1110
END <- end-of-file delimeter
```

The following component types are handled by HITEC and PROOFS:

INPUT
OUTPUT
DFF -- D flip flop
AND
NAND
OR
NOR
NOT -- inverter
BUF -- buffer, output = input
TIE1 -- line tied to 1
TIE0 -- line tied to 0

In addition, PROOFS handles the following component types also:

XOR -- exclusive-OR
XNOR -- exclusive-NOR
BUS -- bus, output goes to unknown if all inputs have Z value
BUS_GOHIGH -- bus, output goes to high if all inputs have Z value
BUS_GOLOW -- bus, output goes to low if all inputs have Z value
TRISTATE -- tristate, active low control signal
TRISTATEINV -- tristate with output inverted
TRISTATE1 -- tristate, active high control signal
MUX2 -- 2-input multiplexer

For the MUX2, TRISTATE, TRISTATEINV, and TRISTATE1 components, the control signal must be the *last* input in the list of inputs. In the multiplexer, the first input is selected when the control signal is 0, and the second input is selected when the control signal is 1. A bus component must have only tristate components as predecessors, and tristate components must have only bus components as successors. The bus output value is the value of the non-Z (high impedance) input. If there is a conflict (a one placed on one bus input and a zero placed on another bus input), the bus output value is unknown.

Last Updated: August 4, 1997
Send any questions to liz@uiuc.edu

Back to HITEC/PROOFS