Final Project Report

VLSI TESTING

KASI L.K. ANBUMONY

<u>TOPICS</u>

(1) LOGIC SIMULATOR
(2) HIERARCHICAL TO FLATTEN NETLIST CONVERSION
(3) SIMULATION TABLE GENERATION

-A Robust Logic Simulator using Dynamic Levelization Algorithm-

Section 1: Algorithm Implemented

Step 1: Read the input and output net names in a dynamically growing array, *netnameI* and *netnameO* respectively
Step 2: Assign values to the input name using another dynamically growing array, *netvalueI* and *netvalueO* respectively
Step 3: Inputs are assigned on the rule of 2^N, where N: number of inputs in the circuit
Step 4: Next netlists (or) statements in the Bench Circuit Description Language used in describing the logic are read one by one
Step 5: Compare the input nets of the given gate with *netnameI*. If all the input nets of the given gate are found in the netnameI, then read out the *netvalueI* to solve the logic using *"solve-user defined function"* else goto Step 7
Step 6: The new output net of the gate evaluated in Step 5 is appended to the *netnameI* and its value to *netvalueI*
Step 7: The netlist or statement which is not evaluated due to lack of data are then stored in another dynamically growing array-*unfinish*
Step 8: Proceed executing next statements in circuit file, sequentially.
Step 9: After every successful execution of given statement (or equivalently resolving the logic for that gate), the unfinished nets are visited once again due to the new condition of the circuit *(netnameI & netvalueI)* by calling the "*redo- user defined function*" (Dynamic Levelization)
Step 10: By the time the program executes the last statement in the netlist all other logics in the circuit are resolved.

Functions created in MATLAB and their purpose

(i)	logicsim: Main function which reads in the circuit file and simulates the circuit and stores the truth table of the given circuit in result.txt and CPU time in cputime.txt
(ii)	expand: If a given circuit description has hierarchy then this subroutine flattens the circuit and stores the netlist in Flat.txt
(iii)	solve: To resolve the logic of a given gate, given the gate name and gate nets. This function resolve gate with any number of inputs based on the associativity property.
(iv)	redo: Dynamic Levelization is implemented by means of feedback from the logicsim main function to redo function.

Section 2: User Information

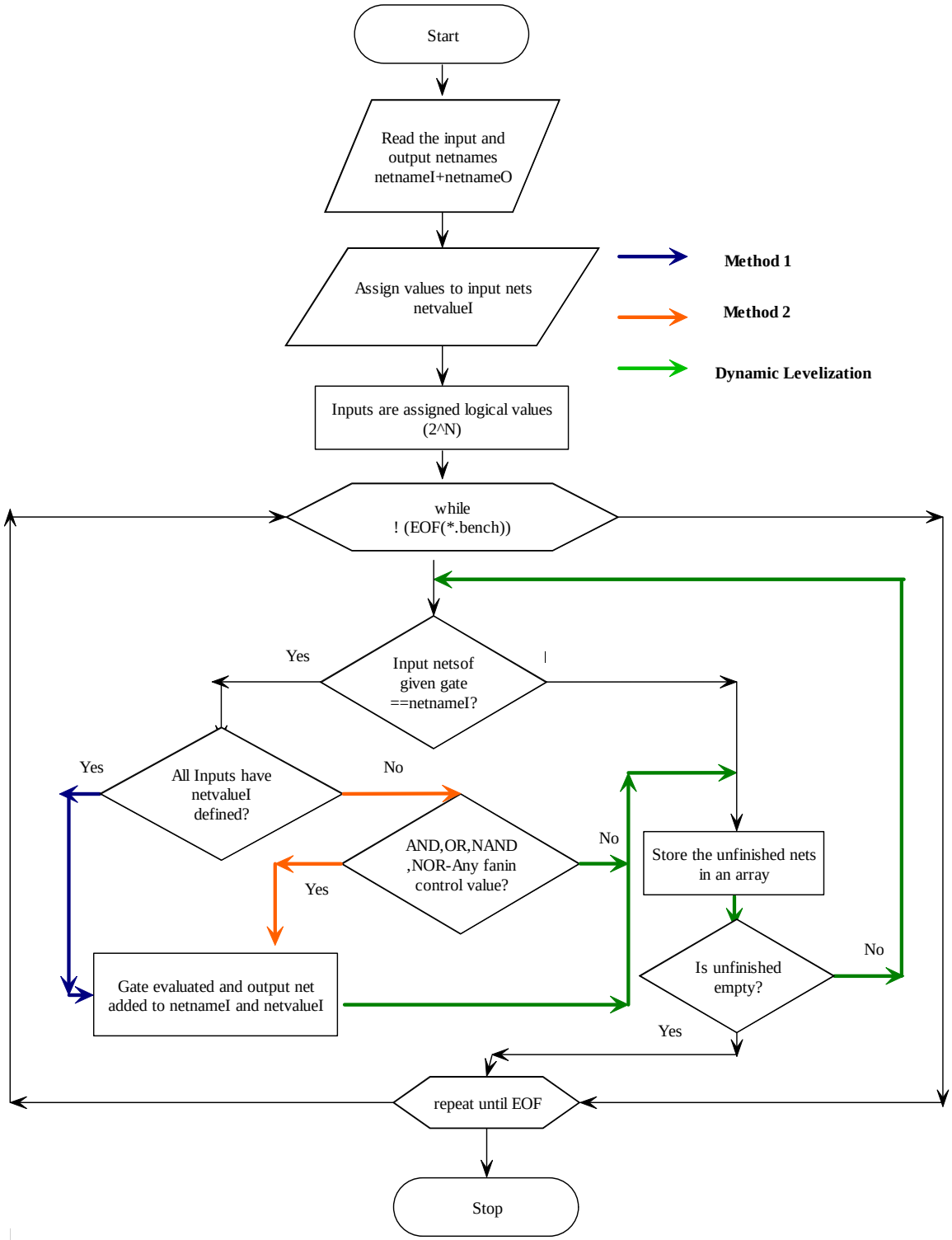
Bench file should be in the format given in http://www.fm.vslib.cz/~kes/asic/iscas/

*Type **logicsim** in MATLAB command window, will prompt for the filename

*It then asks whether the file is in hierarchy (or) flatten format.
-> Type 0 for hierarchical and 1 for Flatten.
This helps in flattening the circuit if the circuit is in hierarchical format

*Circuit need not be levelized because a kind of “Dynamic Levelization algorithm” has been implemented

Flowchart



Section 3: Adder verification and diagnosis

->Verification is produced in Appendix-A.

Section 4: Diagnosis method

Fault Dictionary

Fault	**Test syndrome** **t1 t_2 t_3 t_4**
No fault	0 0 0 0
a_0, b_0, d_0	0 0 0 1
a_1	1 0 0 0
b_1	0 0 1 0
c_0	0 1 0 0
c_1, d_1, e_1	1 0 1 0
e0	0 1 0 1

a_0 : Line a stuck- at-0
t_i = 0, if Ti passes
1, if Ti fails

<u>Algorithm</u>

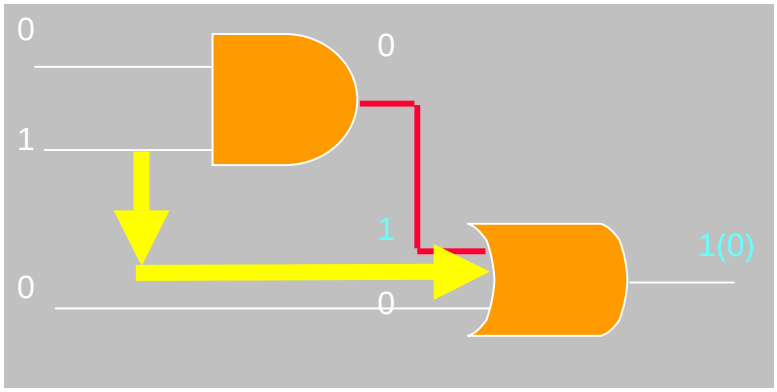
1. Creation of fault dictionary for the specific circuit exhaustively for all stuck-at faults and bridge faults
2. Compare the good circuit and faulty circuit response to generate a signature for all test vectors listed in "fault dictionary"

```
netname_good[] compare with netname_fault[]
netvalue_good[] compare with netvalue_fault[]
```

3. Compare the signature with the signatures in the fault dictionary
4. Any match, then higher probability of fault existing of that type or other faults close to that nets
5. Update the fault list dictionary, if any new faults for the same signature found
6. Use minimum hamming distance to find a closer fault, if no signature match and verify those nets for possible faults

Note: AUSIM and Atalanta can be used to generate Fault Dictionary for bridge faults and stuck-at faults respectively.

Wrong net connection can be predicted using backtracing. Here from the output pin, a correct value is justified in backwards. The point where in the justification fails, all the gates and paths found from that point to output pin is reported (fault cone 1).
Similar analysis is done on other test vectors wherein the combinational logic fails. Thus a
a set of fault cones are formed and they can be intersected to find the gate or path which is occurring number of times.



Sample Outputs
dict.txt

```
max=3
# # # 0 0 0 0
a0 b0 d0 0 0 0 1
a1 # # 1 0 0 0
b1 # # 0 0 1 0
c0 # # 0 1 0 0
c1 d1 e1 1 0 1 0
e0 # # 0 1 0 1
```

```
#True circuit
INPUT(1)
INPUT(2)
INPUT(3)
OUTPUT(5)
4=AND(1,2)
5=OR(4,3)
```

```
#faulty circuit
INPUT(1)
INPUT(2)
INPUT(3)
OUTPUT(5)
4=AND(1,2)
5=AND(4,3)
```

Possible matching faults
means no fault
'e0' '#' '#'

When response for the test vector set yields a signature status=[1 1 1 1];
Closely matched faults using minimum hamming
ans =

'c1'	'd1'	'e1'
'e0'	'#'	'#'

Section 5: Examination of Performance

(1) Plot CPU time vs. number of vectors for 4-bit adder circuit

Method 1: If the netlist is levelized with n gates and an average fanin of each gate is k, then complexity is given by O(kn)
Method 2: If one of the fanin's of a logic gate is a control value (i.e.) for an AND gate it is 0, then we need not wait for other fanin's for that gate to get stabilized. Saving by a factor of say "k1". Other gates: OR,NOR,NAND.

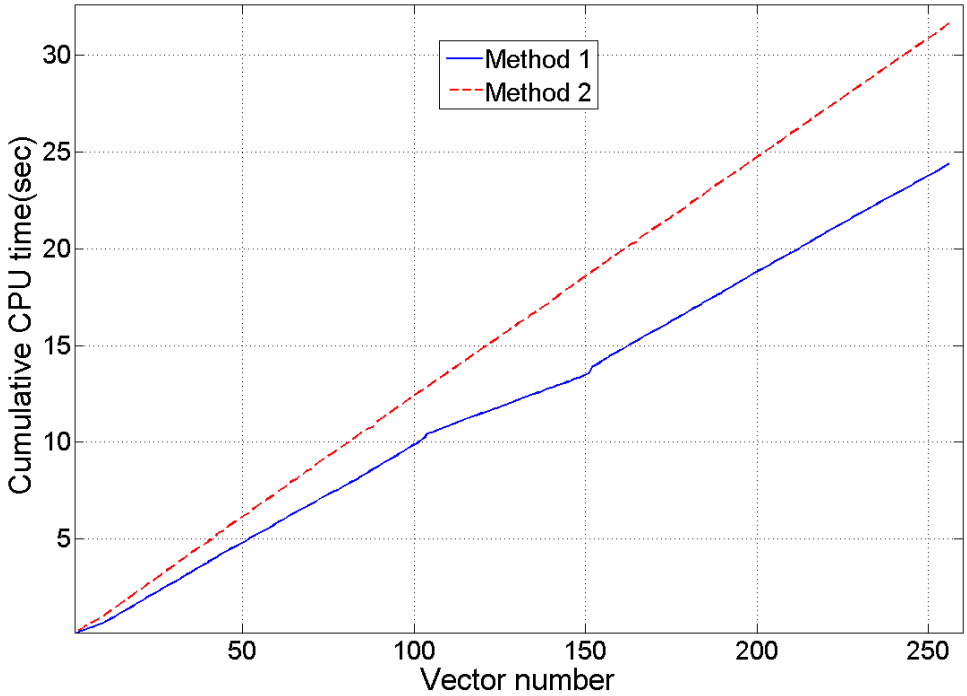
Complexity of Method 2

$$= O_{nand}\left(\frac{kn_1}{k1}\right) + O_{and}\left(\frac{kn_2}{k1}\right) + O_{nor}\left(\frac{kn_3}{k1}\right) + O_{or}\left(\frac{kn_4}{k1}\right) + O_{xor}(kn_5)$$

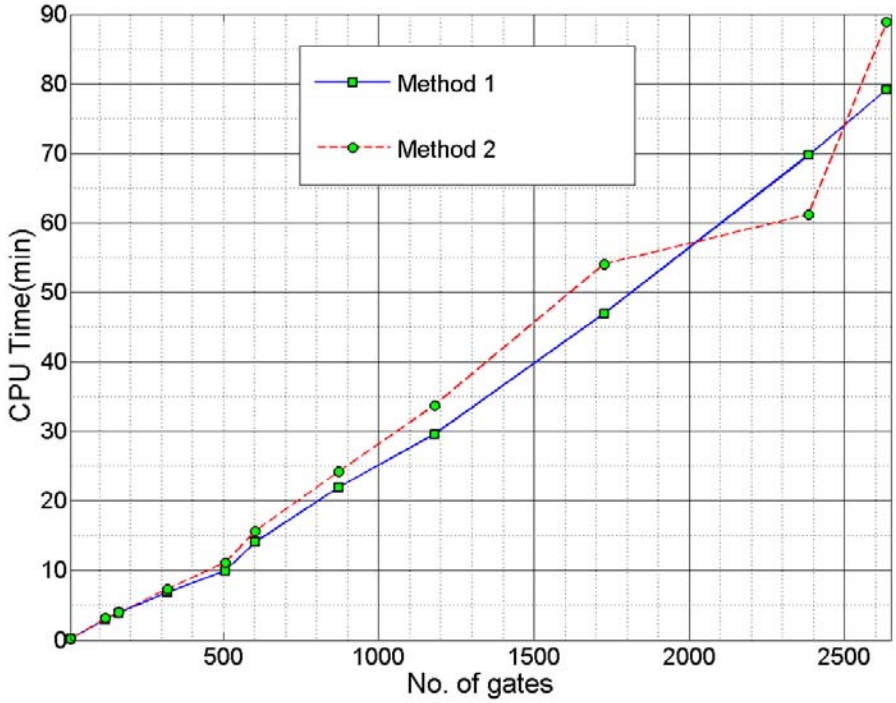
$$+ O_{xnor}(kn_6) + O_{buff}(n_7) + O_{not}(n_8)$$

$$Note: n = n_1 + n_2 + n_3 + n_4 + n_5 + n_6 + n_7 + n_8$$

MATLAB Version: 7.0.4.365 (R14) Service Pack 2
Machine: Intel ® Celeron ™ CPU 1.19 GHz with 384 MB RAM



(2) Plot CPU time vs. number of gates for ISCAS'85 circuits each simulated for 1000 random vectors



Appendix-A
4-bit ripple carry adder Truth Table

Inputs(In)
Outputs(On)

I1	I2	I3	I6	I7	I10	I11	I14	I15	O4	O8	O12	O16	O17
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	1	1	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	1	0	0	1	1	0
0	0	0	0	0	0	1	1	0	0	0	1	1	0
0	0	0	0	0	0	1	1	1	0	0	1	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	0	1	1	0
0	0	0	0	0	1	0	1	0	0	0	1	1	0
0	0	0	0	0	1	0	1	1	0	0	1	0	1
0	0	0	0	0	1	1	0	0	0	0	0	1	0
0	0	0	0	0	1	1	0	1	0	0	0	0	1
0	0	0	0	0	1	1	1	0	0	0	0	0	1
0	0	0	0	0	1	1	1	1	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	1	0	1	0	1	0
0	0	0	0	1	0	0	1	0	0	1	0	1	0
0	0	0	0	1	0	0	1	1	0	1	0	0	1
0	0	0	0	1	0	1	0	0	0	1	1	0	0
0	0	0	0	1	0	1	0	1	0	1	1	1	0
0	0	0	0	1	0	1	1	0	0	1	1	1	0
0	0	0	0	1	0	1	1	1	0	1	1	0	1
0	0	0	0	1	1	0	0	0	0	1	1	0	0
0	0	0	0	1	1	0	0	1	0	1	1	1	0
0	0	0	0	1	1	0	1	0	0	1	1	1	0
0	0	0	0	1	1	0	1	1	0	1	1	0	1
0	0	0	0	1	1	1	0	0	0	1	0	1	0
0	0	0	0	1	1	1	0	1	0	1	0	0	1
0	0	0	0	1	1	1	1	0	0	1	0	0	1
0	0	0	0	1	1	1	1	1	0	1	0	1	1
0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	1	0	1	0	1	0
0	0	0	1	0	0	0	1	0	0	1	0	1	0
0	0	0	1	0	0	0	1	1	0	1	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1	0	0
0	0	0	1	0	0	1	0	1	0	1	1	1	0
0	0	0	1	0	0	1	1	0	0	1	1	1	0
0	0	0	1	0	0	1	1	1	0	1	1	0	1

0	0	0	1	0	1	0	0	0	0	1	1	0	0
0	0	0	1	0	1	0	0	1	0	1	1	1	0
0	0	0	1	0	1	0	1	0	0	1	1	1	0
0	0	0	1	0	1	0	1	1	0	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1	0
0	0	0	1	0	1	1	0	1	0	1	0	0	1
0	0	0	1	0	1	1	1	0	0	1	0	0	1
0	0	0	1	0	1	1	1	1	0	1	0	1	1
0	0	0	1	1	0	0	0	0	0	0	1	0	0
0	0	0	1	1	0	0	0	1	0	0	1	1	0
0	0	0	1	1	0	0	1	0	0	0	1	1	0
0	0	0	1	1	0	0	1	1	0	0	1	0	1
0	0	0	1	1	0	1	0	0	0	0	0	1	0
0	0	0	1	1	0	1	0	1	0	0	0	0	1
0	0	0	1	1	0	1	1	0	0	0	0	0	1
0	0	0	1	1	0	1	1	1	0	0	0	1	1
0	0	0	1	1	1	0	0	0	0	0	0	1	0
0	0	0	1	1	1	0	0	1	0	0	0	0	1
0	0	0	1	1	1	0	1	0	0	0	0	0	1
0	0	0	1	1	1	0	1	1	0	0	0	1	1
0	0	0	1	1	1	1	0	0	0	0	1	1	0
0	0	0	1	1	1	1	0	1	0	0	1	0	1
0	0	0	1	1	1	1	1	0	0	0	1	0	1
0	0	0	1	1	1	1	1	1	0	0	1	1	1
0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0	1	1	0	0	1	0
0	1	0	0	0	0	0	1	0	1	0	0	1	0
0	1	0	0	0	0	0	1	1	1	0	0	0	1

-Project 2-

Aim

In general the problems addressed are handling files and creation of new files by mean of string replacement functions and as well as creation of "data structures" for the components in the 4-BIT Full Adder designed in project1.

Assumptions

As such any "Find and Replace" function suffers from the problem of identifying "S1" and "S11" as one and the same. These problems are to be overcome while naming the net list and also few other considerations are made so as to make the compiler fairly simple in operation.

Algorithm

(1) Program asks for user choice for "default" or "hierarchical" mode. Choice=1 default and Choice=2 hierarchical mode.
(2) In hierarchical mode, generates a simulation table by seeing the circuit as composed of hierarchies connected together. Output is written into a file called "HierSim.txt". Program for hierarchical mode is shown in Table-1 and output is shown in Appendix-A.
(3) In default mode, the program reads the "main circuit description" along with the sub-circuit netlist and generates a new flatten netlist by suitably replacing the sub-circuit with its internal description and also renumbering their internal internal nets. Outputs are written into a file called "Flat.txt" for flattened netlist and "FlatSim.txt" for simulation table generated for flattened netlist. Interestingly the program generates the flattened simulation table by reading the flattened netlist and thereby simplifies the problem. Program for default mode is shown in Table-2 and outputs are shown in Appendix-B and -C.
(4) Suitable data structures in the form of "structure" are created for each gate in the circuit with all its attributes.

Table 1

```
%%%%%%%%%%%%%%%%%%%%program for hierarchical netlist%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%simulation table generation%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all;
clc;
clear all;

fid= fopen('ra4.txt', 'rt');
fid7=fopen('HierSim.txt','wt');
fprintf(fid7,'Hierarchial Simulation Table');
fprintf(fid7,'\n');

y = 0;
i=1;j=1;k=1;
choice=input('Hierarchial/flatten mode');
if(choice==2)
   while feof(fid)==0
    tline = fgetl(fid);
            if(findstr(tline, 'INPUT'))
               v=tline(7:(findstr(tline,')'))-1);
```

```
            inputs(j)=cellstr(v);
            j=j+1;
        elseif(findstr(tline, 'OUTPUT'))
            v=tline(8:(findstr(tline,')'))-1);
            outputs(k)=cellstr(v);
            k=k+1;
        elseif(findstr(tline,'FAU'))
            gate(i).name=sprintf('FAU%d',i);
            fprintf(fid7,'\ngate(%d).name:\t%s\n',i,gate(i).name);
            gate(i).type='FA';
            fprintf(fid7,'gate(%d).type:\t%s\n',i,gate(i).type);

            fanin=tline((findstr(tline,'U'))+2:max(findstr(tline,')'))-1);
            fin=str2num(fanin);
            for t=1:length(fin)
            v=strcat('FA.','FAU',num2str(i),'.',num2str(fin(t)));
            fprintf(fid7,'Fanin(%d):\t%s\n',t,v);
            end;

            fanout= tline(2:(findstr(tline,')'))-1);
            fout=str2num(fanout);
            for t=1:length(fout)
            v=strcat('FA.','FAU',num2str(i),'.',num2str(fout(t)));
            fprintf(fid7,'Fanout(%d):\t%s\n',t,v);
            end;


            gate(i).delay=0;
            fprintf(fid7,'gate(%d).delay:\t%d\n',i,gate(i).delay);

            gate(i).fault=0;
            fprintf(fid7,'gate(%d).fault:\t%d\n',i,gate(i).fault);
            i=i+1;
            fprintf(fid7,'==================================');
        end;
  end
else
 expand1();
end;
```

Table 2

```
%%%%%%%%%%%%%%%%%Program for Flattening the netlist%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%Program for generating its simulation table%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all; clear all;clc;
fid3 = fopen('FAU.txt', 'rt');
fid=fopen('ra4.txt','rt+');

fid1=fopen('flat.txt','wt+');
fprintf(fid1,'Program for flattening by KASI ANBUMONY \n');
fprintf(fid1,'Flattened net list for ra4.txt\n');
fprintf(fid1,'created on:');
fprintf(fid1,date);
```

```
fprintf(fid1,'\n');

cnt=0;j=1;
while feof(fid) == 0
tline = fgetl(fid);

  if(findstr(tline,'INPUT'))
  fprintf(fid1,tline);
  fprintf(fid1,'\n');
  elseif(findstr(tline,'OUTPUT'))
  fprintf(fid1,tline);
  fprintf(fid1,'\n');
  v=tline(8:(findstr(tline,')'))-1);
  outputs(j)=str2num(v);
  mx_net=max(outputs);

  elseif(findstr(tline,'FAU'))
     fanin=tline((findstr(tline,'U'))+2:max(findstr(tline,')'))-1);
     fin=str2num(fanin);
     fanout= tline(2:(findstr(tline,')'))-1);
     fout=str2num(fanout);
     ftot=[fin fout];

     fid3 = fopen('FAU.txt', 'rt');
     cnt=cnt+1;
     m=1;

     fprintf(fid1,'#FAU%d',cnt);
     fprintf(fid1,'\n');

     while feof(fid3)==0
     tline3=fgetl(fid3);
              if(findstr(tline3,'INPUT'))
              v=tline3(7:(findstr(tline3,')'))-1);
              inputs(m)=cellstr(v);
              m=m+1;
              elseif(findstr(tline3,'OUTPUT'))
              v=tline3(8:(findstr(tline3,')'))-1);
              inputs(m)=cellstr(v);
              m=m+1;
              elseif(findstr(tline3,'#'))
              else
                     for k=1:(m-1)
                     tline3=regexprep(tline3,inputs(k),num2str(ftot(k)));
                     end;
                      tline3=strrep(tline3,'S4',num2str(mx_net+cnt));
                      tline3=strrep(tline3,'S6',num2str(mx_net+cnt+1));
                      tline3=strrep(tline3,'S7',num2str(mx_net+cnt+2));
                      tline3=strrep(tline3,'S8',num2str(mx_net+cnt+3));

                     fprintf(fid1,tline3);
                     fprintf(fid1,'\n');
             end;
     end;%end of minor while loop
```

```
       mx_net=mx_net+cnt+3;
    end;%end of else loop
end;%end of major while loop
fclose(fid1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all;
clc;
clear all;

fid= fopen('flat.txt', 'rt');

fid7=fopen('FlatSim.txt','wt');
fprintf(fid7,'Flatten Simulation Table');
fprintf(fid7,'\n');

y = 0;
i=1;j=1;k=1;ia=1;ir=1;ix=1;

   while feof(fid)==0
   tline = fgetl(fid);

               if(findstr(tline, 'INPUT'))
                   v=tline(7:(findstr(tline,')'))-1);
                   inputs(j)=cellstr(v);
                   j=j+1;

               elseif(findstr(tline, 'OUTPUT'))
                   v=tline(8:(findstr(tline,')'))-1);
                   outputs(j)=cellstr(v);
                   k=k+1;

               elseif(findstr(tline,'AND'))
                   gate(i).name=sprintf('AND%d',ia);
                   gate(i).type='AND';

                   fanin=tline((findstr(tline,'D'))+2:max(findstr(tline,')'))-1);
                   fin=str2num(fanin);
                   for t=1:length(fin)
                   v=strcat('AND',num2str(ia),'/',num2str(fin(t)));
                   gate(i).fanin(t).fin=sprintf('Fanin(%d):\t%s\n',t,v);
                   end;

                   fanout= tline(1:(findstr(tline,'='))-1);
                   fout=str2num(fanout);
                   v=strcat('AND',num2str(ia),'/',num2str(fout));
                   gate(i).fanout=v;

                   gate(i).delay=0;
                   gate(i).fault=0;
                   ia=ia+1;
                   i=i+1;
               %%%%%%%%%%%%%%%
```

```
            elseif(findstr(tline,'XOR') & findstr(tline,'OR'))

            gate(i).name=sprintf('XOR%d',ix);
            gate(i).type='XOR';

            fanin=tline((findstr(tline,'R'))+2:max(findstr(tline,')'))-1);
            fin=str2num(fanin);
            for t=1:length(fin)
            v=strcat('XOR',num2str(ix),'/',num2str(fin(t)));
            gate(i).fanin(t).fin=sprintf('Fanin(%d):\t%s\n',t,v);
            end;

            fanout= tline(1:(findstr(tline,'='))-1);
            fout=str2num(fanout);
            v=strcat('XOR',num2str(ix),'/',num2str(fout));
            gate(i).fanout=v;

            gate(i).delay=0;
            gate(i).fault=0;
            ix=ix+1; i=i+1;

           elseif(findstr(tline,'OR'))

            gate(i).name=sprintf('OR%d',ir);
            gate(i).type='OR';

            fanin=tline((findstr(tline,'R'))+2:max(findstr(tline,')'))-1);
            fin=str2num(fanin);
            for t=1:length(fin)
            v=strcat('OR',num2str(ir),'/',num2str(fin(t)));
            gate(i).fanin(t).fin=sprintf('Fanin(%d):\t%s\n',t,v);
            end;

            fanout= tline(1:(findstr(tline,'='))-1);
            fout=str2num(fanout);
            v=strcat('OR',num2str(ir),'/',num2str(fout));
            gate(i).fanout=v;

            gate(i).delay=0;
            gate(i).fault=0;
            ir=ir+1; i=i+1;

          end;%%%%%%%%%%%%%%%%%%%%%%%%%%

  end%%%%%%%%%%%%%end of while loop

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for t=1:(i-1)
fprintf(fid7,'\ngate name:\t%s\n',gate(t).name);
fprintf(fid7,'gate type:\t%s\n',gate(t).type);
fprintf(fid7,'Fanins:\n');
  for m=1:length(gate(t).fanin)
```

```
  fprintf(fid7,'%s',gate(t).fanin(m).fin);
  end;
fprintf(fid7,'Fanout:\t%s\n',gate(t).fanout);
fprintf(fid7,'gate delay:\t%d\n',gate(t).delay);
fprintf(fid7,'gate fault:\t%d\n',gate(t).fault);
fprintf(fid7,'===================================');
end;
fclose(fid7);
disp('Completed');

end;
```

Conclusion

Thus a compiler has been successfully designed for the given specifications using a scripting language-MATLAB.