

AUSIM: Auburn University Simulator - Version L2.2

by Dr. Charles E. Stroud, Professor
Dept. of Electrical & Computer Engineering
Auburn University
February 18, 2004

ABSTRACT

The operation of digital logic simulator called the Auburn University SIMulator (AUSIM) is described. This version of AUSIM is capable of simulating non-hierarchical circuit descriptions at the elementary logic gate level and provides area and performance audits of the circuit being simulated.

1. INTRODUCTION

Logic simulation has become essential in ensuring that a digital design is correct prior to actual implementation of the hardware. The process of ensuring the correctness of a digital logic circuit through simulation is often referred to as **design verification**. One of the inputs to any logic simulation tool is a description of the digital design in some hardware description language. The hardware description language (also referred to as a **netlist**) for Auburn University Simulator (AUSIM) is the Auburn Simulation Language (ASL) [1]. In addition to providing simulation for debugging and verifying digital designs, AUSIM provides audits which can aid in debugging a circuit or in analyzing a circuit in terms of area and performance metrics. This version of AUSIM (version L2.2) simulates digital circuits described as elementary logic gates (AND, OR, NOT, NAND, and NOR gates). However, this version of AUSIM does not support hierarchical ASL circuit descriptions. Therefore, a brief overview of the ASL format is given in Section 2 to facilitate a shorter learning curve for simulation with AUSIM. Another required input to any logic simulation is an input stimulus (also referred to as **vector**) file. A brief overview of the vector file format for AUSIM is given in Section 3. The actual operation of AUSIM version L2.2 is described in Section 4 including a description of the output files produced by AUSIM. Throughout this document, an example circuit (a 2-to-1 multiplexer in this case) will be used to illustrate all input and output files as well as the simulation process with AUSIM L2.2. Note that the input files (ASL file and vector file) can be generated by any text editor program but that these files should be saved as text files.

2. ASL CIRCUIT DESCRIPTION

To begin, consider the 2-to-1 multiplexer (sometimes referred to as a selector), illustrated in Figure 1, which selects one of two possible inputs (A or B) to pass to the output (Z) based on the logic value of the control input (S). If $S=0$, then $Z=A$, otherwise, if $S=1$, the $Z=B$. A typical logic symbol for a multiplexer is shown in Figure 1a while the actual gate level logic diagram is shown in Figure 1b. The logic circuit consists of two 2-input AND gates, a NOT gate (or inverter) to invert the control input S, and a 2-input OR gate to generate the output Z. The fact that the two AND gates are unique, in that each gate has a different set of input signals, can be easily seen visually from the logic diagram. When discussing the circuit, a designer may refer to "the top AND gate" or "the bottom AND gate" in order to distinguish between the multiple uses of the same type of gate. Similarly, the internal signal nodes (also referred to as **nets**) of the circuit are

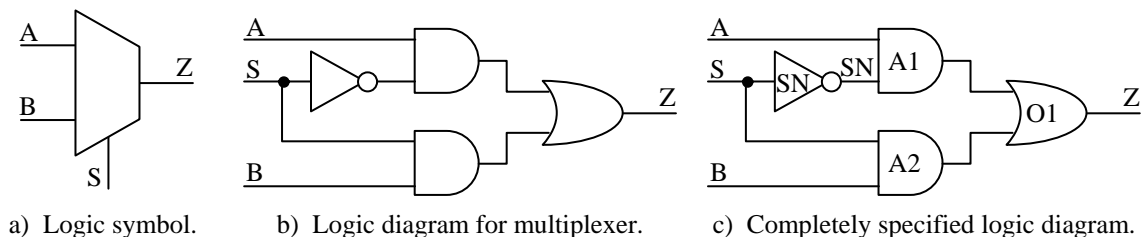


Figure 1. Example 2-to-1 multiplexer circuit.

also unique (for example, "the output of the top AND gate" or the "output of the inverter"). While the uniqueness of the various gates and nets is graphically apparent in the logic diagram of Figure 1b, the ASL description represents a textual description of the circuit and requires explicit reference to a given gate or net. Therefore, an unambiguous, completely specified logic diagram for the multiplexer is given in Figure 1c where each gate and net of the circuit is given an arbitrary but unique name. For example, the two AND gates have been named A1 and A2. In addition, the nets driven by the outputs of the two AND gates have been given the same names (A1 and A2). Naming a net the same as the name of the driving gate is not a rule (as can be seen by the OR gate, named O1, which produces the output signal Z) but rather aids in identification of the source of the signal as well as simplifies the effort of inventing names for all of the gates and nets in the circuit. From the logic diagram of Figure 1c, the ASL description for the multiplexer can be written. The ASL description consists of a single **circuit statement** and one or more **component statements**. The "circuit statement" includes the name of the complete logic circuit, its primary inputs, and its primary outputs. The circuit statement for the full adder would be written as follows:

```
CKT: MUX IN: A B C OUT: Z ;
```

The syntax of the circuit statement begins with the keyword "CKT:" to indicate that the statement is a circuit statement as opposed to a component statement. The "CKT:" keyword is followed by the name of the logic circuit (arbitrarily called FADD in this case for full adder) and separated by a delimiter. In ASL, delimiters can be any combination of SPACE, TAB, or NEW LINE characters. The circuit name is followed (after another delimiter) by the keyword "IN:" to indicate that the primary inputs to the circuit will subsequently be defined. The names of all of the primary inputs to the circuit are given immediately following the "IN:" keyword and are separated by delimiters. After all primary inputs have been listed, the keyword "OUT:" is used to indicate the beginning of the list of primary outputs. The list of primary outputs is specified in the same manner as the primary inputs with delimiters separating each entry. All ASL statements, regardless of type, are concluded with a delimiter followed by a semicolon (";").

The circuit statement indicates the beginning of the circuit description and is followed by component statements, each of which describes one component (or gate in this case) used to construct the complete circuit. The syntax of the component statement is similar to that of circuit statement with the exception of the "CKT:" keyword. In the component statement, the type of component ("NOT:", "AND:", "OR:", "NAND:", or "NOR:") is used as the keyword instead of the "CKT:" keyword. In the same manner as the circuit statement the component is given a unique name followed by the "IN:" keyword, the list of inputs to the component, the "OUT:" keyword, the output of the gate, and the semicolon indicating the end of the statement. Once the logic diagram has been completely specified with unique gate and net names, the circuit and component statements can be generated directly from the information in the logic diagram. As a result, the complete circuit description for the multiplexer of Figure 1c would be given by:

```
CKT: MUX IN: A B S OUT: Z ;
NOT: SN IN: S OUT: SN ;
AND: A1 IN: A SN OUT: A1 ;
AND: A2 IN: B S OUT: A2 ;
OR: O1 IN: A1 A2 OUT: Z ;
```

It should be noted that the use of upper case letters is not mandatory in generating the ASL such that lower case letters may be used. For example, "ckt:", "in:", "out:", "and:", "or:", "nand:", "nor:", and "not:" are also valid keywords. However, the upper and lower case characters do not alias such that "a" and "A" are not the same and if used as net names may represent two separate nets or signals. For example, if the output of the first AND gate is labeled "A1" and the input to the OR gate is labeled "a1", no connection between the gates.

Comments in circuit descriptions are useful in improving the readability of the circuit description or to include notes about the functional operation. A comment statement in ASL is indicated by a "#" at the beginning followed by a delimiter (remember that a delimiter in ASL is any combination of SPACE, TAB, or NEW LINE characters). The text of the comment follows

and can extend over any number of lines. The comment is completed in the same manner as all other statements with a delimiter followed by a semicolon. Comment statements can be included at any point in the ASL description as long as they are inserted between circuit and component statements and not within statements. For example, a valid commented version of the multiplexer circuit description could be as follows (note the use of lower case characters in this example):

```
# ASL description for a 2-to-1 multiplexer ;
ckt: mux in: a b s out: z ;
# inverter for select signal ;
not: sn in: s out: sn ;
# AND gates ;
and: a1 in: a sn out: a1 ;
and: a2 in: b s out: a2 ;
# output OR gate ;
or: o1 in: a1 a2 out: z ;
# end of ASL description for multiplexer ;
```

3. VECTOR FILE

In addition to the ASL circuit description, AUSIM requires an input stimulus file which specifies the input patterns (or vectors) to be applied to the circuit as well as the order in which they are to be applied. Comments can be included in the vector file but those comments must observe the same format and syntax as the comment statements in ASL description described above. Aside from comments statements, the input vectors are specified by writing 1s and 0s for each primary input described in the CKT: statement in the ASL description for the circuit. The 1s and 0s of a given vector MUST NOT be separated by spaces, tabs, or new lines. The order of the vectors corresponds to the order of the primary inputs in the CKT: statement in the ASL circuit description. In other words, each column of 1s and 0s will correspond to each primary input with the first column specifying the stimuli for the first primary input and the last column specifying the stimuli for the last primary input. Therefore, there should be exactly the same number of columns in the vector file as there are primary inputs in the CKT: statement. As an example, the following set of inputs vectors is given for the multiplexer with the first, second, and third columns of vectors to be applied to the A, B, and S inputs, respectively:

```
# the following input vector should cause Z=0 ;
000
001
010
101
# the following input vectors should cause Z=1 ;
011
100
110
111
# end of multiplexer vectors ;
```

4. SIMULATING WITH AUSIM

To access AUSIM L2.2, copy the AUSIM executable (ausim.exe) into the directory containing the ASL and vector files you want to simulate. From *Windows Explorer* or *My Computer*, double click the ausim.exe icon in the directory. The AUSIM window, shown in Figure 2a, should appear. The AUSIM window consists of three main areas: 1) the Input/Output File Control section at the top left hand side of the window, 2) the Circuit Statistics list at the top right hand side of the window, and 3) the Process and Simulate buttons and Status dialog area at the bottom of the window.

4.1 Entering File Names

The first step upon entering AUSIM is to specify the names of the input and output files. These can be entered individually by moving the cursor to the appropriate file name box, clicking the left mouse button, and entering the full name of the associated file name. The input files include the ASL file and input vector (or VEC) file described above. The output files include the audit (or AUD) file and simulation results (or OUT) file; these files will be discussed in Section 3.2 and 3.3, respectively. Alternatively, default file names can be specified by the user by typing the file prefix into the text box that appears in the top center of the Input/Output File Control section. As the file prefix is typed in the “prefix” text box, the names of the various input and output files appear in their respective text boxes with their default file suffix (“.asl” for the ASL file, “.vec” for the VEC file, “.aud” for the AUD file, and “.out” for the OUT file). By using the default naming convention, all files associated with a given circuit have the same prefix and can be easily found in directories containing multiple circuits or versions of a circuit. In order to use the default naming convention the ASL and VEC files must be saved and named with the correct file suffix (“.asl” and “.vec”, respectively).

4.2 Process ASL File

Once the names of the files have been specified, the ASL file must be processed by AUSIM. This is accomplished by clicking the Process button. The processing performed on an ASL file as a result of this command includes the following steps:

1. Check to see that the ASL file exists. Failure to find the file will result in a status box message “can’t find ASL file”.
2. Check syntax of the ASL file to determine if there are any obvious syntax errors in the ASL description. Syntax errors will result in a status box message “Syntax errors in ASL file - check 'ausim_errs.txt' for details”.
3. Count the number of loads on (gates driven by) signal nets the internal data structures for subsequent simulation.
4. Perform an audit of the circuit to look for connectivity problems. Problems found will result in a status box message – “Circuit errors encountered - check 'ausim_errs.txt' and AUD file. The audit of the circuit looks for problems such as:
 - a. unconnected gate inputs (nets with no driving source)
 - b. unconnected outputs (nets with no loads, other than primary outputs)
 - c. multiple gates driving the same signal net
 - d. duplicate gate names
5. Generate an audit file for the circuit and post the circuit statistics box with data for the circuit.

If no problems were found during the processing of the ASL file the screen appears as illustrated in Figure 3a with a status box message “Data structures loaded & Audit complete - circuit appears to be OK”. (Note that the message says “appears to be OK”, it does not say that the circuit works since that can only be determine by the designer via simulation.) In addition, the circuit statistics are posted including the number of primary inputs, primary outputs, number of gate, total number of gate inputs and outputs (gate I/O) for the circuit described in the ASL file, worst case number of gate delays and propagation delay through the circuit. These circuit statistics can often be useful in debugging the circuit when they do not agree with the designer’s expectations.

If problems are encountered during the processing they are reported in the status box with instructions to check the file named ‘ausim_errs.txt’. This file begins with a header comment and then lists errors and warnings found during the processing of the file. Note that warnings do

not prevent the circuit from being simulated but should be investigated at by the designer. Errors on the other hand, must be corrected prior to simulation. An example of the output in the "ausim_errs.txt" is given below. If no errors or warnings are found, the "ausim_errs.txt" will contain only the header comment.

```
# errors and warnings when processing ASL file 'mux.asl' ;
ERROR - multiple gates driving net 'a1'
```

The AUD (audit) file produced by AUSIM contains area analysis information for the circuit in terms of the number and types of gates used in the circuit along with the circuit statistics posted in the AUSIM window. The AUD file also contains connectivity information in terms of the number of loads on each signal net, the source of each signal net, and a generic propagation associated with each signal net. This AUD file information can be very useful in debugging design errors. For example, when a signal net has no driving source (a gate input is unconnected), the signal(s) can be found in the AUD file by looking at the net names in the bottom of the file to find those that do not have a driver specified – the driver column indicates "no source". In addition, gates or primary inputs with no loads are indicated by a '0' in the number of loads column. The area/performance analysis file produced for the full adder is given as follows:

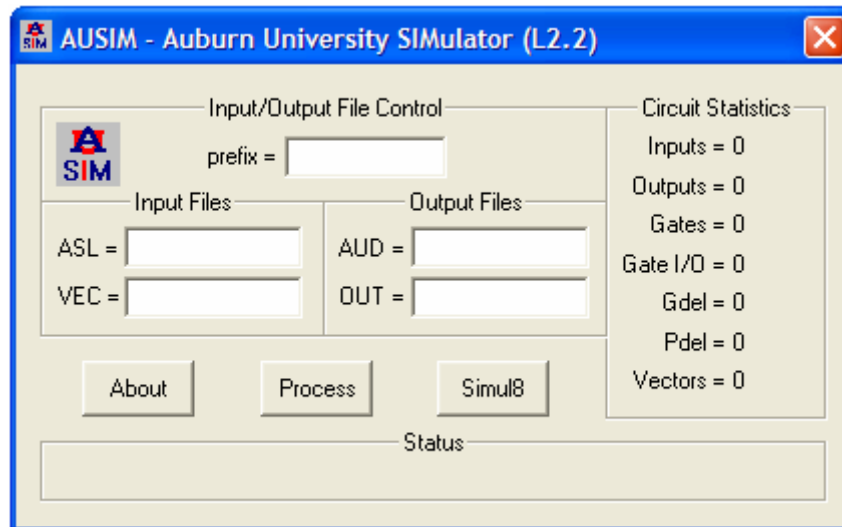
```
AUSIM (L2.2) Area/Performance Analysis Results
Circuit 'mux' from ASL file 'mux.asl'
Area Analysis:
Number of primary inputs:  Pi= 3
Number of primary outputs: Po= 1
Number of gates:           G= 4
Number of gate I/O pins:   Gio= 11
Gate type and number of uses:
AND:    2
OR:     1
NOT:    1
NAND:   0
NOR:    0
Loading and delays:
Name  Loads  Driver  Delay=intrinsic+extrinsic:
a     1     Input   1=0+1
b     1     Input   1=0+1
s     2     Input   2=0+2
z     0     OR       2=2+0      Output
sn    1     NOT     2=1+1
a1    1     AND     3=2+1
a2    1     AND     3=2+1
Worst Case Timing Path Analysis:
path= z->a1->a: Gdel=2, Pdel=5
path= z->a1->sn->s: Gdel=3, Pdel=7
path= z->a2->b: Gdel=2, Pdel=5
path= z->a2->s: Gdel=2, Pdel=5
Worst Case: Gdel=3, Pdel=7
```

The first portion of the AUD file gives the area metrics for the circuit in terms of the number of gates and in terms of the total number of inputs and outputs to the gates of the circuit. The latter metric gives a more accurate metric for comparison of circuits since it considers the size of each gate in terms of the number of inputs to the gate (which ultimately determines the size of the gate, i.e., it is proportional to the number of transistors required to construct the gate). The second portion of the area analysis gives an inventory of the numbers and types of elementary logic gates used in the design of the circuit.

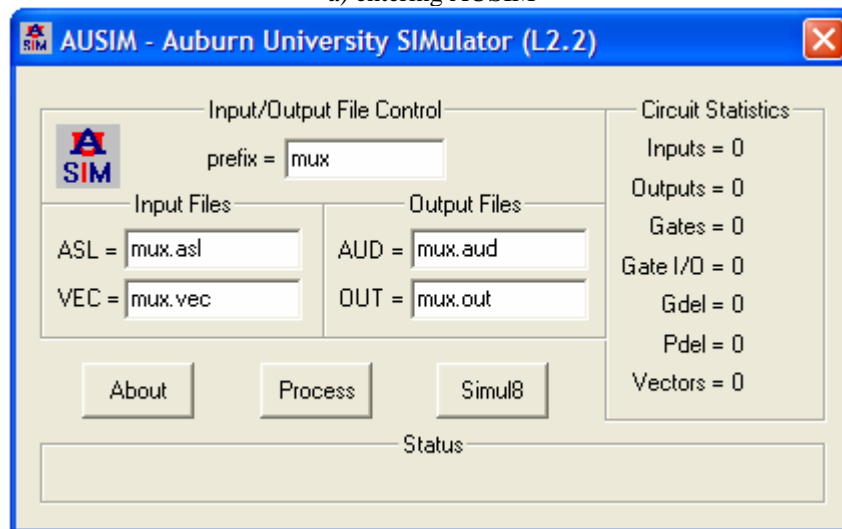
The next portion of the AUD file presents information helpful in the debugging and performance analysis of the circuit. Specifically, for each net in the circuit, the net name is given

along with the number of loads on that net (the number of gate inputs which the net drives), the type of gate or primary input that drives the net, and delay characteristics of the net. The delay characteristics are given as a summation of intrinsic and extrinsic parameters where the intrinsic parameter is given by the number of inputs to the driving gate and the extrinsic parameter is given by the number of loads on the net.

The final portion of the AUD file gives data from a worst case timing path analysis of the circuit in terms of the number of gate delays (Gdel) and propagation delay (Pdel). Note that this worst case path analysis was not performed in AUSIM version L2.1 [2]. In the timing analysis, all possible paths through the circuit (working from primary outputs back to primary inputs) are given with their associated gate delay and propagation delay. At the end of the timing analysis, the worst case gate delay and propagation delay for the entire circuit is given. Note that there may be multiple paths that give the same delay and that the worst case path for gate delay may not be the same as the worst case path for propagation delay. Also the path tracing analysis can be useful in debugging a circuit when feedback is inadvertently inserted into the combinational logic circuit, in which case an error message is given in the 'ausim_errs.txt' file.



a) entering AUSIM



b) entering default file names

Figure 2. AUSIM version L2.2

4.3 Simulate Circuit

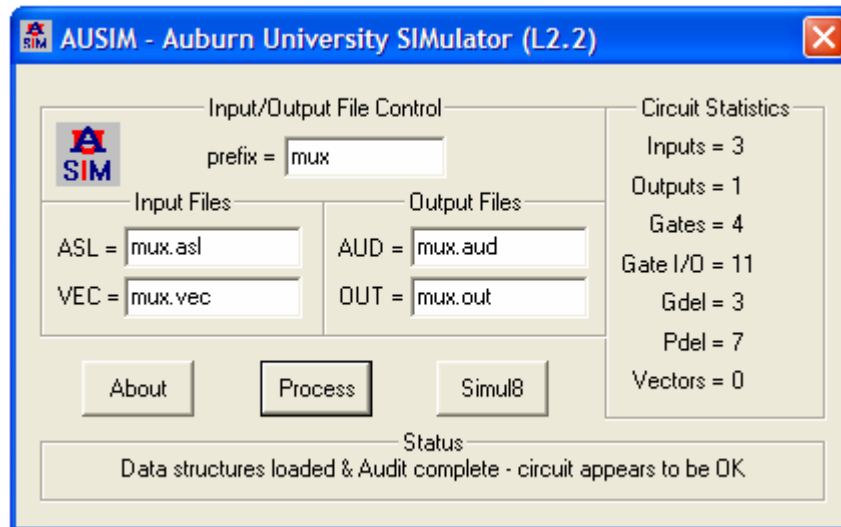
Once the data structures are loaded, the circuit can be simulated by clocking the Simul8 button. During logic simulation, the input vectors are applied in turn to the circuit while the resultant output responses are written to the simulation results (OUT) file. The number of vectors simulated is indicated in the Circuit Statistics box. Once the simulation is complete, a message appears in the Status message box as shown in Figure 3b. The simulation output file also contains any and all comments included by the user in the input stimulus file as well as additional comment lines which give the version of AUSIM being run and a vertical column listing of the primary inputs and outputs specified by the CKT: statement in the ASL description. The results of the simulation of the individual input vectors are displayed along with the corresponding output responses on a per line basis with the input stimulus and output response separated by a space. The stimulation of the full adder results in the following output file.

```
# AUSIM (L2.2) Simulation Results ;
# abs z ;
# the following input vector should cause Z=0 ;
 000 0
 001 0
 010 0
 101 0
# the following input vectors should cause Z=1 ;
 011 1
 100 1
 110 1
 111 1
# end of multiplexer vectors ;
```

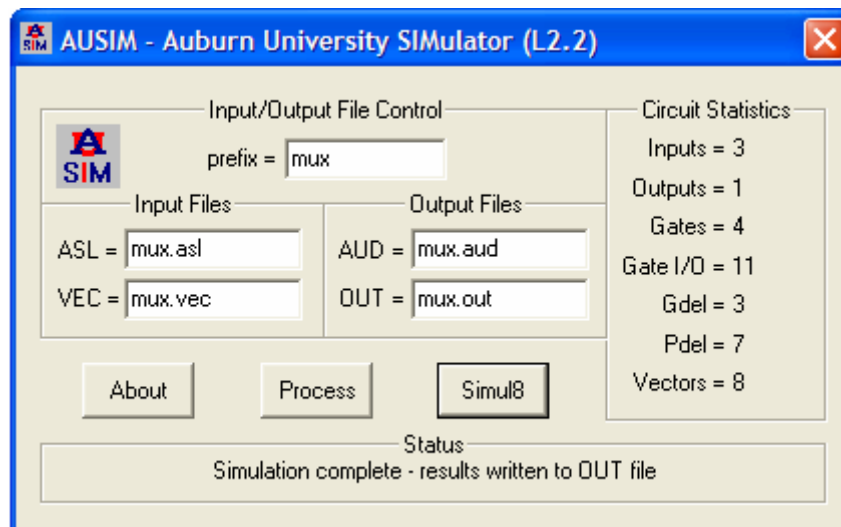
It should be noted that an output response of "2" as opposed to a "0" or a "1" may be observed. The 2 output value indicates an unknown or un-initialized logic value which can be either a 0 or a 1 in an actual hardware implementation of the circuit. An output value of 2 indicates a design error if the 2 value occurs in a combination logic. However, it should be noted that the 2 output value is a valid simulation result in the case of sequential circuits free of any design errors.

The simulation output file is the objective of any logic simulation effort and should be studied closely by the designer to ensure that the circuit is behaving as intended in all cases. Design verification is the most important part of the design process and must be pursued with dedication on the part of the designer since it is the primary point at which design is debugged. A good rule of thumb in design verification is: "If it doesn't look right, stop everything, and find out WHY!" If a bug is found, the designer should feel good that the design error has been found and fixed. On the other hand, if it is determined that the circuit is responding in the proper manner, the designer should feel more confident in the design.

Thus far, the AUSIM logic simulator has been described from a user's standpoint. From an operation standpoint, AUSIM reads in the ASL description, constructs a graph for the circuit (where a vertex represents each gate and edges represent the nets of the circuit), performs audits, and initializes all logic values in the circuit to a value of "2". AUSIM then applies the input stimulus to the circuit one vector at a time while collecting the output response. During the simulation of a given input vector, the logic values are applied to the primary inputs and the value of each node in the circuit is evaluated. If any node changes values as a result of the input stimulus, all nodes are re-evaluated to determine further propagation of logic values. Each complete evaluation of all nodes is called a time step such that a given vector may require multiple time steps of evaluation before the logic values stabilize and the final output response can be obtained. When the circuit has stabilized, (none of the nodes experience any further change in logic values) the output response is collected and the next input vector is applied.



a) processing ASL file



b) simulation

Figure 3. AUSIM version L2.2

5. SUMMARY

The AUSIM logic simulator has been described from a user's standpoint. The AUSIM simulator is menu driven and provides a number of features useful in debugging and analyzing digital logic designs. AUSIM requires that the circuit be described in ASL format. This document is intended to allow a logic designer to effectively use AUSIM but is not intended as instruction in the design process or in the art of logic design. AUSIM is representative of many of the logic and fault simulation capabilities incorporated into CAD tools used by logic and VLSI designers in industry. As a result, a good understanding and knowledge of AUSIM will provide a sound basis for the use of most other logic and fault simulation CAD tools.

REFERENCES

- [1] Charles E. Stroud, "ASL: Auburn Simulation Language", Dept. of Electrical & Computer Engineering, Auburn University, July 7, 2003.
- [2] Charles E. Stroud, "AUSIM: Auburn University Simulator - Version L2.2", Dept. of Electrical & Computer Engineering, Auburn University, January 22, 2004.