How to Use Tabular I/O with ValidSIM

Steve Munich

Valid Logic Systems
Applications Engineering

OVERVIEW

Tabular Input/Output is a very useful tool for use in both interactive and batch mode simulation.

What is a tabular output? It is merely a tabled format of simulation results. By use of a script or by interactive inputs, the SCALDsystem user can specify which signals in the design are to be traced, much like the specification of signals to be opened in waveform mode. During simulation, the tabular output file is built. The format of the file is shown by this example:

```
FILE_TYPE = TABULAR_TRACE;
Q<3..0> ,2
ENABLE ,2
CLEAR*,2
CLK !C 0-4*,2
PRESET*,2
START_TAB_TRACE;
      0 / 0000,1,1,0,1;
     30 / 0000,1,1,0,1;
     60 / 0000,1,1,1,1;
     90 / 0000,1,1,1,1;
    120 / 0001,1,1,0,1;
    150 / 0001,1,1,1,1;
    180 / 0001,1,1,1,1;
    210 / 0001,1,1,0,1;
    240 / 0010,1,1,1,1;
    270 / 0010,1,1,1,1;
    300 / 0010,1,1,0,1;
END_TAB_TRACE;
END.
```

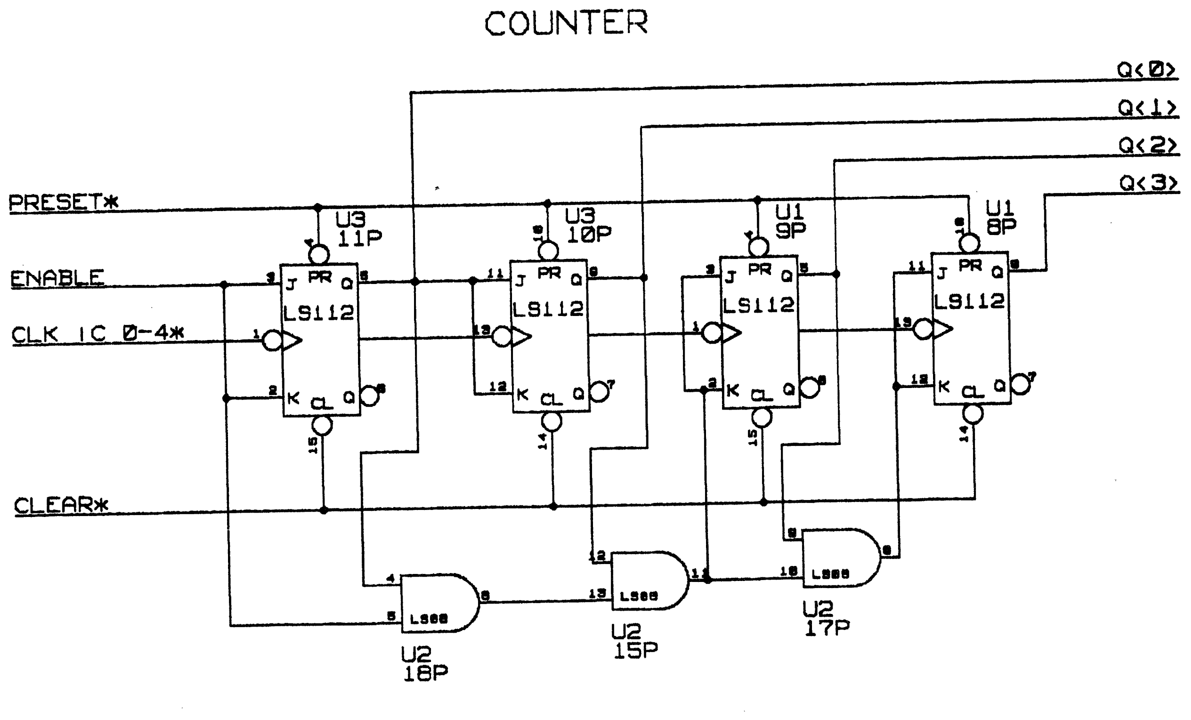
The first line of the file identifies the SCALD file type. Following this is the list of all the signals that were traced. Each signal name is followed by a comma, followed by the radix used (binary, octal, decimal or hexadecimal). The traced data is enclosed between statements demarcating the start and end of the trace block. At the beginning of each line of data is the relative time in nanoseconds, followed by "/" , then the data. Signals represented by single data bits are displayed with each bit separated by commas from other data. Data for vectored signals such as Q<3..0> have no spaces between each bit. A semicolon at the end of each line marks the end of that time sample. Notice that the left-to-right order of the data in each line follows the top-to-bottom order of the signal list. An "END." statement closes out the file. If the data in each sample were to exceed 80 characters, a tilda (~) would be used as the last character of the sample to be continued on the next line.

The format for a tabular input file is exactly the same as the output file shown above. The input file is used to stimulate a circuit in a manner analogous to interactively depositing onto signals using the "deposit" command.

It is important that the simulate.cmd file contains the directive "**tabular_trace on;**". This allows a tabular output file to be created. Below is the directives file (called simulate.cmd) used by ValidSIM for all of the tabular I/O examples in this paper. Note that the only directive of any consequence is " tabular_trace on;".

```
root_drawing 'counter';
clock_period 100;
clock_intervals 8;
terminal gcluster;
output list, command_log;
session_log on;
tabular_trace on;
end.
```

A simple 4-bit counter, shown below, was used to generate all of the example tabular output files in this application note. It may be a useful exercise to generate these results yourself using ValidSIM.



TABULAR OUTPUT

The command set for using tabular I/O is a subset of the simulator commands. All the tabular commands begin with the word "trace". For example, to produce the tabular output file on page 1 using the counter circuit, a simple set of commands was used:

```
trace_open
logic_init -*
dep ENABLE, 1
trace Q
trace ENABLE
trace CLEAR*
trace CLK !C 0-4*
trace PRESET*
trace_interval 30
trace_start
sim 300
trace_stop
trace_close
```

The first command, "trace_open", opens the tabular output file called tabfile.dat. The command "logic_init -*" initializes every node in the circuit to its non-asserted state. The next deposit command puts the circuit in the proper state to begin counting. In this example, the signals to be traced are specified directly, using the command "trace <signalname>". Notice that "trace Q" defaults to tracing Q<3..0>, or all the bits of the bussed signal.

The next command, "trace_interval 30", tells ValidSIM to trace the signals every 30 ns. If this command's argument was 0, or if the statement was left out (making the argument 0 by default), the signals will be traced whenever any of the signals change. Again, this mode will cause a huge tabfile.dat if simulating for a long period of time and if all signals are being traced in a large design. "Trace_start" tells the simulator to begin creating "vectors" in tabular mode. "Trace_close" closes tabfile.dat. Also, the signals are traced in the order that the "trace <signalname>" commands are entered.

It is a common misconception that a signal has to be "opened" in the simulation before it can be traced. The waveform display of signals and the tracing of signals are two separate processes.

Instead, the command file may look like this:

```
trace_open
trace_all
logic_init -*
dep ENABLE, 1
trace_start
sim 200
trace_stop
sim 1000
trace_start
sim 300
trace_stop
trace_close
```

Here,"trace_all" tells ValidSIM to trace all named and unnamed signals in the drawing. This is useful only when the drawing being simulated is relatively small. Note that the signal list and data from the resultant tabular output file shown on page 5 is much larger than what one might expect from such a small circuit. The size is due to the extra "unnamed signals" that have been recorded. In most cases it makes sense not to trace unnamed signals, since usually they are unnamed because their analysis is not necessary. The order of the signal listing in tabfile.dat is chosen by ValidSIM when "trace_all" is used.

Sometimes it is only necessary to view the simulation results at intermittent times, rather than continuously tracing the signals for the entire simulation. The command "trace_stop" lets the simulation continue without adding to the tabfile.dat, saving cpu time because these unneeded vectors are not created. The trace_start command can be used to continue the tracing. This start/stop sequence can be repeated indefinitely. All the tracing is still in the same single file, tabfile.dat. For example, the 1000 ns simulation period is not included in tabfile.dat, shown below, but the "sim 200" and "sim 300" intervals are included. This is evidenced by the fact that there is a large gap between the last recorded vector at time 180 and the next vector that starts at 1200 ns.

Note that in this example, the trace_interval command has been left out. This causes ValidSIM to write a vector to the output file only when one of the signals being traced makes a transition. This is the most efficient way to record the tabular output, because again, no cpu time is wasted by creating unused vectors.

```
FILE_TYPE = TABULAR_TRACE;
(COUNT LS112.8P)UN$1$4 MUX$1P$I1$2 ,2
(COUNT LS112.9P)UN$1$4 MUX$1P$I1$2 ,2
(COUNT LS112.10P)UN$1$3 AND$15P$T$4 ,2
(COUNT LS112.10P)UN$1$3 AND$16P$T$4 ,2
(COUNT LS112.11P)UN$1$3 AND$15P$T$4 ,2
(COUNT LS112.10P)UN$1$3 AND$17P$T$4 ,2
(COUNT LS112.11P)UN$1$3 AND$16P$T$4 ,2
(COUNT LS112.11P)UN$1$3 AND$17P$T$4 ,2
(COUNT LS112.10P)UN$1$4 MUX$1P$I1$2 ,2
(COUNT LS112.11P)UN$1$4 MUX$1P$I1$2 ,2
Q<3..0> ,2
CLEAR*,2
ENABLE ,2
PRESET*,2
CLK !C 0-4*,2
UN$1$LS08$15P$B ,2
UN$1$LS08$15P$Y ,2
UN$1$LS08$17P$Y ,2
(COUNT LS112.8P)UN$1$3 OR$18P$T$4 ,2
(COUNT LS112.9P)UN$1$3 OR$18P$T$4 ,2
(COUNT LS112.10P)UN$1$3 OR$18P$T$4 ,2
(COUNT LS112.11P)UN$1$3 OR$18P$T$4 ,2
(COUNT LS112.8P)UN$1$3 AND$15P$T$4 ,2
(COUNT LS112.8P)UN$1$3 AND$16P$T$4 ,2
(COUNT LS112.9P)UN$1$3 AND$15P$T$4 ,2
(COUNT LS112.8P)UN$1$3 AND$17P$T$4 ,2
(COUNT LS112.9P)UN$1$3 AND$16P$T$4 ,2
(COUNT LS112.9P)UN$1$3 AND$17P$T$4 ,2
START_TAB_TRACE;
      0 / 0,0,0,0,0,0,1,0,0,0000,1,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0;
     40 / 0,0,0,0,0,0,1,0,0,0000,1,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0;
    100 / 0,0,0,0,0,0,1,0,1,0000,1,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0;
    120 / 0,0,0,0,1,0,0,0,1,0001,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,0;
    135 / 0,0,0,0,1,0,0,0,1,0001,1,1,1,0,1,0,0,0,0,1,0,0,0,0,0,0,0;
    140 / 0,0,0,0,1,0,0,0,1,0001,1,1,1,1,1,0,0,0,0,1,0,0,0,0,0,0,0;
    200 / 0,0,0,0,1,0,0,1,0,0001,1,1,1,0,1,0,0,0,0,1,0,0,0,0,0,0,0;
   1220 / 1,1,0,0,0,0,0,1,0,0,1100,1,1,1,0,1,1,0,1,0,0,1,1,0,0,0,0,0;
   1235 / 1,1,0,0,0,0,0,1,0,0,1100,1,1,1,0,0,0,1,0,1,0,1,0,0,1,0,0,0;
   1240 / 1,1,0,0,0,0,0,1,0,0,1100,1,1,1,1,0,0,1,0,1,0,1,0,0,1,0,0,0;
   1250 / 1,1,0,0,0,0,0,1,0,0,1100,1,1,1,1,0,0,0,1,1,0,1,1,0,1,0,0,0;
   1300 / 1,1,0,0,0,0,0,1,0,1,1100,1,1,1,0,0,0,0,1,1,0,1,1,0,1,0,0,0;
   1320 / 1,1,0,0,0,1,0,0,0,1,1101,1,1,1,0,0,0,0,1,1,1,0,1,0,1,0,0,0;
   1335 / 1,1,0,0,0,1,0,0,0,1,1101,1,1,1,0,1,0,0,1,1,1,0,1,0,1,0,0,0;
   1340 / 1,1,0,0,0,1,0,0,0,1,1101,1,1,1,1,1,0,0,1,1,1,0,1,0,1,0,0,0;
   1400 / 1,1,0,0,0,1,0,0,1,0,1101,1,1,1,0,1,0,0,1,1,1,0,1,0,1,0,0,0;
   1420 / 1,1,1,0,0,0,0,1,1,0,1110,1,1,1,0,1,0,0,1,1,1,1,1,0,1,0,0,0;
   1435 / 1,1,1,0,0,0,0,1,1,0,1110,1,1,1,0,0,1,0,1,0,1,1,1,0,0,0,0,0;
   1440 / 1,1,1,0,0,0,0,1,1,0,1110,1,1,1,1,0,1,0,1,0,1,1,1,0,0,0,0,0;
   1450 / 1,1,1,0,0,0,0,1,1,0,1110,1,1,1,1,0,0,1,0,1,1,1,0,0,1,0,0,0;
   1465 / 1,1,1,0,0,0,0,1,1,0,1110,1,1,1,1,0,0,0,1,1,1,1,1,0,1,0,0,0;
   1500 / 1,1,1,0,0,0,0,1,1,1,1110,1,1,1,0,0,0,0,1,1,1,1,1,0,1,0,0,0;
END_TAB_TRACE;
END.
```

In each example above, since no tracing radix was specified, the binary trace radix is used by default. It is very useful to be able to view the simulation results with other radices, particularly hexadecimal. In the example tabular output file below, the same set of commands was used as in the last example, except that the command "trace_radix 16" was used in place of "trace_all", and that only the named signals were traced. The "trace_radix" command allows for the specification of octal, hex or binary tracing. The command's argument can be represented by "h" or "16" for hex, "b" or "2" for binary, and "o" or "8" for octal. The results of hex tracing on the above example are shown below. This will create a more compact tabular output file and increased simulation speed. This speed increase is especially noticeable when a large number of vectored signals are being traced.

```
FILE_TYPE = TABULAR_TRACE;
Q<3..0> ,16
CLEAR*,16
ENABLE ,16
PRESET*,16
CLK !C 0-4*,16
START_TAB_TRACE;
       0 / F,1,1,1,0;
      30 / 0,1,1,1,0;
      60 / 0,1,1,1,1;
      90 / 0,1,1,1,1;
     120 / 1,1,1,1,0;
     150 / 1,1,1,1,1;
     180 / 1,1,1,1,1;
    1200 / B,1,1,1,0;
    1230 / C,1,1,1,0;
    1260 / C,1,1,1,1;
    1290 / C,1,1,1,1;
    1320 / D,1,1,1,0;
    1350 / D,1,1,1,1;
    1380 / D,1,1,1,1;
    1410 / D,1,1,1,0;
    1440 / E,1,1,1,1;
    1470 / E,1,1,1,1;
    1500 / E,1,1,1,0;
END_TAB_TRACE;
END.
```

An equivalent method of chosing a different radix is to specify the signal to be traced with the notation "trace <signalname>, [radix]", where the radix can be specified in terms of octal, hex or binary using the same argument terminology as the "trace_radix" command. Using the the "trace <signalname>" command in this way also allows for the mixing of radixes in a single tabular output file. Note that the "trace_radix" command's argument will override any radix specification if "trace <signalname>, [radix]" if both methods are used together.

The initial production release of ValidSIM will only record tabular output in binary format when the command "trace_all" is used. The radix of output tracing has no regard for the radix of the waveform display. The "radix" command in the simulator only controls the waveform radix display.

TABULAR INPUT

Tabular input files can be a very effective way to stimulate a circuit. The format of the input file is exactly the same as that for the output file.

One way to create the input file is to set up a trace of the input signals to be stimulated. The initial simulation would done by entering simulator commands through a script or interactively in the simulator, including opening and depositing into signals. The tabular output file essentially keeps a record of the input vectors. The input script can now be greatly simplified with the use of the new stimulus file and up to two more commands. To read in a stimulus file, use the command "trace_read <filename>". If the stimulus file was created from a tabular output, it is advisable to rename tabfile.dat. This is absolutely necessary when a stimulus file is being used and another output file is being created in the same simulation session.

It is possible to use any radix as an input stimulus file. Make sure that the signal list at the beginning of the tabular input file contains the proper radix information , i.e. **trace <signalname>,16** for hex stimulus files.

The tabular input file can also be created by hand, if desired. For more information on this, see section 7 of the SCALDsystem Reference Manual.

It is possible that one may want to use several different stimulus files in one simulation session, or read in the same stimulus file several times during the same simulation session. In this case, the "**trace_reset**" command must be used before each different tabular input file is read into the simulation. Note that if two or more stimulus files are used in a single ValidSIM session, none of the input files can have overlapping time stamps. The only exception is when a "logic init" command is executed with each "trace_reset" command, reseting simulation time to zero. Be aware that if an input stimulus file is being used while at the same time a tabular output file is being generated, be sure to name the input file something other than "tabfile.dat".

APPLICATIONS OF TABULAR I/O

HIERARCHICAL DESIGN ENVIRONMENT

Tabular I/O is found to be very useful in the hierarchical design environment. Each designer, working on his own separate module, can take the tabular output from the connecting input module for input stimulus. This can be used to simulate each module's effect on another before concatenating the modules into one large design and invoking ValidSIM. The input stimulus file can also be useful when there is more than one module that could be tested with the same stimulus. For example, an IC designer may design functionally equivalent cells in different semiconductor technologies or with different design rules, and compare the response of each cell to another by giving each the same stimulus. The same is true for gate array and standard cell designs, whereby the stimulus file allows for a fair comparison between vendors. The only requirements are that each circuit is functionally equivalent, and that the input signals to be stimulated are given the same signal names. Using the unix command "diff" is an easy way to compare any two files. This command will display the differences between any two ascii files, and will report the differences between binary files. The syntax is "diff file1 file2".

TEST VECTOR GENERATION

Another application of tabular output files is to interface to automatic test equipment. Here, the ValidSIM tabular output must first be formatted for each different tester using an interface program, but the basic idea is still the same. The formatted test vectors are then used to compare the simulation outputs to that of the real circuit being tested. Fault simulation can be used to determine if the fault coverage is adequate and if test time is minimized. Each fault simulator uses its own format, so a different interface program must be used to convert the SCALDsystem's tabular output vector format to that of the fault simulator.

An automatic tester gives the most reliable results if the circuit is sampled at its most stable state, being the instant of time right before its outputs are going to change. If the circuit is clocked every 100 ns, causing the outputs to change with this same period, it would be desirable to take our first sample at around 90 ns, and then every 100 ns thereafter. This can be done by creating an effective offset for the first simulation by initially simulating for the offset period before opening the tabular trace output file, as shown in this final example:

```
logic_init -*
dep CLEAR*,0
sim 90
trace_open
trace_interval 100
trace Q
trace CLEAR*
trace ENABLE
trace PRESET*
trace CLK !C 0-4*
dep CLEAR*,1
dep ENABLE,1
trace_start
sim 1000
trace_stop
trace_close
```

```
FILE_TYPE = TABULAR_TRACE;
Q<3..0> ,2
CLEAR*,2
ENABLE ,2
PRESET*,2
CLK !C 0-4*,2
START_TAB_TRACE;
      90 / 0000,1,1,1,1;
     190 / 0001,1,1,1,1;
     290 / 0010,1,1,1,1;
     390 / 0011,1,1,1,1;
     490 / 0100,1,1,1,1;
     590 / 0101,1,1,1,1;
     690 / 0110,1,1,1,1;
     790 / 0111,1,1,1,1;
     890 / 1000,1,1,1,1;
     990 / 1001,1,1,1,1;
    1090 / 1010,1,1,1,1;
END_TAB_TRACE;
END.
```