ValidCOMPILER Usage and Anomalies

(Program Release ValidCOMPILER 1.0.3)

09 Dec 85 Bill Hunsicker

1.0 ValidCOMPILER

The ValidCOMPILER package greatly improves the speed of compilation and analysis tool loading over the SCALD compiler. Analysis tools using this mechanism can be run without previously having to compile anything. The tool will then run a page-at-a-time compiler (ValidPAGECOMP) that will compile only those pages that have been updated (or affected by a changed text macro definition) and will link them together (with ValidLINKER). If all is ok (no compiler or linker ERROR messages), the tool will then run; otherwise, the error messages will be gathered together in the compiler list file (cmplst.dat) for inspection. When the errors have been fixed, the tool may be run again, and the fixed pages will automatically be recompiled. The compilations are performed according to the contents of compiler.cmd in the current directory (as before).

ValidSIM is the first tool released with this capability. It can be run with an expansion file (the old way) or with the ValidCOMPILER mechanism. Specifying a ROOT_DRAWING directive in the simulate.cmd file or specifying a root drawing in the command line:

```
simulate "my drawing"
```

causes the ValidCOMPILER mechanism to be used (no previous compilation is required).

Other tools (such as the packager) still require expansion files (produced by running the compiler before running the tool).

2.0 Error Messages

The ValidCOMPILER mechanism compiles drawings page-by-page. The results for each page are stored until a change is detected that requires the page to be recompiled. These pages are then linked together to form the entire design. Each page can contain ERROR, OVERSIGHT or WARNING messages; these can also be generated when linking the pages together. As the error messages are stored with the output, a mechanism is necessary to gather them together for viewing so that they can be fixed.

When ERRORs occur in the language processing (using ValidCOMPILER or ValidSIM), then a program called COMPERR is run automatically to produce a cmplst.dat file containing all ERROR, OVERSIGHT, and WARNING messages. This program can also be run directly by a user at any time. It is not automatically run if only WARNINGs and/or OVERSIGHTs are present (as these usually do not prevent continuing), so it is recommended that COMPERR be run occasionally to allow these to be cleaned up. COMPERR is described completely in its own document, but in brief it takes root drawing and compile type arguments (like COMPILE) and these take default values in the same way that they do for the COMPILE command.

The error listing produced (in cmplst.dat) lists "compiler" messages (organized page by page) and "linker" messages (which are generated when gathering the pages together to form the design). The page compiler catches as many errors as it can without looking at other pages. The rest are caught during the linking of the pages.

3.0 Access Control

When running a ValidCOMPILER analysis tool (such as ValidSIM) on a drawing, that drawing and all of its sub-drawings become candidates for compilation during that run. The result of each (page) compilation is stored with the page that was compiled (meaning within the drawing directory where the page's connectivity and other GED files are stored). This allows one compiled result to be shared between users and designs (saving a considerable amount of disk space). This means that if Joe's design uses Bill's drawing, then Bill's drawing may be compiled (with results stored in its directory under Bill's account) when Joe simulates his design. In order to do this, Joe must have read, write, and execute access to the directory containing any of Bill's drawings that he uses and read/write access to any of the compiler files therein.

The operating system on the S32 provides access control for the user, group, and world. If Joe and Bill are in the same group, then allowing full access to the group will allow them to use each others drawings without running into access problems during the page compilations. The compiler automatically generates files with read/write access for the group. (It does this by executing "umask 2" in the file /u0/scald/linker/lnkassign.sh.) GED (because it also executes /u0/scald/linker/lnkassign.sh) will now generate drawing directories with read/write/execute access for the group. Old GED drawings probably will NOT have write permission for the group -- this can be changed by executing "chmod +w *" in the directory that contains all of the drawings. (If this feature is NOT desired, have the system manager delete the "umask 2" line from /u0/scald/linker/lnkassign.sh. The compiler and GED will then run with umask set as specified by each user's login command file.)

When accessing library drawings (where the drawing directory is owned by the "lib" account) the compiler will run as if "lib" was the user; otherwise it runs as the real user. (For those familiar with the operating system, "real user" here means exactly that and not "effective user".) This makes it possible for library drawings to be automatically compiled for any user though only "lib" has write access to them.

If a drawing is used that exists on another node of the network, the user must have an account on both nodes (unless only drawings owned by "lib" will be accessed remotely) and the local node must be listed in that node's /hosts.equiv file. (These are the conditions that allow rsh to work.) Additionally, ValidCOMPILER must be installed on both nodes.

4.0 Anomalies

The following anomalies exist with the new mechanism (ValidCOMPILER or ValidSIM using the ROOT_DRAWING directive). They are intended to be fixed in some upcoming release.

1. The MASTER_LIBRARY compiler directive does not work.

2. The PRIMITIVE compiler directive is not implemented.

3. The SUPPRESS compiler directive works only on compiler messages. Linker oversights and warnings cannot be suppressed. This applies to the WARNINGS OFF and OVERSIGHTS OFF directives as well.

4. The REPORT compiler directive is not implemented.

5. The OUTPUT CHIPS compiler directive is not implemented.

6. The ERROR_HELP ON compiler directive is not implemented.

7. The linker will not report signals synonymed to their own complement.

8. Error messages not directly associated with a page (such as "Drawing not found") are reported only to the monitor (and cmplog.dat), so are not included in the list file (cmplst.dat) produced by COMPERR.

9. The compiler does not detect different property attributes (such as PERMIT(PIN) INHERIT(PIN)) when deciding whether to recompile a page. (Property attributes describe how a property can be used, not what its value is -- see the SCALD manual for a full description.) This kind of change is usually made very infrequently. If changes of this kind are made, then all drawings can be forcibly marked "dirty" by going to all working directories (and library directories) and executing the command rm /schema in each. The absence of this file from a drawing directory causes all pages of the drawing to be recompiled. All users sharing libraries or drawings must specify the same property attributes.

10. The compiler does not detect different values for the BUBBLE_CHECK, SUPPRESS, WARNINGS or OVERSIGHTS directives when deciding whether to recompile a page. Unlike the above, this is by design and there are no plans to change it.

11. The mechanism used to detect changes to plumbing drawings is not entirely foolproof (this may or may not be changed). In particular, if a different "standard" library is specified than was used the last time the page was compiled while leaving the old library files in place this change will not be detected.

5.0 Language Changes

The following are language changes adopted with the ValidCOMPILER. They will not be changed. The old compiler still does things the old way. The first concerns a change that must be made to all FLAG body drawings to make them compatible with the new system. The next two concern errors now reported that were not previously reported. They will be of interest if they are discovered in a design that previously compiled without error. The last two changes concern rather advanced topics and are subtle, so readers not well familiar with the SCALD III language should skip them.

1. FLAG bodies must have the BODY_TYPE="FLAG_BODY" property on the drawing body in both the LOGIC drawing and the PART drawing. (Previously this was needed only in the PART drawing.) It is NOT necessary to update any drawings using the flag bodies after making this change (as none of the FLAG.BODY drawings have been changed).

2. Signals are checked against their complements for consistent scope and kind (vector or scalar). In other words, it is an error to have (assuming format 1) a signal A along with a signal A<2..0>* just as it would be an error to have both A and A<2..0>. Similarly, one cannot have both A\I and A*\L just as one cannot have both A\I and A\L in the same drawing.

3. Drawings having more than one (different) abbreviation are reported as oversights.

4. Text macros specified on DEFINE bodies are defined only within all pages of that drawing, version, and extension. (Such as MYDRAWING.LOGIC.1.). Previously, the definition was inherited downward through the hierarchy. Note: where this inheritance was used, it was usually used to define some global text macros in a define body of the root drawing, which then allowed them to be used throughout the design. This same result can be obtained by defining unreserved global text macros in a text macro file. The SCALD manual describes how to define reserved global text macros this way, and unreserved macros are defined in the same way, but with the word UNRESERVED appearing after the definition and before the semicolon that terminates it.
   Ex:
   ```
   my_city "San Jose" unreserved;
   ```
   in a text macro file defines the text macro MY_CITY which expands to "San Jose" and is unreserved (can be locally overriden by a parameter or DEFINE body specification). NOTE: If your drawing counted on the downward inheritance of text macros defined by DEFINE bodies, then it will not work with ValidCOMPILER until the macros are defined in a text macro file as described above. The SCALD compiler will successfully compile the drawing as it did in the past.

5. PRIM and PART drawings for drawings with one version must not depend on their parameterization. In other words, any drawing body properties in the PRIM or PART drawings should not have values that use the % text macro expansion mechanism where the macro to be expanded is to be passed in as a parameter.

Additionally, there have been bug fixes that effect compilation results. These bugs have been fixed for both the SCALD compiler and the ValidCOMPILER. Directives have been added to turn the bug fixes on or off for backward compatibility. The directives default to the value that fixes the bug. Use the SCALD compiler with the directive set as specified (below) to get backward compatibility -- BUT ONLY WHEN ABSOLUTELY NECESSARY.

1. BUG -- Upon encountering an interface signal for a pin FOO which is not connected, the compiler creates a local signal PINNAME$FOO to replace the interface signal. The problem is that PINNAME$FOO is treated strictly as a local signal when in fact it should continue to be treated as an "almost" interface signal. The problem is particularly acute when using X (XSTEP < SIZE) replication in the compiler. When this is done, local signals are local to each replication of the drawing, while interface signals are shared between all replications. Treating the PINNAME$FOO signal (which was originally FOO\I) as a local signal causes separate local signals to be generated for each replication, when it was intended to be shared by all replications. It therefore is a signal with more global scope than any local signal and should always be chosen as the base signal over a local signal (unless the local signal has "name" properties).

   FIX -- The signals generated to represent interface signals for unconnected pins are considered to have greater scope than local signals and are shared by all replications of a drawing when performing X-replication. Because they have greater scope, they are chosen as base over local signals (unless preempted by name properties in the local signal). This has been fixed in the ValidCOMPILER and the SCALD compiler (version 7.27a:27Nov85 or later). The SCALD compiler supports a directive that can restore the original anomalous behavior.

   This bug fix results in a PINNAME$FOO signal being picked as the base signal where previously a local signal synonymed to it (such as MYSIG) may have been picked. This results in an equivalent circuit, but with PINNAME$FOO substituted for MYSIG. To the packager, this is a changed circuit. If compilation the old way (with the bug, so that the packager sees the same base signal names) is desired, this can be done by using the SCALD compiler with the directive LOCALLY_GLOBAL OFF. This directive is not supported by the ValidCOMPILER. (The name of the directive derives from the fact that, when ON, the compiler generated PINNAME$ signals are to be treated as "locally global" instead of "local".) Note that this directive exists only to allow a previous bug to be recreated for old unchanged designs where this degree of backward compatibility is essential. It should not be used for other cases.

2. BUG -- The parameter value evaluation mechanism for string-valued parameters did not work as documented (and intended). A (non-integer) parameter such as MAX_DELAY="20NS\PARAMETER" would be transformed into MAX_DELAY="20". Similarly, a parameter such as CAP="3.0E-12" (meant to be passed as a string to the analysis tool which then treats it as a real number) would be passed as CAP="3.0-12". In short, trailing characters would be deleted from what looked like an "integer". This came about because the compiler parsed the value with the same parser

that it uses to parse integer parameter values (such as SIZE="1B") and signal names, so it treated the parameter value as a string of "tokens" (words and punctuation marks) with strings like 20NS or 0E being treated as a single token that was equivalent to the integer without the trailing characters. (This mechanism exists to allow documentation to be added to the integers used to define values for integer-valued parameters. It was not intended to affect the way that string-valued parameters are handled.)

FIX -- The SCALD compiler and ValidCOMPILER now parse string parameter values as stated in the documentation. Text macro expansion is done on identifiers, but the value is otherwise unmodified. Identifiers must be delimited to become candidates for text macro expansion (meaning they must be preceded and followed by a character not allowed in identifiers or by the start or end of the parameter value).

If backward compatibility in this is absolutely necessary, the compiler directive TOKENIZE_PARAMS ON can be used. It is highly recommended that this be used only with the SCALD compiler and not with the ValidCOMPILER (though it works for both) so that any new designs (or old designs being changed anyway) which use parameters erroneously will be fixed.