VELOCITY CAE™

TESTING AT THE SPEED OF LIGHT

Velocity CAE Program Generator

For Simulation to ATE and ATE to ATE Conversion

Release 8.1.0.0

User's Guide

Velocity CAE Program Generator User's Guide

COPYRIGHT NOTICE

Documentation version 7.6.5

Any technical documentation that is made available by Alliance ATE Consulting Group is the copyrighted work of Alliance ATE Consulting Group and is owned by Alliance ATE Consulting Group.

NO WARRANTY. The technical documentation is being delivered to you AS-IS and Alliance ATE Consulting Group makes no warranty as to its accuracy or use. Any use of the technical documentation or the information contained therein is at the risk of the user. Documentation may contain technical or other inaccuracies or typographical errors. Alliance ATE Consulting Group, Inc. reserves the right to make change without prior notice.

No part of this publication may be copied without the express written permission of Alliance ATE Consulting Group, 3080 Olcott St Suite 110C, Santa Clara, CA 95054.

TRADEMARKS

Velocity CAE Program Generator, ShellConstructor are trademarks of Alliance ATE Consulting Group.

TABLE OF CONTENTS

Page #

Copyright Notice i
Trademarks i
GENERAL INFORMATION 1
VELOCITY CAE ARCHITECTURE 2
Default Text editor 2
LOG FILE STORAGE 2
User Interface 4
General Overview and Usage 4
Selected Outputs 6
Write Pattern, Timing, etc 7
Create Tester Setup Files 7
Create Tests and Flow 7
Create Verilog Files 7
Compile Output Files for Target 8
Append Mode 8
Debug Mode 8
Build Options 8
Optimization Level 9
Normalize Timing (AC specs track with periods) 10
Enable Scan Mode 10
Import AC/DC Test Setups (Functional Tests only if disabled) 10
Data Bit Rate 10
Input Clock Edges Per Output Data Strobe 10
Enable Snapping 11
Output Files 12
Program 12
Patterns 12
Jobs 12
Tester 12
Procedures 12
STIL 12
Code 12
Binary 12

Verilog 12
PRE-DEFINED DEFAULT VALUES 13
SERVER 13
VCDPAGE 13
PAGE 14
SCALE 15
LIBRARY 15
BUS 16
CAP 17
METHOD 17

MEMORY 21
MODEL 21
OPTION 22
Command Line Usage 23
Usage 23
Options 24
Universally Available options 24
J750 and J973 specific options 25
VCD/EVCD Specific options 25
WGL Specific Options 25
Appendices .....
26 Source File Types .....
26

GENERAL INFORMATION

The purpose of this document is to describe the purpose and usage of the Velocity Toolkit. This documentation is included as part of the release of the tool kit and includes detailed descriptions of the following subjects:

- The architecture of the tool as it integrates with other existing tools or programming methods
- General Usage definition
- Selected Outputs
- Build Options
- Output File Organization
- Command line execution

VELOCITY CAE ARCHITECTURE

Velocity is a tool that is designed to facilitate quick creation of compatible, ready-to-use test programs for a number of ATE models. The generated programs will have a minimum set of features that will provide the user with quicker access to methods for executing pre-defined test sequences, copying reusable source files, and packaging local versions of pattern files as part of the test program so that these files can be manipulated without fear of destroying information that exists in the production or database versions of these files.

Tester files and program source files will be created and each pattern will be created as a standalone representation so that incremental compile can be used for multiple pattern test programs.

DEFAULT TEXT EDITOR

The default text editor that isgoing to be “nedit” for Linux systems. If this is not available or a user simply wants to use a different editor, the following environment variable can be used to reset what is used

VELOCITYEDITOR = ‘alternateEditorCommand’

This alternateEditorCommand can be “vi”, or “gvim” or any other editor that is wanted. The syntax for setting this variable depends on the shell type that is chosen. You can ask your local IT for support on how to set this. Examples are as follows:

setenv VELOCITYEDITOR=gvim

or

export VELOCITYEDITOR gvim

LOG FILE STORAGE

By default, log files are always saved in the directory from which the Velocity CFG is loaded. Additionally, a globally exported coy is sent to “/usr/local/AllianceATE/common”. If this directory is blocked from write priveleges, the user can redirect this elsewhere by defining the “HOME” environment variable to point somewhere else

HOME = ‘somePath’

Examples are as follows:

setenv HOME=/home/velocityUser

or

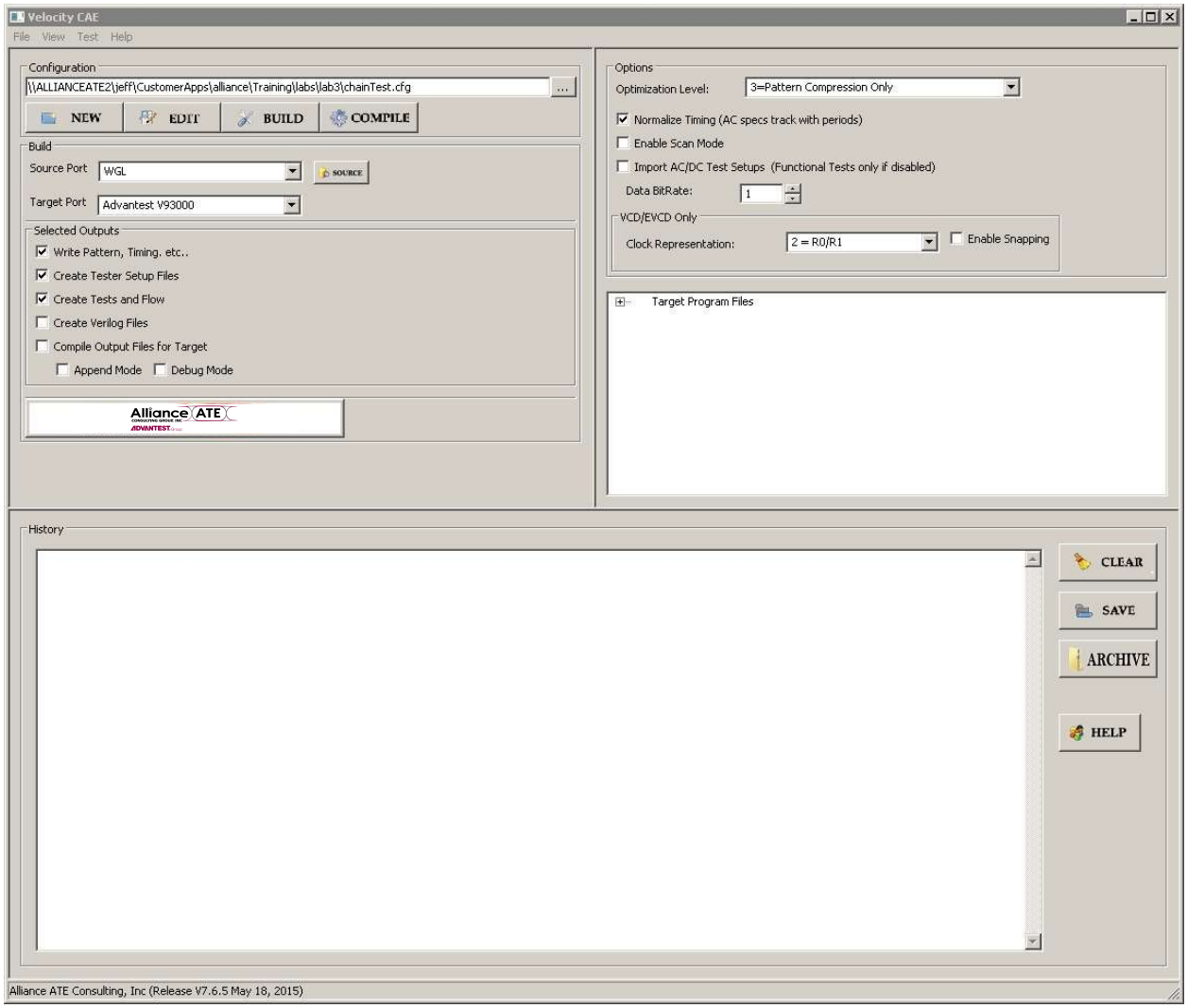
export HOME /home/velocityUser

USER INTERFACE

The GUI simplifies some aspects of usage for those who do not require the scripted control of the command line programs. All of the options available to the command line are available from the GUI. In addition, some other features are available to speed up the program creation process.

General Overview and Usage

All operations of Velocity CAE can be handled through the GUI. These actions include loading and building configuration files, editing these configuration files, building and compiling target files, viewing and even editing target files.

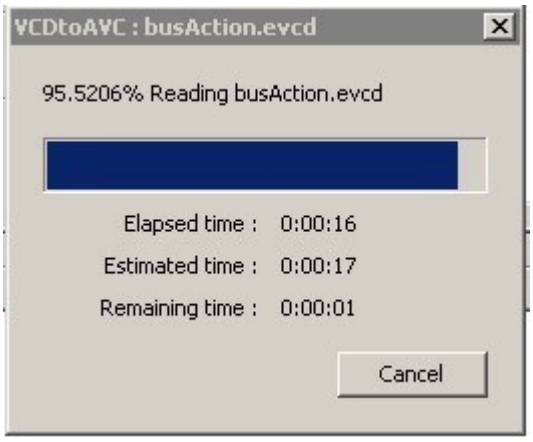


The window itself is organized into 5 sections as follows

- Configuration: This section displays the active configuration that is loaded. This also contains all of the buttons that initiate Velocity actions.
- Build: In this block, there are pulldowns for selecting the active source and target file types. Additionally, there are check boxes that will enable/disable certain output files content.
- Options: The section contains pulldowns and check boxes that are used to enable/disable commonly used features to tune the way output files are created.
- Target Program Files: This is a tree View that will give the user quick access to various output files for editing and or modification.
- History: This is a text field that shows what has been executed as well as any warning or error messages.

The ability to generate new configuration files automatically from any of the defined input sources is available. These CFG files can also be edited from within the GUI.

Output files can be viewed easily from the tool to allow simple edits to any of the ascii files prior to compilation.

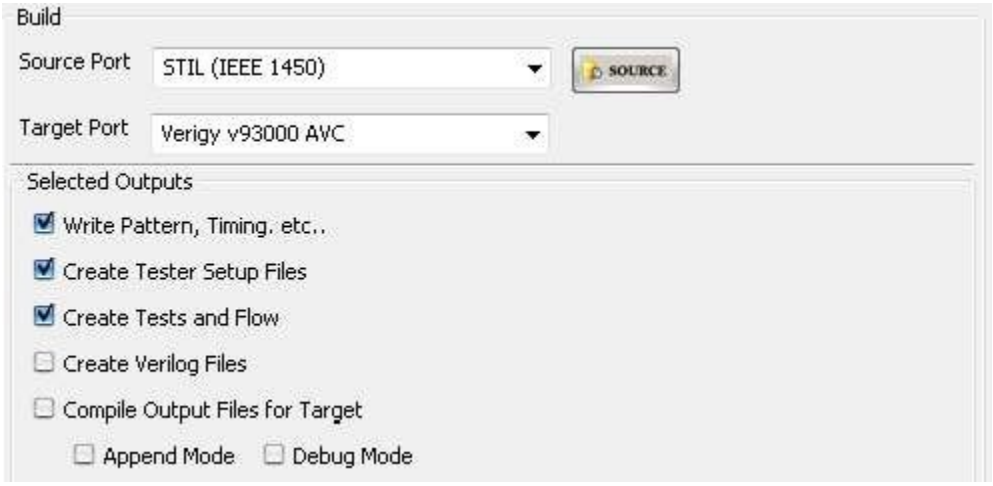


The progress indicator displays in terms of percentage complete to give the user a more accurate idea of the time remaining in a given translation.

For a step by step recipe on using the GUI, please refer to the Velocity CAE Quick Start Guide.

Selected Outputs

Various output files and content can be enabled and disabled with check boxes located in the middle left box of the Velocity GUI. These outputs and the effects of enabling or disabling these fields is detailed here



Write Pattern, Timing, etc.

(Enabled By default) This will enable the export of all ascii pattern and timing files. When disabled, other enabled files will be exported, but the patterns and timings will be blocked. Generally you would never need to disable this, but if you wanted to update a testflow without touching the underlying patterns and timing you could disable these.

Create Tester Setup Files

(Enabled By default) This will enable the export of all setup files that are required for either loading or compiling the resulting binary files. Setup files are the files that are used as inputs to the compilation process separate from the actual timing and pattern files. These are like the makefiles for a C++ compilation. Once again, you would typically never run with this deactivated. If you want to perform a custom compilation that differs from Velocity's typical supported formats, you can disable this feature and build your own setups. This is not recommended however.

Create Tests and Flow

(Enabled By default) This will enable the export of the testflows that will be used on the target to load and execute the files created by the Build and Compile buttons. In some cases, the testflow itself is not needed. The user may have a pre-defined flow and may not care about the flow that Velocity automatically generates. You can disable this feature here.

Create Verilog Files

Velocity has a powerful feature where Verilog testbenches can be automatically created for every pattern. This is activated by enabling with this check box. When enabled, every translated pattern will also export a *.v (simulation testbench) and *.evcd (viewable EVCD) that can be sent back to designers or test groups to verify that the resulting ATE patterns and timing are valid. Enabling this can increase the time required for translation as these files tend to be quite large.

Compile Output Files for Target

If this box is checked, then after the Build process is done to create all ascii ATE files the binary compilation tool will be automatically called. This is essentially equivalent to clicking Build and then separately clicking Compile. It is a simple time saver that allows you to do both actions with a single click of Build.

Append Mode

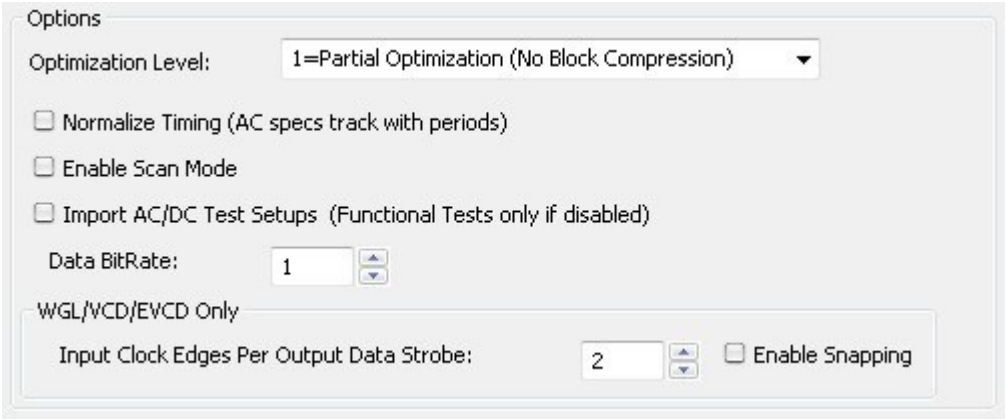
Enabling Append Mode will allow you to add new information to an existing program. When disabled (as it is by default), the target files and compilation setup files are cleared so they can all be rebuilt. The most important effect of Append Mode is on the resulting timing. In Append Mode, the timing is NOT recompiled. Therefore any new patterns will be compiled assuming the timing that already exists. Ascii timings will still be updated as needed, but the binary compilation of this timing will be blocked in Append Mode. Binary compilation of patterns will proceed as always. But, you may see compile errors that will let you know that the old timing is not compatible with the new patterns. When this happens, all you need to do is disable Append Mode and compile again. If Append Mode is disabled, binary timings will always be rebuilt.

Debug Mode

This option will instruct Velocity to only translate a small portion of each pattern that is chosen. This is helpful when checking out new patterns to verify that the source files are valid.

Build Options

This section contains boxes and dropdowns that are used to tune the way that various output files are exported



Optimization Level

This drop down is used to define the timing and pattern optimization level that will be used.

Timing optimizations refer to whether common device cycles are maintained for each pattern or whether separate device cycles are used for each pattern.

Pattern optimization refers to whether compression is enabled or disabled. When enabled, Velocity will automatically search for single and multi-line repeats that can reduce vector depth and increase vector readability.

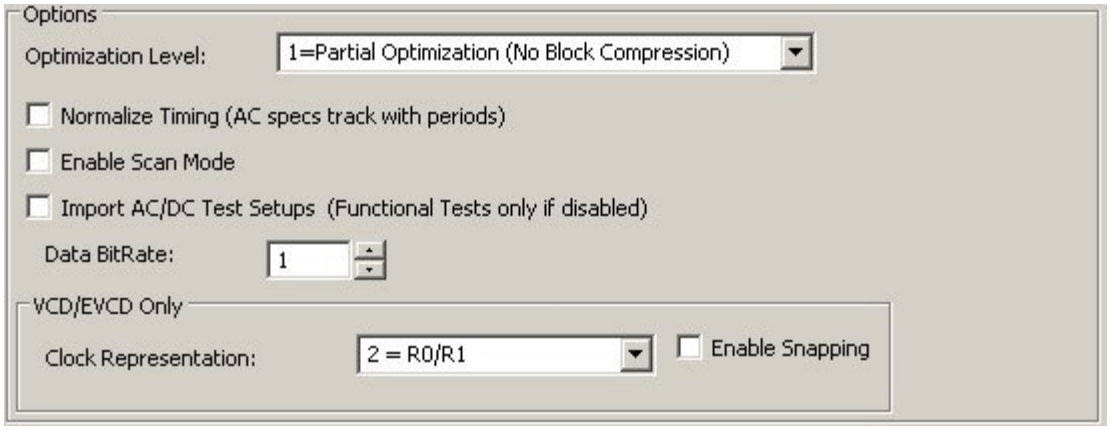
There are four separately controlled levels of optimization.

Level 0: No timing or pattern optimizations. Each pattern will have its own separate device cycles. No pattern compression will be employed on the patterns. Additionally, any existing loops that may have been present in the source files will be expanded.

Level 1: (default level) Timing optimization is enabled which will allow Velocity to create a common device cycle setup that can be used on all patterns that are being translated. This results in the cleanest and easiest-to-read timing. But, in some cases, you can lose edge variations that might need to exist from pattern to pattern. Patterns will not be compressed either. But, any loops that exist in the source files will be left intact

Level 2: This is what is called “full optimization”. Timings are optimized as described in Level 1, which results in the simplest and cleanest timings. Additionally, logic is enabled that will search for and perform compression on the patterns, allowing loops and repeats to be inserted where possible. This level results in the cleanest and smallest-possible timings and patterns.

Level 3: This will perform all pattern compression as described in Level 2, but the timings will be left un-optimized. This level would be used when you want to compress patterns but you still need to have each pattern retain its own unique timing.



Normalize Timing (AC specs track with periods)

By default, this is disabled. When disabled, timing edges will be imported exactly as they are defined in the source. If defined with discrete edges as they are in VCD files, then the resulting timing will use discrete edges. If the source file uses equations and specs for timing, then these equations will be passed exactly as they are defined. When normalization is enabled, all edges will be evaluated and automatically expressed in simple equations that will allow edges to easily track with a period spec variable.

Enable Scan Mode

By default, all scan patterns are translated and flattened into regular, parallel vector files. Scan information may have been evaluated during the translation process to build the parallel representation for the ATE, but that information will not be exported to the target. When this option is enabled, Velocity will automatically export additional files that can then be used by the test program to provide additional log information such as scan instance names and scan cell number information.

Import AC/DC Test Setups (Functional Tests only if disabled)

When translating from an existing test program, there may be additional DC and AC test information available, such as force values for DC tests, AC spec settings, etc.. When this option is enabled, all of that information will be imported from the source and exported to equivalent functionality in the target, when available. In many cases, simple functions like leakage, continuity, and VIL/VIH tests can be directly mapped to existing canned functions. In other cases, custom tests appear in the source that will not map directly to any pre-existing function. When this happens, the arguments and setups are exported anyway, but there will also be warnings that will let the user know that manual intervention to assign content to nonexistent functions will need to happen.

Data Bit Rate

This field will allow the user to define the data bit rate that is to be used by the target compiler. Additionally, this will be used to tune certain optimization features. For example, pattern compression must be done so that loop sizes and length are a clean modulus of the data rate. So, this does have some effect on ascii files as well as the resulting binaries. You must make sure that you use the same Data Bit Rate for the Build process that you use for the Compile process.

Input Clock Edges Per Output Data Strobe

(ONLY used by VCD/EVCD translations) Output data is often dependent on clock edges. Sometimes strobes will happen on only one edge of a clock -- either the rising or falling edge. Some devices require active strobing on BOTH edges of the clock. This option will affect the way serial data from a VCD/EVCD is cyclized. If you want to strobe on both edges of the clock that means that 1 clock edge = 1 strobe edge. Therefore, you would assign this value to “1”. If you only want to strobe on one edge of the clock, that means that 2 clock edges = 1 strobe edge. Therefore, you should assign edges = 2.

This will then affect how input clocks are expressed in ascii patterns and timings.

The general rule is as follows:

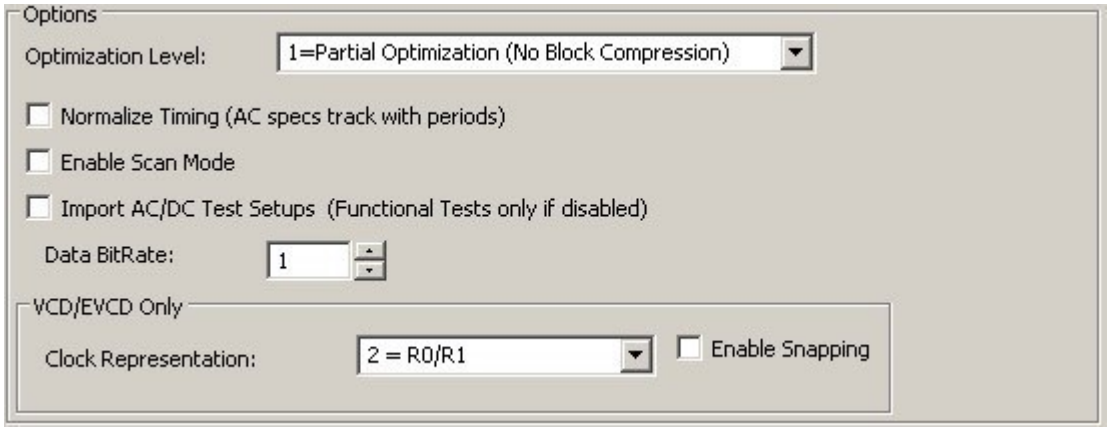
1 Edge = NRZ clock data

You will see”1010101010..: in resulting clock data columns.

2 Edge =R0/R1 clock data

You will see “CCCCCC…” for R0 clock data.

You will see “ccccccccc…” for R1 clock data.



Enable Snapping

Sometimes there can be asynchronous actions in a pattern. When the asynchronous behavior is very slow compared to the sample rate, these variations will be automatically handled in such a way that ATE resources are not exhausted.

When asynchronous behavior is happening at a higher rate, this correction cannot be made as cleanly. There are essentially two options. First, you can modify your CFG PINLIST and create multiple ports or multiple time domains. That way asynchronous pin groups can be cyclized separately. The other option is to enable snapping. This is risky and intended to be a last resort.

When snapping is enabled, all edges will be blindly and automatically snapped to a grid. Any asynchronous behavior will be automatically corrected. This can fundamentally change behavior.

It is very useful to couple the enable of snapping with the usage of the Verilog feedback. You can then resimulate with the corrections applied to see if any of this movement of edges causes DUT problems

Output Files

All output file are viewable from this Target Program Files tree view. Double clicking on any file in any of these folders will automatically open the appropriate view/editor for the given file type

Program

Any collective top level program file that exists

Patterns

ASCII pattern, timing, and scan cell information

Jobs

ASCII compile setup files, compile logs, Data combination files

Tester

All files that are loaded on the tester. Some of these are outputs of compilation (patterns and timing). Others are automatically created by Build (testflow, levels, pin files)

Procedures

(only for STIL targets) STIL procedure calls are stored here

STIL

(only for STIL targets) common STIL files that are accessed from multiple patterns are stored here

Code

Any C/C++ code that is generated for the target. Most targets will not create this. If nothing is created, this folder will be empty. That is by design.

Binary

Some targets will produce binary data for patterns. These files can be viewed here. The viewer will be a text editor, but in some cases this is still helpful

Verilog

If Verilog Feedback is enabled, these files will be placed here. You can view the testbench in any text editor. The EVCD file can be viewed in a graphical tool that is also a part of the Velocity tool library

PRE-DEFINED DEFAULT VALUES

All variables that can be definee in the the Velocity Configuration file for a given device have default values when they are left out of the Configuration. Some, but not all, can also be pre defined in a system level file called the "Velocity Libraries" file. This file is located in the "common" directory where Velocity was installed. By default, this installation location is "/usr/local/AllianceATE", but this can be altered by the installer if needed. The "common" directory will be under underneath this.

The file itself will be a hidden file named ".velocity_libs". Variables that are entered here will apply new default states for all users on the system. Each variable that is legal in this file and its usage is described in this section.

SERVER

Velocity licenses can either be node locked or server based. If a floating/server based licensing scheme is employed then this libs file variable is required to tell the client where to find the server and optionally which port number to use for socket communication

Syntax:

SERVER serverName|IP_address [portNumber]

where, serverName = name of remote machine where license server is running IP_address = alternately, the server can be referenced via IP address
portNumber = optionally, the port number for socket communication can be specified here. If not defined, then it will look through port number 3490

Examples:

```
SERVER      192.168.1.55

SERVER      velocityServer      1000
```

VCDPAGE

This allows you indirectly control the size of pages that are used during a VCD translation. The value is in percentage. This default to 100% but you can increase or decrease the page size depending on this value. This is used to limit the size of loops when made smaller or remove page boundary issues by making the page larger.

The default page value is 100%. The page is then calculated based upon the number of pins and the density of activity. This is not something tht is specifically controlled to a fixed value. This is a general property that can be used to give you larger or smaller page sizes relative to the base.

Syntax

VCDPAGE value

Examples:

```
VCDPAGE     10  # Use smaller page

VCDPAGE     500 # Use larger page
```

PAGE

This directive is used to redefine the number of scan instances that will be included in a single vector file. By default, scan patterns will be broken into separate files that cab be bursted together. The reason that these are broken up is because it can make processing and debug easier in that you can mask certain chunks of patterns to make loading quicker.

But, since some patterns don't like the way that bursts are issued, this flag will allow you to make the page size larger or smaller to increase or possibly remove entirely, the need for paging.

PAGESIZE will also be used to define the seed size to use for the paging calculation for VCD/EVCD translation. By default a seed of 1000 cycles is used for this calculation. In some cases, you may want to change the size of pages to address paging issues caused by staggered busses or asynchronous behavior in simulations.

Syntax:

PAGESIZE numberPerPage

Example:

```
PAGESIZE    10000  # This will break file every 10,000 scan instances    # Or a seed value of 10000 for VCD/EVCD files
```

```
PAGESIZE        50    # This will break file every 50 scan instances        # Or a seed value of
                50 for VCD/EVCD files
```

SCALE

This variable is used to toggle automatic scale corrections for platforms that will not automatically scale these for you. Advantest 93K and Teradyne platforms will scale for you so this variable is redundant and therefore unlikely to ever be needed.

If scaling is requested for a target plaform that requires this, then all edges will be scaled to the appropriate level that architecture rules are not violated.

By default, SCALE is OFF.

Syntax:

SCALE ON|OFF

where,

ON = scaling will be done automatically

OFF = scaling will not be done

Examples:

```
SCALE ON

SCALE OFF
```

LIBRARY

This variable is used to define the name and location of any predefined libraries that you want to have included in the target program. This is only used by platforms that require specific user defined libraries to be pre defined. 93K users are unlikely to ever need this variable.

Syntax:

LIBRARY EXTERNAL|LOCAL LIBRARY_FILE_NAME

where,

EXTERNAL|LOCAL directs Velocity whether to physically copy this library to the target program or simply refer to its path through a makefile or some other method depending on the target.

LIBRARY_FILE_NAME is the path and name of the library file itself Examples:

HEADER	LOCAL /usr/local/lib/someLibrary.so	
HEADER	EXTERNAL	/usr/local/lib/ someLibrary.so

NOTE: referenced library file must be present and valid for the target

BUS

This variable is used to toggle automatic removal of brackets from pin names when generating new Configuration files. Additionally, this variable is used to determine whether Verilog loopback will express pins in busses or whether these will be flattened into singular instances for each pin. If BUS is enabled, brackets will be included in when configurations are used and busses will be used when exporting to Verilog. If BUS is disabled, brackets will be removed from the tester pin names and pins will be expressed individually in Verilog feedback files.

By default, busses are enabled.

Syntax:

BUS ON|OFF

where,

ON = Brackets in pin names will be used and Bus expression used in Verilog

OFF = Brackets removed from pin names and individual pin entries will be used in Verilog

Examples:

```
BUS ON
```

```
BUS  OFF
```

CAP

This variable allows the user to toggle a feature where pin names from the simulation are automatically converted to all caps when creating a new CFG. When enabled, pin names will automatically convert all lower case letters to upper case. When disabled, pin names will be left as they are defined in simulation or whatever source file is used to create the CFG's PINLIST

By defaut, CAP is enabled.

Syntax:

CAP ON|OFF

where,

ON = tester pin names will be ALL CAP

OFF = tester pin names will be left as they are

Examples:

```
CAP ON
CAP  OFF
```

METHOD

Specific to the Advantest 83K and 93K ports.) Specifies the type of test method to be used: Classic (CTM) or Universal (UTM). Used for generating the appropriate format for the test flow file.

Syntax:

METHOD CTM|UTM

Example:

```
METHOD          CTM     # Classic Testmethod
```

```
METHOD      UTM    # Universal Testmethod
```

MEMORY

Compilation to the 93K can be done using Vector Memory or Sequencer Memory. By default, vector memory is used. If you want to only use sequencer memory you can override using this directive .

Syntax:

MEMORY SM|VM

Example:

```
MEMORY      SM    # sequencer memory

MEMORY      VM    # vector memory
```

MODEL

Syntax:

MODEL modelType

where, modelType is a tester model type. This Control is specific to the Advantest 83K and 93K tester ports. Valid entries are:

- F330
- C400
- P1000 □ PS400
- PS800
- PS6800
- PS3600

By default MODEL = PS6800, which is the SmartScale systems.

Examples

```
MODEL       PS3600   # PinScale

MODEL       P1000              # Single Density 93K

            PS6800             # SmartScale
```

MODEL

OPTION

This variable is used to define the default options used by the 93K binary pattern compiler. These are the options that have consistently provided the optimal balance of speed and readability.

Syntax:

OPTIONS ‘V2B_OPTIONS’

where,

V2B_OPTIONS is a srting that will apply pattern compilation options.

Examples:

OPTION -MU -FPZgux

COMMAND LINE USAGE

Usage

There are multiple command line translation paths available. Each contains a similar set of options, but some have options that are specific to the platform being extracted. For example WGL and EVCD have no concept of time sets and spec sets. Therefore these programs do not have options to set these. By default, the usage is:

velocity -translationTypeSwitch [options] configFile [sourceFile [sourceFile2 sourceFile3 …]]

where, options are listed and desctibed in detail in the Command Line Options

section below

translationTypeSwitch is the directive that tells velocity which translation engine to use. The allowable values for this are as follows in the Translation Types section

configFile is the velocity configuration file. See the Velocity Configuration Guide for a detailed description of the contents and usage of this file

sourceFile is a list of source files that will depend on the value of the translationTypeSwitch. The section Source Files will completely define the requirements and limitations of this field of arguments

Translation Types: This option is essentially formed by combining various source and target ports in a single string as in “SOURCEtoTARGET”. Available sources and targets will be dependent on the licenses that any given customer has activated. The complete lists are in the following two charts

SOURCE PORTS TARGET PORTS

Options

A set of command line directives is available that will allow for various optimizations to the output. These options are defined below

Universally Available options

+/-o

Turns on and off optimization of output. Specifically, the timing and levels blocks that are included in the source files but are not used by the listed patterns will be removed from the output program. This facilitates smaller test programs that will compile and load much faster

+/-s

Turns on and off the output of new STIL files. If –s is used, the previously created output files will be reloaded and left untouched in the current Shell creation. Therefore, these STIL files will need no recompilation.. +s will ensure that the STIL files are recreated

+/-t

Turns on and off the output of all tester files for the D10 program. This includes all non-STIL and nontest program files (*.cpp *.h). –t will mask the generation of these files. +t will ensure recreation or new JOB, SIG, MAKE, etc.

+/-p

Turns on and off the generation of D10 test program files. –p will prevent existing *.h and *.cpp files from being overwritten. + p will create new source files.

+/-c

Turns on and off the automatic binary compilation of the target program.

+/-n

Turns on and off auto normalization of timing. When used a spec set will used that will allow global and waveform table specific scaling of timing values. This is an easier method of providing period scaled timing than the Custom Timing blocks

+a

Turns on “append mode”. When in use, this feature will automatically add new patterns, timing, etc. to the existing program without overwriting and/or removing existing data.

+q

Turns on “quiet mode”. When in use, this feature will automatically block the progress meter GUI so that the translations can be run remotely without needing to export a display

+h

Prints out the complete usage man page to screen

J750 and J973 specific options

+ac=/+dc=

Allows a global override for the spec categories that are used for the AC and DC tests respectively. Useful if a specific timing is need for all tests regardless of what is used in the source program

+ts=/+ls=

Allows manual override to redefine globally the timing or leves set/sheet that is used for all tests regardless of what is predefined by the source program

VCD/EVCD Specific options

+xN

Define the maximum number of databits to be defined in a single data cycle. This will automatically adjust the waveform tables to allow more than one transition with in a single cycle. N=1 by default is option is not used

+eN

Adjust the resolution for edge snapping. Drive edges will snap forward, Receive edges will snap back. N will determine how many regions are used. By default, N=1, which means that there will be a single Drive edge and a single receive edge. N=4 will divide the period into 4 equal regions. Edges will snap into based on the region within the period the raw edge falls into.

WGL Specific Options

+/-m

Enable or Disable the usage of scan macros for representation of scan instances. If enabled, STIL syntax which uses macros will be implemented so that scan instances are defined as serial scan. If disabled, the scan information will be converted to standard parallel vector.

APPENDICES

Source File Types

Source files listed on the command line will depend on the value of the translationTypeSwitch. From the command line the source port and target port are defined with a string of the form:

‘SourcePorttoTargetPort’

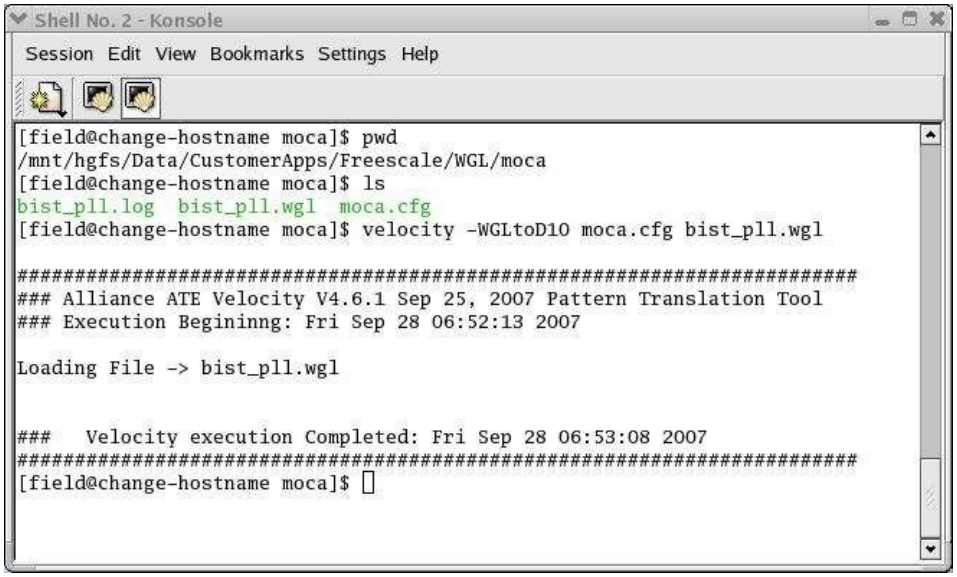
This will always be the first argument of the velocity command line call.

Translation Source Port	File List Description
IEEE STIL	*.STIL and *.stil files
Verigy 93K AVC	*.dvc files to define the timing; *.avc to define the patterns
Teradyne J750	*.xml files to define program, levels, and timing; *.atp files to define the patterns
Teradyne J973	*.pinadr to define pin lists; *.waveadr to define waveform tables; *.specadr files to define timing spec values; *.lvmadr files to define patterns; optional *.svmadr to define subroutines; optional *.ts to redefine timset to waveform table mappings
WGL	*.wgl files to define pins, specs, timing, and patterns
VCD/EVCD	*.evcd files to define timestamp based events

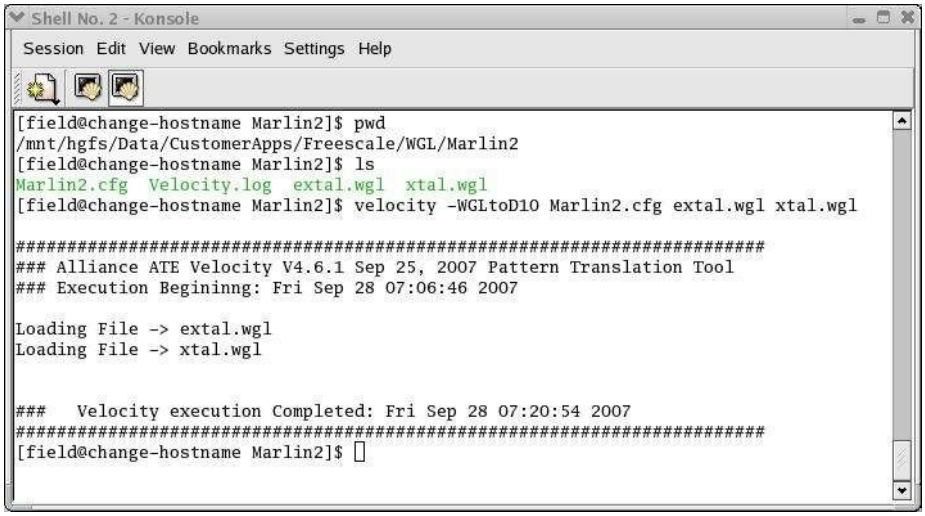
If used properly, each program will create the target program files in the directory defined in the configuration. The following figure shows a successful usage of the Velocity CAE command line program.

There are a number of usage examples that will result in a variety of errors. The following are examples of common errors

- Non-Existent or misspelled configuration file name
- Non-existent or misspelled source files
- Missing references caused by include statements in source files that can't be resolved.
- Having no write privileges to the directories defined by configuration file



The previous example translates a program that uses only a single pattern. You can include multiple patterns by simply including all relevant patterns on the command line. The next example shows the results when multiple valid patterns are included in a single translation.


Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help
[field@change-hostname Marlin2]$ pwd
/mnt/hgfs/Data/CustomerApps/Freescale/WGL/Marlin2
[field@change-hostname Marlin2]$ ls
Marlin2.cfg Velocity.log extal.wgl xtal.wgl
[field@change-hostname Marlin2]$ velocity -WGLtoD10 Marlin2.cfg extal.wgl xtal.wgl
Alliance ATE Velocity V4.6.1 Sep 25, 2007 Pattern Translation Tool
Execution Begininng: Fri Sep 28 07:06:46 2007
Loading File -> extal.wgl
Loading File -> xtal.wgl
Velocity execution Completed: Fri Sep 28 07:20:54 2007
[field@change-hostname Marlin2]$