VELOCITY CAE

TESTING AT THE SPEED OF LIGHT

Velocity CAE® Program Generator

For **Simulation to ATE** and **ATE to ATE Conversion**

Release 5.4.3

Quick Start Guide

for the Verigy 93000

Velocity CAE Program Generator

Quick Start Guide

COPYRIGHT NOTICE

Documentation version 3.0

TRADEMARKS

QUICK START GUIDE

TABLE OF CONTENTS

Page #

Copyright Notice ..... *ii*

Trademarks ..... *ii*

1.0 GENERAL INFORMATION *1-1*

1.1 What is Velocity CAE? **1-1**

1.2 Running Velocity CAE **1-1**

1.2 What are the tools and components that make up Velocity CAE? **1-1**

1.3 What Design-to-Test Tools are available? **1-2**

1.4 What Tester-to-Tester Tools are available? **1-2**

1.5 What kinds of files do Velocity CAE converters create? **1-4**

2.0 RUNNING VELOCITY CAE (COMMAND LINE) *2-1*

2.1 Using Configuration Files **2-1**

2.2 Design-to-Test Conversion Options **2-1**

2.2.1 Optimize Output 2-1
2.2.2 Create Tester Setup Files 2-2
2.2.3 Compile 2-2
2.2.4 Normalize Timing 2-2
2.2.5 Append 2-2
2.2.6 Quiet Mode 2-2
2.2.7 Feedback 2-3
2.2.8 Data BitRate 2-3

3.0 VELOCITY CAE CONVERTERS (COMMAND LINE) *3-1*

3.1 Running the WGL Converter **3-1**

3.1.1 Configuration File Settings 3-1
3.1.2 WGLtoAVC Conversion Options 3-1
3.1.3 Running WGLtoAVC 3-1

3.2 Running the VCD/EVCD Converter **3-2**

3.2.1 Configuration File Settings 3-2
3.2.2 VCDtoAVC Conversion Options 3-3
3.2.3 Running VCDtoAVC 3-4
3.2.4 Verifying Correct Cyclization, Using the Cycled VCD Output 3-4

3.3 Running the VCT Converter **3-4**

3.3.1 Configuration File Settings 3-5
3.3.2 VCTtoAVC Conversion Options 3-5
3.3.3 Running VCTtoAVC 3-5

4.0 RUNNING VELOCITY CAE (GUI-Based) *4-1*

4.1 Using Configuration Files **4-1**

4.2 Running a Design-to-Test Converter 4-1

4.3 Design-to-Test Conversion Options 4-5

4.3.1 Optimization Level 4-5
4.3.2 Write Pattern, Timing, etc. 4-5
4.3.3 Create Tester Setup Files 4-5
4.3.4 Compile Output Files for Target 4-5
4.3.5 Normalize Timing 4-6
4.3.6 Append Mode 4-6

5.0 VELOCITY CAE CONVERTERS (GUI-BASED) 5-1

5.1 Running the WGL Converter 5-1

5.1.1 Configuration File Settings 5-1
5.1.2 WGL Conversion Options 5-2
5.1.3 Running WGL Conversion 5-2

5.2 Running the VCD/EVCD Converter 5-2

5.2.1 Configuration File Settings 5-3
5.2.2 VCD Conversion Options 5-3
5.2.3 Running VCD Conversion 5-4
5.2.4 Verifying Correct Cyclization, Using the Cycled VCD Output 5-4

5.3 Running the VCT Converter 5-5

5.3.1 Configuration File Settings 5-5
5.3.2 VCT Conversion Options 5-5

6.0 CONFIGURATION FILES 6-7

6.1 (Optional) Automatically Generate an Initial Configuration File 6-7

6.2 Understanding the Configuration File Structure 6-8

6.3 Create (or Edit) the Program Path 6-9

6.4 Create (or Edit) the Cyclization Timing 6-10

6.5 Create (or Edit) the Pinlist 6-11

6.6 Create (or Edit) Custom Timing 6-12

6.8 Customizing Patterns 6-13

1.0 GENERAL INFORMATION

1.0 GENERAL INFORMATION

A brief look at the various elements and capabilities of the Velocity CAE Program Generator and Velocity CAE Design-to-Test Tools.

1.1 What is Velocity CAE?

Velocity CAE (Computer Aided Engineering) is a suite of software tools for automatically generating ATE patterns, timing, and other test program files from either a simulation output or files from other ATE platforms.

The following sections of this overview will introduce all the various components of Velocity CAE while the rest of this manual will be focused on the Velocity CAE's Design-to-Test Tools.

1.2 Running Velocity CAE

Velocity CAE can be run from either the command line (shell) or a GUI (Graphical User Interface). The GUI makes it extremely easy to make lightning fast conversions without the pain of customizing each run. However, the command line will give power users the ability to customize their conversions and run them as a batch process.

For details on running Velocity CAE from the Command Line, see ***2.0 Running Velocity CAE (Command Line);*** or, for running Velocity CAE from a GUI, see ***4.0 Running Velocity CAE (GUI)***.

1.2 What are the tools and components that make up Velocity CAE?

Velocity CAE for the Verigy 93000 consists of the following major components:

Component	Description
GUI	The graphical interface to all Velocity CAE functionality
Core Engine Analyzer	Central processing "hub" of Velocity CAE. Required for all Design-to-Test and Tester-to-Tester conversion tools.
Design-to-Test Tools	Convert simulation output to tester patterns and timing
Tester-to-Tester Tools	Convert pattern and timing data from one test platform to another.

1.3 What Design-to-Test Tools are available?

Velocity CAE for the Verigy 93000 offers the following Design-to-Test Tools:

Tool	Description
WGLtoAVC	Converts patterns and timing from WGL (Waveform Generation Language) files to Verigy's AVC and DVC format along with files needed for Verigy's ASCII translator
VCDtoAVC	Converts patterns and timing from VCD (Verilog Change Dump) files to Verigy's AVC and DVC format along with files needed for Verigy's ASCII translator
VCTtoAVC	Converts cycled patterns from Verilog .vct output files and timing from RCONFIG files to Verigy's AVC and DVC format along with files needed for Verigy's ASCII translator
STILtoAVC	Converts any STIL based pattern representation into Verigy's AVC and DVC format along with files needed for Verigy's ASCII translator

1.4 What Tester-to-Tester Tools are available?

Velocity CAE offers the following Tester-to-Tester Tools:

Tool	Description
J750toAVC	Converts patterns and timing from Teradyne J750 files to Verigy's AVC and DVC format along with files needed for Verigy's ASCII translator
J973toAVC	Converts patterns and timing from Teradyne J973 files to Verigy's AVC and DVC format along with files needed for Verigy's ASCII translator
FLEXtoAVC	Converts patterns and timing from Teradyne UltraFlex files to Verigy's AVC and DVC format along with files needed for Verigy's ASCII translator

Tool	Description
SAPHtoAVC	Converts patterns and timing from Credence Sapphire files to Verigy's AVC and DVC format along with files needed for Verigy's ASCII translator
S9KtoAVC	Converts patterns and timing from Schlumberger S9000 files to Verigy's AVC and DVC format along with files needed for Verigy's ASCII translator
ASLtoAVC	Converts patterns and timing from Credence ASL files to Verigy's AVC and DVC format along with files needed for Verigy's ASCII translator
AVCtoAVC	Processes existing AVC and DVC patterns and timing from the Verigy 93000 to new AVC and DVC files, making use of Velocity optimization features

1.5 What kinds of files do Velocity CAE converters create?

Depending on the options you select for the conversion and the targeted output format, Velocity converters can create all the setup and pattern files required to support a fully-functional test program. These generated files include:

Setup Files

- **.pin**
- **.tim**
- **.binl**
- **.pmfl**
- **.aic**

Pattern Files

- **.avc**
- **.dvc**

2.0 RUNNING VELOCITY CAE (COMMAND LINE)

2.0 RUNNING VELOCITY CAE (COMMAND LINE)

Information on how to configure and start up Velocity CAE Design-to-Test Tools, and how to set common execution options.

2.1 Using Configuration Files

All Velocity CAE tools require a configuration file, which is a human-readable ASCII text file that you create. This file will customize the Core Analyzer Engine for specific conversions. For more information on understanding, creating, and editing Velocity configuration files, refer to the section in this guide called "Configuration Files," or for more details see the *Velocity CAE Configuration Guide*.

To start a new conversion without customization you can run Velocity CAE with a blank configuration file.

2.2 Design-to-Test Conversion Options

All Velocity tools, including the Design-to-Test Converters, support a set of command-line execution options. Most of the options are common to all of the tools. The following section lists the common options for the Verigy 93000.

2.2.1 Optimize Output

Enables the performance of various optimizations on the conversion output.

If this option is turned on, Velocity will perform one of two levels of optimization, depending on the number specified after the letter 'o'. It will also enable the use of other optimization options that can be individually turned on or off. (Those additional optimization options are described later in this guide, especially the XMode and Snap Resolution options that are specific to conversions from VCD/EVCD format.)

Command-line:

+o1	(Default) Turn partial optimization on: do not create new repeats and loops, but enable other optimization options.
+o2	Turn full optimization on: create new repeats and loops if possible.
+o	Same as +o1.
-o	Turn all optimizations off. If any repeats and loops exist in the source pattern, they will be "flattened" into straight, sequential vectors.

2.2.2 Create Tester Setup Files

Enables the converter to create new test setup files for the 93000 program, such as ?, etc.

Command-line:

+t Turn tester setup file creation on
-t Turn tester setup file creation off

2.2.3 Compile

Enables Velocity to automatically run AIT and AIC on the generated files immediately after conversion.

Command-line:

+c Turn automatic test program compilation on
-c Turn automatic test program compilation off

2.2.4 Normalize Timing

Enables Velocity to automatically create a period-scaled timing set.

Command-line:

+n Turn auto-normalization on
-n Turn auto-normalization off

2.2.5 Append

Enables the converter to add new patterns, timing, etc. to an existing program, instead of overwriting the existing data.

Command-line:

+a Turn appending on

2.2.6 Quiet Mode

Enables Velocity to operate in the background, with no pop-up status reporting. This is especially useful for running conversions remotely.

Command-line:

+q Turn quiet mode on
-q Turn quiet mode off

2.2.7 Feedback

Enables Velocity to output a VCD file representation of the converted pattern, along with a full simulator test bench, in addition to the selected output format. This is useful for "feeding back" the converted pattern to the simulator to verify that the conversion will work with the device.

Command-line:

+f Turn feedback on
-f Turn feedback off

2.2.8 Data BitRate

Sets the number of data bits per tester cycle after conversion. Corresponds with 93000 X modes. The AIC file generated as part of the conversion will use an X mode of N.

Command-line:

+xN Set number of data bits per tester cycle to N

3.0 VELOCITY CAE CONVERTERS (COMMAND LINE)

3.0 VELOCITY CAE CONVERTERS (COMMAND LINE)

Velocity CAE comes with independent converters for each combination of source file type and target file type. In this way, multiple conversions can be run at once. These converters are run through the Command Line (shell).

3.1 Running the WGL Converter

Information on how to configure and run WGLtoAVC.

BACKGROUND: This section focuses on Velocity settings specific to the WGLtoAVC Converter. For more information on the use of Velocity Design-to-Test Converters and their common settings, refer to the previous section of this guide called "Using Velocity Design-to-Test Tools."

3.1.1 Configuration File Settings

There are no Configuration file settings specific to the WGL Converter. You can use any of the common settings described in the section of this guide called "Creating a Configuration File," or in the *Velocity Program Generator User's Guide*. However, you should note that, since WGL is a cyclized format, the PERIOD and EDGES Control definitions are not required in the Configuration file.

3.1.2 WGLtoAVC Conversion Options

There are no conversion options specific to the WGL Converter. You can use any of the common options described in the section of this guide called "Design-to-Test Conversion Options," or in the *Velocity Program Generator User's Guide*.

3.1.3 Running WGLtoAVC

At a command line prompt in a terminal window, type "velocity -WGLtoAVC", followed by any required command-line options and input filenames, and press ENTER.

The following is an example of WGLtoAVC command-line syntax:

```
velocity -WGLtoAVC +o +s +t +p ConfigFile.cfg
myPatterns.wgl myPatterns2.wgl
```

This example illustrates that you can name multiple WGL files for conversion, listed after the required Configuration file name.

Note the four options – +o, +s, +t, and +p – immediately after the Converter name. Velocity tool command-line options always begin with a + to indicate the setting of the option, or with a – to indicate the unsetting of the option.

Please see the section in this guide called "Design-to-Test Conversion Options" for more information on the meaning and usage of the various options.

Also note the use of a configuration file name on the command-line (myConfigFile.cfg in this example). As stated previously, all Velocity tools require a configuration file.

3.2 Running the VCD/EVCD Converter

Information on how to configure and run VCDtoAVC.

BACKGROUND: This section focuses on Velocity settings specific to the VCDtoAVC Converter. For more information on the use of Velocity Design-to-Test Converters and their common settings, refer to the previous section of this guide called "Using Velocity Design-to-Test Tools."

3.2.1 Configuration File Settings

There are two Configuration file settings specific to the VCD Converter: **PERIOD** and **EDGES**.

The **PERIOD** Control definition is used to specify a target tester period to be applied to the VCD pattern in order to "cyclize" the stream of events, and to assign a port name to be used for multiport.

The **EDGES** definition specifies the maximum number of timing edges to expect within a period. By default, this value is 2. It takes precedence over the Edges Per Data value entered in the GUI or the +eN option entered on the command line.

Both the PERIOD and EDGES Control definitions are optional. Velocity can automatically determine the appropriate tester period from the input event stream. PERIOD definitions are only really required if you wish to define multiple ports (for a multiport application). For more information on how the PERIOD definition is used for defining ports, see the configuration guide for more details.

TIP: A reliable method for ensuring that your PERIOD and EDGES configuration file settings are appropriate for your VCD conversion source is to automatically generate an initial configuration file from the VCD source. Then, you can edit other settings in the configuration file as required.

Before auto-generating the configuration file, be sure to set the Edges Per Data option in the GUI to the desired value of EDGES.

For more information on how to automatically generate a configuration file, refer to the section in this guide called "Creating a Configuration File."

For more information on these Configuration settings, or any of the common settings, refer to the section of this guide called "Creating a Configuration File," or to the *Velocity Program Generator User's Guide.*

3.2.2 VCDtoAVC Conversion Options

There are two conversion options specific to the VCD Converter: **Edges Per Data** and **Snap Enable**.

The **Edges Per Data** option is used for specifying the number of edges to be used per tester period in the converted patterns and timing.

On the command-line, the edge count option is +eN, where N is the number of edges per period.

To set the edges per period from the GUI, enter a number in the Edges Per Data field.

The **Snap Enable** option is used for correcting "imperfect" simulation data in which an edge placement within the tester period might vary from period to period. The option does so by specifying a number of equal-size snap "windows" with which to divide the tester period. A raw edge from the simulation file that falls within a particular snap window will be automatically moved to a specific time within the window.

On the command-line, the Snap Enable option is +s,

To set the Snap Enable option from the GUI, check the Snap Enable box

TIP: Although not specific to the VCD Converter, another useful option for converting VCD files is **Debug Mode**. This option enables a shortened conversion run, allowing you to examine the initial results for problems before proceeding with the full conversion.

To select the **Debug Mode** option, click on (to put a checkmark in) the **Debug Mode** checkbox in the GUI.

You can use any of the common options described in the section of this guide called "Design-to-Test Conversion Options," or in the *Velocity Program Generator User's Guide*.

3.2.3 Running VCDtoAVC

At a command line prompt in a terminal window, type "velocity -VCDtoAVC", followed by any required command-line options and input filenames, and press ENTER.

The following is an example of VCDtoAVC command-line syntax:

```
velocity -VCDtoAVC +o +e2 +s +t +p ConfigFile.cfg
Pattern1.evcd Pattern2.evcd
```

This example illustrates that you can name multiple VCD files for conversion, listed after the required Configuration file name.

3.2.4 Verifying Correct Cyclization, Using the Cycled VCD Output

For each input VCD file, the VCDtoAVC Converter generates an output VCD file at the end of conversion that contains any modifications to events that may have been caused by the cyclization process. For example, if Snap-to Timing is enabled and an edge from the original VCD file is moved by the Snap process, the output VCD file will contain the moved edge.

Design Engineers can feed this "cycled" VCD file back into the simulation to verify that the modified waveforms will still work for the device.

Each output VCD file will be named with the same base filename as the input VCD file, but with "_Cycled" appended. For example, if the input VCD file is named myPattern.vcd, the output VCD file will be named myPattern_Cycled.vcd.

3.3 Running the VCT Converter

Information on how to configure and run VCDtoAVC.

BACKGROUND: This section focuses on Velocity settings specific to the VCTtoAVC Converter. For more information on the use of Velocity Design-to-Test Converters and their common settings, refer to the previous section of this guide called "Using Velocity Design-to-Test Tools."

3.3.1 Configuration File Settings

There are no Configuration file settings specific to the VCT Converter. You can use any of the common settings described in the section of this guide called "Creating a Configuration File," or in the *Velocity Program Generator User's Guide*. However, you should note that, since VCT is a cyclized format, the PERIOD and EDGES Control definitions are not required in the Configuration file.

3.3.2 VCTtoAVC Conversion Options

There are no conversion options specific to the VCTtoAVC Converter. You can use any of the common options described in the section of this guide called "Design-to-Test Conversion Options," or in the *Velocity Program Generator User's Guide*.

3.3.3 Running VCTtoAVC

At a command line prompt in a terminal window, type "velocity -VCTtoAVC", followed by any required command-line options and input filenames, and press ENTER.

The following is an example of VCTtoAVC command-line syntax:

```
velocity -VCTtoAVC +o +s +t +p ConfigFile.cfg
myTiming.rconfig myPattern.vct
```

Note, in this example, that immediately after the name of the configuration file is the name of the .rconfig file, followed by the name of the .vct file.

4.0 RUNNING VELOCITY CAE (GUI-BASED)

4.0 RUNNING VELOCITY CAE (GUI-BASED)

Information on how to configure and start up Velocity CAE Design-to-Test Tools, and how to set common execution option through the Graphical User Interface.

4.1 Using Configuration Files

All Velocity CAE tools require a configuration file, which is a human-readable ASCII text file that you create. This file will customize the Core Analyzer Engine for specific conversions. For more information on understanding, creating, and editing Velocity configuration files, refer to the section in this guide called "The Velocity CAE Configuration File," or for more details see the *Velocity CAE Configuration Guide.*

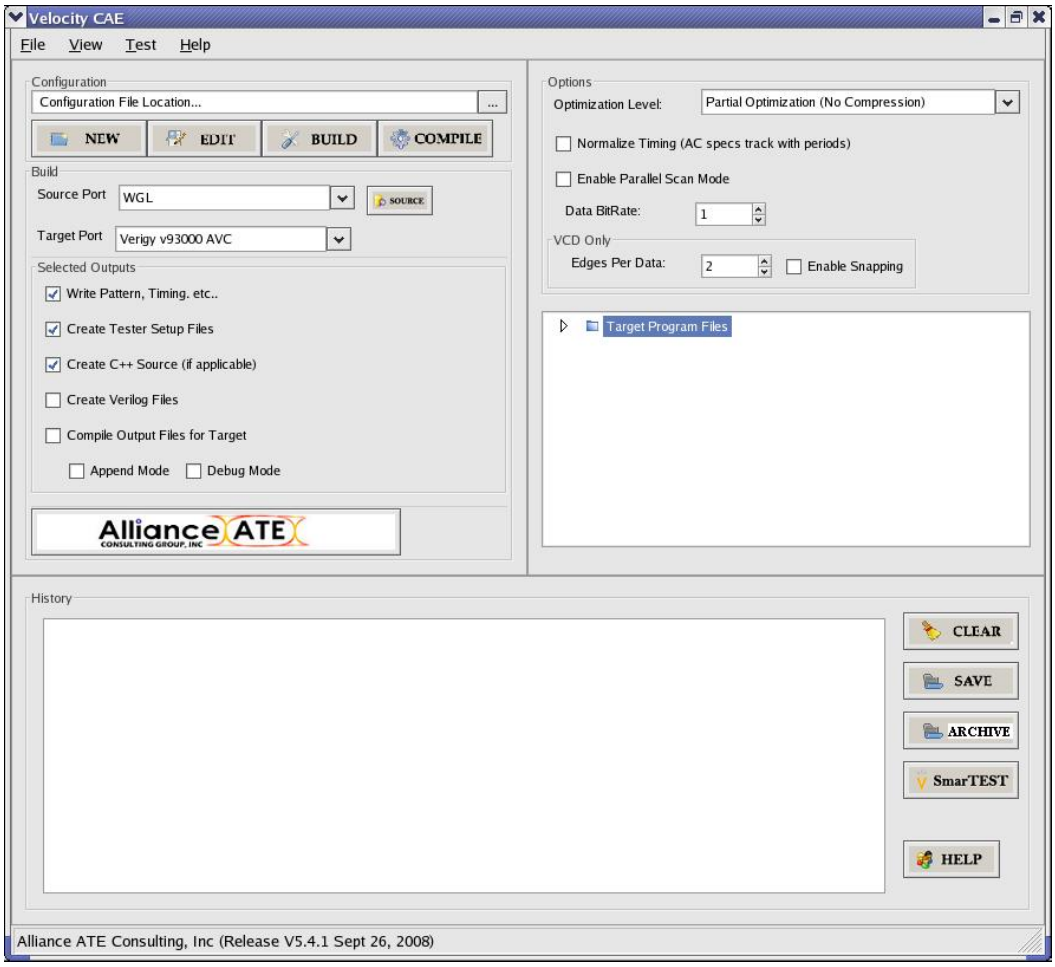
To start a new conversion without customization you can run Velocity CAE with a blank configuration file.

4.2 Running a Design-to-Test Converter

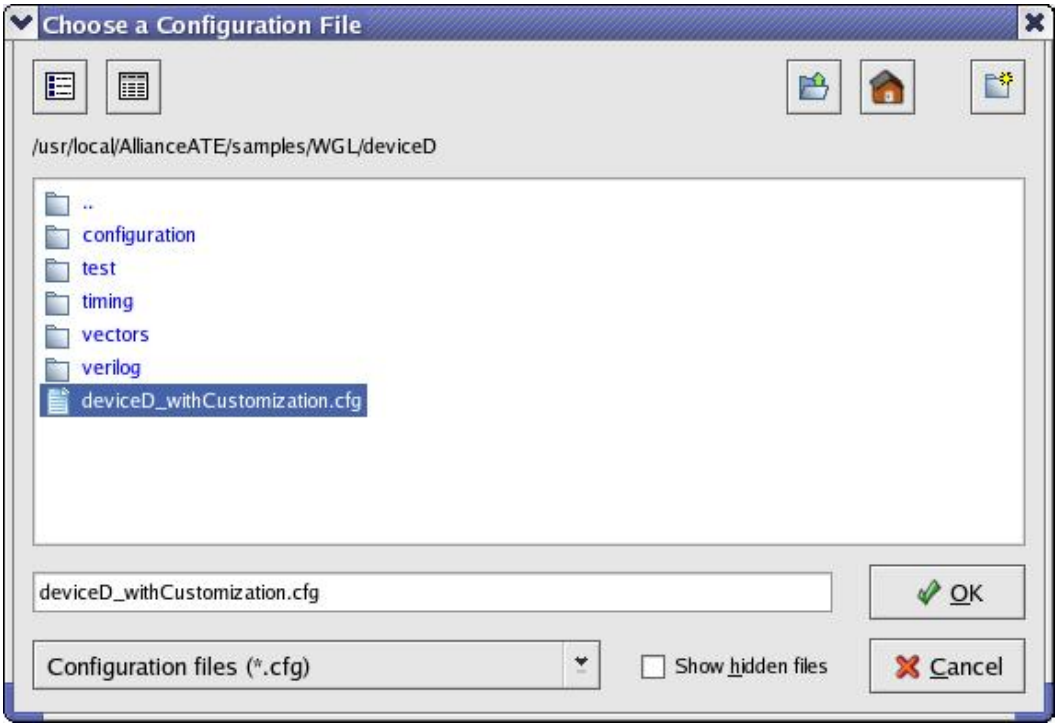
All of the Design-to-Test Converters can be accessed with options through the GUI. The following example demonstrates the selection of the WGLtoAVC Converter.

First you can start up *velocity* from the desktop (if a desktop icon was created) or from the /usr/local/AllianceATE/Velocity/bin folder. **MAKE SURE YOU HAVE A LICENSE FILE BEFORE STARTING THE PROGRAM.** Having an incorrect license file can cause errors to occur in your program.

If a correct license file was used you should see the following screen:

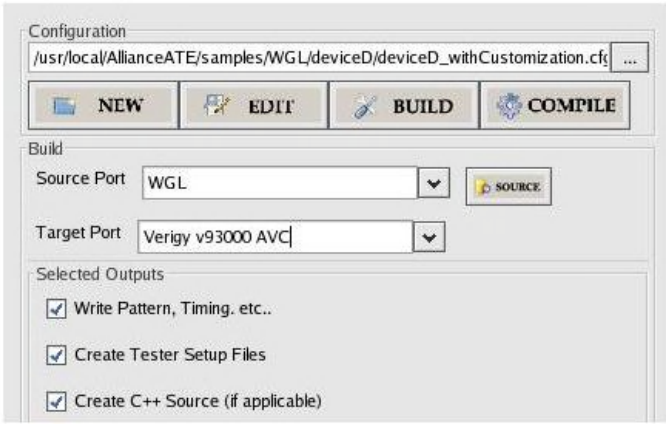

Velocity CAE
File
View
Test
Help
Configuration
Configuration File Location...
NEW
EDIT
BUILD
COMPILE
Build
Source Port
WGL
SOURCE
Target Port
Verigy v93000 AVC
Selected Outputs
Write Pattern, Timing, etc..
Create Tester Setup Files
Create C++ Source (if applicable)
Create Verilog Files
Compile Output Files for Target
Append Mode
Debug Mode
Alliance ATE
CONSULTING GROUP, INC
Options
Optimization Level:
Partial Optimization (No Compression)
Normalize Timing (AC specs track with periods)
Enable Parallel Scan Mode
Data BitRate:
1
VCD Only
Edges Per Data:
2
Enable Snapping
Target Program Files
History
CLEAR
SAVE
ARCHIVE
SmarTEST
HELP
Alliance ATE Consulting, Inc (Release V5.4.1 Sept 26, 2008)

To select the WGLtoAVC Converter, first load an appropriate configuration file by clicking on the button to the right of the Configuration file field. You will be see the following File selection window prompting you to enter a configuration file.

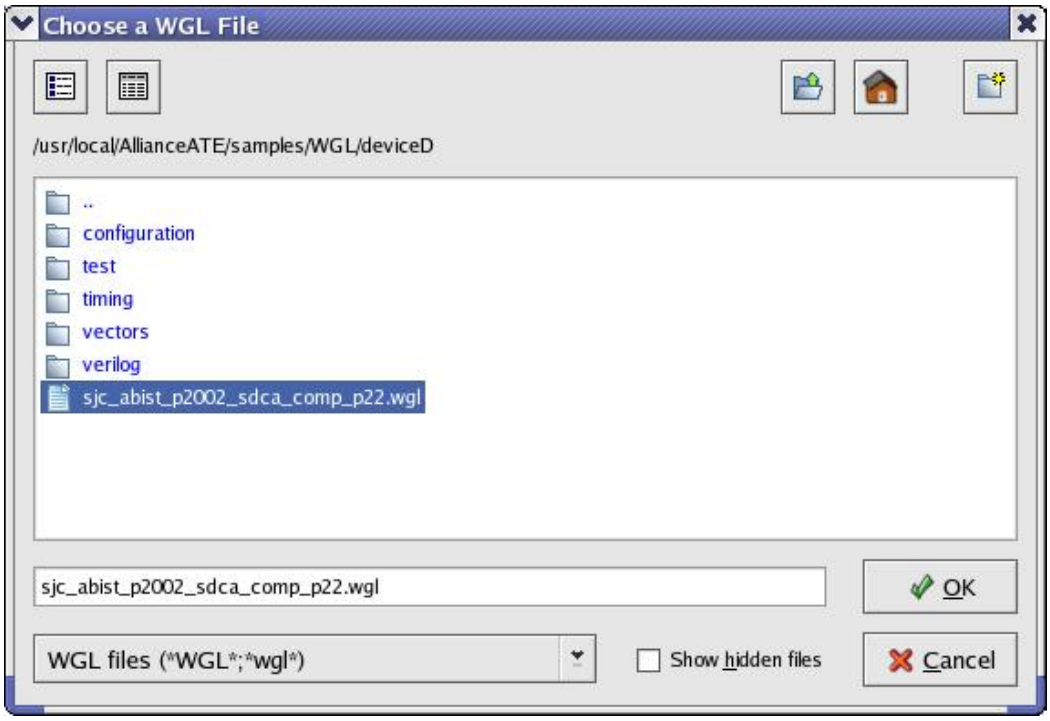


(For help on creating this file please see the section "***6.0 Configuration Files***" or refer to the ***Velocity CAE Configuration Guide***.)

Next, select "WGL" on your Source Port pull-down, and select "Verigy v93000 AVC" on the Target Port pull-down. After you have made all of the selections mentioned above, your GUI screen should look something like the following window segment:



In order to generate AVC and DVC files for the 93000's ASCII Interface from a simulation file, click on the Build button below the Configuration file field. You will then see the following File selection window prompting you to enter a simulation file of the specified Source type:



After clicking on OK, Velocity will begin the conversion process. You will see text appear in the History field at the bottom of the GUI, showing the status of the process. Also, the Target Program Files tree view at the right side of the GUI will be updated with a hierarchy of generated test program files.

Please see the next section in this guide, "Design-to-Test Conversion Options," for more information on the meaning and usage of the various options fields in the GUI.

4.3 Design-to-Test Conversion Options

All Velocity tools, including the Design-to-Test Converters, support a set of execution options. Most of the options are common to all of the tools. The following section lists all of the common options.

4.3.1 Optimization Level

Enables the performance of various optimizations on the conversion output.

This field allows the selection of one of three different levels of optimization:

- No Optimizations (All Loops Expanded): will expand any existing loops in the source patterns;
- Partial Optimization (No Compression): will retain existing loops from the source patterns, but will not create new loops;
- Full Optimization (Pattern Compression Enabled): will retain existing loops from the source patterns and will create new loops, if possible.

If Partial or Full Optimization is selected, Velocity will – at a minimum – remove any unreferenced timing and levels blocks that were defined in the source files. It will also enable the use of other optimization options that can be individually turned on or off. (Those additional optimization options are described later in this guide, especially the XMode and Snap Resolution options that are specific to conversions from VCD/EVCD format.)

4.3.2 Write Pattern, Timing, etc.

Enables the converter to create AVC and DVC files if the Verigy 93000 Target is chosen.

4.3.3 Create Tester Setup Files

Enables the export of the 93000 Pin Configuration file, the AIC file and the other files required to run the ASCII translator (AIT & AIV).

4.3.4 Compile Output Files for Target

Enables Velocity to automatically run the Verigy ASCII translator (AIT & AIV) immediately after a conversion to AVC.

4.3.5 Normalize Timing

Enables Velocity to automatically create a period-scaled timing set.

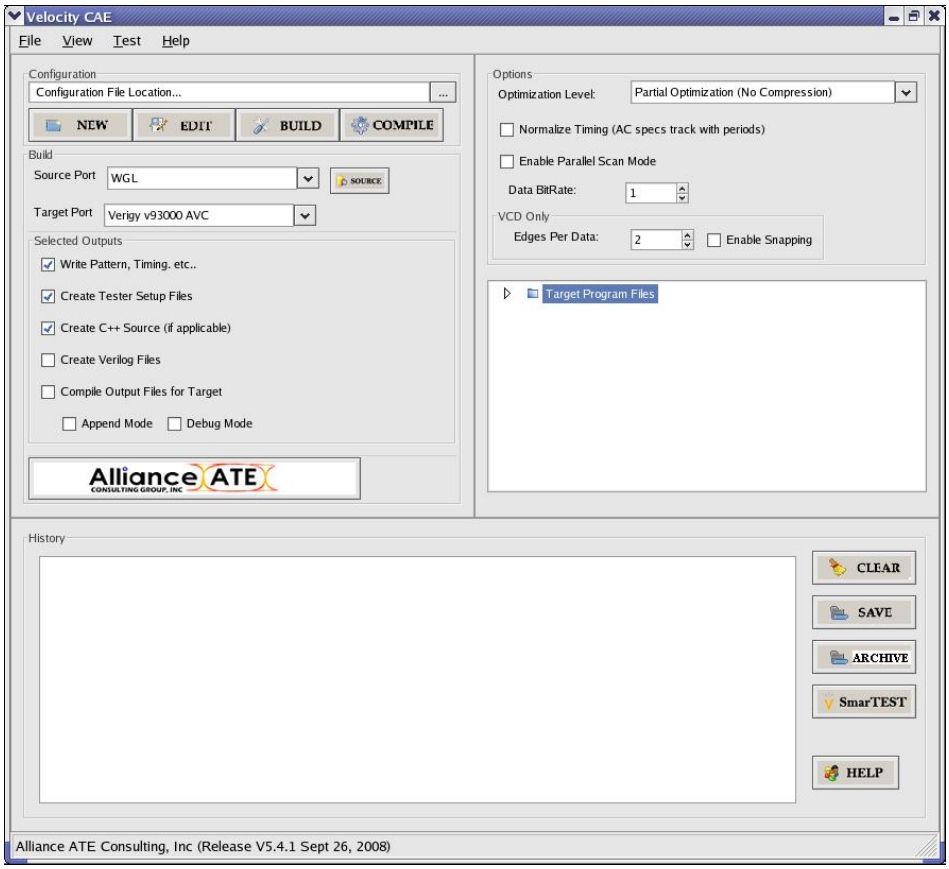
4.3.6 Append Mode

Enables the converter to add new patterns, timing, etc. to an existing program, instead of overwriting the existing data.

5.0 VELOCITY CAE CONVERTERS (GUI-BASED)

5.0 VELOCITY CAE CONVERTERS (GUI-BASED)

Velocity CAE comes with independent converters for each combination of source file type and target file type. In this way, multiple conversions can be run at once. These converters can be run from the GUI (Graphical User Interface).



5.1 Running the WGL Converter

Information on how to configure and run WGL conversion.

BACKGROUND: This section focuses on Velocity settings specific to the WGL Converter. For more information on the use of Velocity Design-to-Test Converters and their common settings, refer to the previous section of this guide called "Using Velocity Design-to-Test Tools."

5.1.1 Configuration File Settings

There are no Configuration file settings specific to the WGL Converter. You can use any of the common settings described in the section of this guide called "Creating a Configuration File," or in the *Velocity*

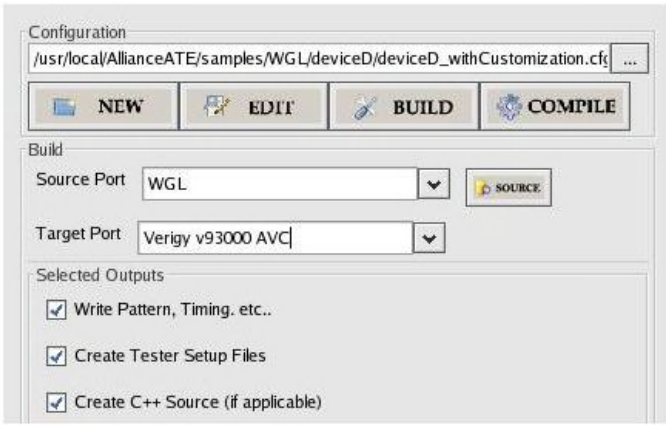
Program Generator User's Guide. However, you should note that, since WGL is a cyclized format, the PERIOD and EDGES Control definitions are not required in the Configuration file.

5.1.2 WGL Conversion Options

There are no conversion options specific to the WGLtoAVC Converter. You can use any of the common options described in the section of this guide called "Design-to-Test Conversion Options," or in the *Velocity Program Generator User's Guide*.

5.1.3 Running WGL Conversion

Select "WGL" under Source Port and choose "Verigy v93000 AVC" under Target Port. To start the conversion, click on the Build button.



5.2 Running the VCD/EVCD Converter

Information on how to configure and run VCD conversions.

BACKGROUND: This section focuses on Velocity settings specific to the VCD Converter. For more information on the use of Velocity Design-to-Test Converters and their common settings, refer to the previous section of this guide called "Using Velocity Design-to-Test Tools."

5.2.1 Configuration File Settings

There are two Configuration file settings specific to the VCD Converter: **PERIOD** and **EDGES**.

The **PERIOD** Control definition is used to specify a target tester period to be applied to the VCD pattern in order to "cyclize" the stream of events.

The **EDGES** definition specifies the maximum number of timing edges to expect within a period. By default, this value is 1. However, an RZ or R1 formatted signal would require 2 edges.

TIP: A reliable method for ensuring that your PERIOD and EDGES configuration file settings are appropriate for your VCD conversion source is to automatically generate an initial configuration file from the VCD source. Then, you can edit other settings in the configuration file as required.

Before auto-generating the configuration file, be sure to set the Snap Resolution option in the GUI to the desired value of EDGES.

For more information on how to automatically generate a configuration file, refer to the section in this guide called "Creating a Configuration File."

For more information on these Configuration settings, or any of the common settings, refer to the section of this guide called "Creating a Configuration File," or to the *Velocity Program Generator User's Guide*.

5.2.2 VCD Conversion Options

There are two conversion options specific to the VCD Converter: **Edges Per Data** and **Enable Snapping**.

The **Edges Per Data** option is used for specifying the number of databits to be used per tester period in the converted patterns and timing.

To set the **Edges Per Data** from the GUI, enter a number in the **Edges Per Data** field.

The **Enable Snapping** option is used for correcting "imperfect" simulation data in which an edge placement within the tester period might vary from period to period. The option enables the conversion to snap edges as specified by the EDGES control in the Configuration File. The EDGES control specifies a number of equal-size snap "windows" into which to divide the tester period. A raw edge from the simulation file that falls within a particular snap window will be automatically moved to a specific time within the window.

To set the **Enable Snapping** from the GUI, check the **Enable Snapping** box

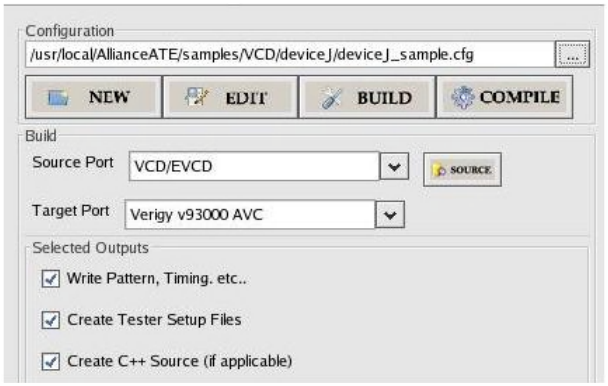
TIP: Although not specific to the VCD Converters, another useful option for converting VCD files is **Debug Mode**. This option enables a shortened conversion run, allowing you to examine the initial results for problems before proceeding with the full conversion.

To select the **Debug Mode** option, click on (to put a checkmark in) the **Debug Mode** checkbox in the GUI.

You can use any of the common options described in the section of this guide called "Design-to-Test Conversion Options," or in the *Velocity Program Generator User's Guide*.

5.2.3 Running VCD Conversion

Select "VCD/EVCD" under Source Port and select "Verigy v93000 AVC" under Target Port. To start the conversion, click on the Build button.



5.2.4 Verifying Correct Cyclization, Using the Cycled VCD Output

For each input VCD file, the VCDtoAVC Converter generates an output VCD file at the end of conversion that contains any modifications to events that may have been caused by the cyclization process. For example, if Enable Snapping is selected and an edge from the original VCD file is moved by the Snap process, the output VCD file will contain the moved edge.

Design Engineers can feed this "cycled" VCD file back into the simulation to verify that the modified waveforms will still work for the device.

Each output VCD file will be named with the same base filename as the input VCD file, but with "_Cycled" appended. For example, if the input VCD file is named myPattern.vcd, the output VCD file will be named myPattern_Cycled.vcd.

5.3 Running the VCT Converter

Information on how to configure and run VCT converters.

BACKGROUND: This section focuses on Velocity settings specific to the VCT Converter. For more information on the use of Velocity Design-to-Test Converters and their common settings, refer to the previous section of this guide called "Using Velocity Design-to-Test Tools."

5.3.1 Configuration File Settings

There are no Configuration file settings specific to the VCTtoAVC Converter. You can use any of the common settings described in the section of this guide called "Creating a Configuration File," or in the *Velocity Program Generator User's Guide*. However, you should note that, since VCT is a cyclized format, the PERIOD and EDGES Control definitions are not required in the Configuration file.

5.3.2 VCT Conversion Options

There are no conversion options specific to the VCTtoAVC Converter. You can use any of the common options described in the section of this guide called "Design-to-Test Conversion Options," or in the *Velocity Program Generator User's Guide*.

6.0 CONFIGURATION FILES

6.0 CONFIGURATION FILES

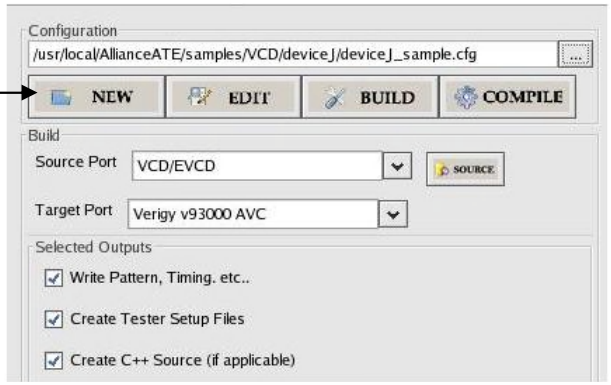
A brief tutorial on creating a basic Configuration File.

BACKGROUND: Velocity requires a Configuration file for every pattern conversion or program generation process. The Configuration file allows you to specify key aspects of the test program creation process in a simple format, including pin mapping, levels, timing, tests, and test flow.

TIP: It is recommended that, at a minimum, your Configuration file contain a program **PATH** name specification and a **PINLIST** definition.

6.1 (Optional) Automatically Generate an Initial Configuration File

To automatically generate an initial configuration file, press the New Icon under the configuration box. In the file selection window that pops up, enter the name of a simulation file from which you want to base the configuration, and press save. This will generate an initial Velocity CAE Configuration File.



Below is the first part of an example Configuration file automatically generated by Velocity from a VCD file.

```
##########################################################################
##
##   Device and Path Information
##########################################################################
##
PATH        .
DEVICE      deviceJ
PROGRAM     test
```

```
SOURCE_PORT VCD

##########################################################################
##
##   PinList Definition
##########################################################################
##
PINLIST
AVDD_RXMIX                              default  IO    AVDD_RXMIX
AVSS_RXMIX                              default  IO    AVSS_RXMIX
AVDD_FVGA                               default  IO    AVDD_FVGA
AVSS_FVGA                               default  IO    AVSS_FVGA
AVDD_LNA0                               default  IO    AVDD_LNA0
AVDD_LNA1                               default  IO    AVDD_LNA1
AVSS_LNA0                               default  IO    AVSS_LNA0
```

6.2 Understanding the Configuration File Structure

- A Configuration File is an ASCII text file with a simple, human-readable format.
- The main elements are:
  - **Control Definitions**, which define particular aspects of the conversion and program generation process; and,
  - **comments**, which begin with the '#' symbol and continue to the end of the line.
- Control Definitions can be categorized into one of two forms: **single-line** or **multi-line**.
  - A single-line definition begins with a **keyword**, includes one or more **parameters**, and continues to the end of the line or to the beginning of a comment. The following PERIOD definition is an example of a single-line Control definition:

```
PERIOD      5.000ns default
```

Note that the keyword is **PERIOD**, and that the two parameters are **5.000ns** (the value of the target period for cyclization) and **default** (the port name given to this particular target period).

  - A multi-line definition (also called a **block**) consists of a starting line, zero or more sub-parameter lines, and an ending line. The starting line begins with a keyword and includes zero or more parameters. The ending line consists of the keyword **END** and the starting line keyword. The following PINLIST block definition is an example of a multi-line Control definition:

```
PINLIST
ANALOG_VDD                              default  IO    ANALOG_VDD
CVDD                                    default  IO    CVDD
HOLDn                                   default  IO    HOLDn
END PINLIST
```

Note that the block begins with the keyword **PINLIST** and ends with **END PINLIST**. In between are lines that begin with a pin name and consist of several parameters that define properties of the pin.

The following sections describe how to create and edit the most common Control Definitions in a Configuration File.

6.3 Create (or Edit) the Program Path

- The first thing that you should define in the Configuration file is the location of the target test program files.
- Velocity divides the test program location into three parts:
  - base path – Points to the directory typically used as the parent directory of all 93000 test programs
  - Device name – Appended to the base path. Categorizes test programs by device.
  - Program name – Appended to the base path / Device name combination. Contains all the files that make up a specific 93000 test program.
- Define the base path using the keyword **PATH**, followed by a directory path specifier. For example,

```
PATH   /home/93Kprograms
```

- Define the device name using the keyword **DEVICE**, followed by a device name. For example,

```
DEVICE      coolChip
```

- Define the program name using the keyword **PROGRAM**, followed by the test program name. For example,

```
PROGRAM     finalTest
```

- In the above example, Velocity would create test program files for the 93000 Build in the directory /home/93Kprograms/coolChip/finalTest.

6.4 Create (or Edit) the Cyclization Timing

BACKGROUND: ATE test systems, such as the Verigy 93000, output functional stimulus to the device in the form of a vector sequence. The vectors are presented at a particular rate defined by the **cycle time**.

Many simulation and test data formats, such as WGL and STIL, also have a concept of vectors and cycle times, which can be translated directly to 93K format. However, the VCD (Verilog Change Dump) format is **non-cyclized**. That is, signal patterns are represented as a continuous stream of events, where an event is a change of state at a particular point in time relative to the beginning of the pattern.

For VCD, Velocity will analyze the spacing of timing events for each signal, and determine a best-fit tester cycle time and edge delays for your 93K test program. The cycle time is also known as the **period**.

- For VCD files, you can assist Velocity's cyclization process by specifying one or more target tester periods in the Configuration File. Velocity will attempt to divide the VCD event stream into the specified period (or periods), and determine the resulting drive, tri-state, and compare edge delays within the period.

- Define a target period using the keyword **PERIOD**, followed by a time value. Optionally, each Period definition can take a port name as a second parameter. This port name can be assigned to one or more Pins later in the Configuration file for multiport applications. (See the PINLIST definition, below.)

The following is an example of a PERIOD definition:

```
PERIOD              1608ps      domain622
```

Note that the time value parameter can include units immediately after the number (no whitespace in between). Units can include all the common scaling letters, such as n (for nano), u (for micro), m (for milli), etc. Also note that the port name "domain622" has been assigned.

- Use the EDGES Control definition if your VCD pattern contains waveforms that can be translated as RZ or R1 formats on the 93K. The EDGES definition specifies the maximum number of timing edges to expect within a period. An RZ or R1 formatted signal would require 2 edges.

Define the number of timing edges per Period using the keyword **EDGES**, followed by an integer value, as in the following example:

```
EDGES 2
```

6.5 Create (or Edit) the Pinlist

- The PINLIST block allows you to define, per pin, the 93K tester channel assigned and any alternate versions of that name used in the simulation or ATE conversion source.

The tester channel information that can be specified includes:

- Port Name: **default,** or a port name as specified in the PERIOD definition.
- Pin Type: **I**, **O**, **IO**, **CLK**, **REF**, **POW**, **A**, **NC**, or **R**. (See the *Velocity CAE User's Guide* for more information on these types.)
- Slot number
- Channel number

The alternate pin names are known as Aliases. You can specify as many Aliases on a pin line – separated by whitespace – as you need. Velocity uses Aliases to match simulation or ATE pin names that are different from the target 93K pin name.

- The following is an example of a PINLIST definition:

```
#########################################################################
##   PinList Definition
#########################################################################
PINLIST
ANALOG_VDD                  default        POW   0   2
HOLDn                       domain622   I         1   17   hold_n holdn
WPn                         domain622   O         1   8    wp_n
anapadext_data_n            default        ANA   0   3
END PINLIST
```

Note that pin ANALOG_VDD uses channel 2 of a DPS card in slot 0. It is defined as a pin type of POW, meaning a Power pin.

Also, note that pin HOLDn (the pin name to be used in the target 93K test program) has aliases of hold_n and holdn, meaning that it can take its data from simulation or ATE conversion sources that use either of those alias names.

Finally, note that pin anapadext_data_n is defined as pin type ANA, meaning an Analog pin.

- The GROUP Control definition allows you to assign a name to a group of pins, for easier reference elsewhere in the Configuration file.

- To define a Group, use the keyword **GROUP** followed by a Group name, followed by an equals sign (=) and a comma-separated list of pin names enclosed in double-quotes (“”). The following is an example of a Group definition:

```
GROUP DBUS = “D0, D1, D2, D3, D4, D5, D6, D7”
```

6.6 Create (or Edit) Custom Timing

BACKGROUND: Although Velocity will create appropriate Time Sets for your 93K program, based on the simulation or ATE files used as source for the conversion, you can create your own custom timing to apply to tests.

To define custom timing for a group of pins, create the following Control definition block.

- On the first line, use the keyword **TIMING** followed by a pin or group name. Optionally, you can use the word **default** for the pin specification to indicate all pins.

- On the next line, use the keyword **PERIOD** followed by a time value. This will be the period of the tester’s pattern sequencer.

BACKGROUND: All TIMING blocks in a particular Configuration file must use the same PERIOD value. This ensures that the 93K will be able to use the resulting .tim (Timing Setup) file.

TIP: In order to use TIMING blocks with different PERIOD values in your test program, use separate Configuration files for each of the different periods and run separate conversions with each.

– On subsequent lines, use the following keywords followed either by a time value or a percentage:

OFFSET – Time delay of first edge for a pin of type CLK
RISE – Time delay of second edge for a pin of type CLK, if a rising edge
FALL – Time delay of second edge for a pin of type CLK, if a falling edge
DUTY – Duty cycle for a pin of type CLK, expressed only as a percentage
DRIVE – Time delay of a drive edge for a pin of type I or IO
RECEIVE – Time delay of a compare edge for a pin of type O or IO

BACKGROUND: If you specify a timing parameter as a percentage, Velocity interprets it as a percentage of the PERIOD time. This provides a convenient way to scale edge delays with a sequencer period.

– For the last line, use the keywords **END TIMING**.

- The following is an example of a Timing definition:

```
TIMING default
   PERIOD 100ns
   OFFSET 0ns
   DUTY   50%
   DRIVE  25%
   RECEIVE 90%
   PULSE  5%
END TIMING
```

6.8 Customizing Patterns

BACKGROUND: If your Velocity package includes Optimization options, Velocity can automatically search for compression opportunities when converting patterns, and create appropriate repeats and loops in your 93K patterns.

However, even without Optimization, you can manually customize your pattern files using Configuration file control. With this capability, you can specify explicitly not only repeats and loops, but also selective output masking (pin-by-pin and cycle-by-cycle) and arbinrary data manipulations such as bit inversions.

The following section describes how you can define some of the most common pattern customizations in the Configuration file. However, there is much more that you can do with this control structure. For more information, refer to the *Velocity CAE Program Generator User's Guide*.

- To modify an existing pattern, use a Pattern block with the following structure:
  - On the first line, use the keyword **PATTERN** followed by a new pattern name.
  - For the last line of the block, use the keywords **END PATTERN**.
- Inside a Pattern block, you can define output masking and vector loops or repeats.
- To define an output mask for a particular pin or group of pins and for a particular vector range, use the keyword **PINS** followed by a comma-separated (no spaces) pin/group list followed by a vector range, as in the following example:

```
PATTERN func_pat_masked
   PINS Q0,Q1,Q2,Q3 55-83;
   PINS D0 MAP[01:Z] COND[RESET 0];
END PATTERN
```

Note the use of the hyphen for the vector range specifier, indicating that the masking occurs from vector 55 through vector 83. In this example, the output pins to be masked are Q1, Q2, Q3, and Q4. In addition, D0 wil have its drive data mapped to Z while RESET is asserted.

- When defining any loops or repeats in a Pattern block, you must also include an indication of the base pattern to which the loops or repeats apply. To indicate the name of the base pattern, use the keyword **BASE** followed by the base pattern name.
- To define a loop on an existing pattern sequence, use the keyword **LOOP** followed by comma-separated start and stop vector numbers followed by an optional loop count. The following example adds a loop to the previous pin masking example:

```
PATTERN func_pat_compressed
   PINS Q0,Q1,Q2,Q3 55-83
   BASE func_pat
   LOOP 100,131 1000
END PATTERN
```

Note that the start vector number of the loop is 100, the stop vector is 131, and that the vector block is looped on 1000 times.