



# Velocity CAE<sup>®</sup> Program Generator

---

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

Release 53

Quick Start Guide

---

---

# Velocity CAE Program Generator

## Simulation-to-Test Tools Quick Start Guide

---

### **COPYRIGHT NOTICE**

Copyright © 2008 Alliance ATE Consulting Group, Inc.



All rights reserved

Documentation version 2.0

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, D10Shell, and ShellConstructor are trademarks of Alliance ATE Consulting Group.

Diamond, D10, and ITE are trademarks of Credence Systems Corporation.

SmarTest, 93000 and 93K are trademarks of Verigy

---

# QUICK START GUIDE

## TABLE OF CONTENTS

	<u>Page #</u>
<i>Copyright Notice</i> .....	<i>ii</i>
<i>Trademarks</i> .....	<i>ii</i>
<b>1.0 GENERAL INFORMATION</b> .....	<b>1-1</b>
<b>1.1 What is Velocity CAE?</b> .....	<b>1-1</b>
<b>1.2 Running Velocity CAE?</b> .....	<b>1-1</b>
<b>1.2 What are the tools and components that make up Velocity CAE?</b> .....	<b>1-1</b>
<b>1.3 What Simulation-to-Test Tools are available?</b> .....	<b>1-2</b>
<b>1.4 What Tester-to-Tester Tools are available?</b> .....	<b>1-2</b>
<b>1.5 What kinds of files do Velocity CAE converters create?</b> .....	<b>1-4</b>
<b>2.0 RUNNING VELOCITY CAE (COMMAND LINE)</b> .....	<b>2-1</b>
<b>2.1 Using Configuration Files</b> .....	<b>2-1</b>
<b>2.2 Running a Simulation-to-Test Converter</b> .....	<b>2-1</b>
<b>2.3 Simulation-to-Test Conversion Options</b> .....	<b>2-3</b>
2.3.1 Optimize Output.....	2-3
2.3.2 Write Pattern, Timing, etc.....	2-3
2.3.3 Create Tester Setup Files .....	2-3
2.3.4 Create C++ Source.....	2-4
2.3.5 Compile.....	2-4
2.3.6 Normalize Timing.....	2-4
2.3.7 Append.....	2-4
<b>3.0 VELOCITY CAE CONVERTERS (COMMAND LINE)</b> .....	<b>3-1</b>
<b>3.1 Running the WGL Converter</b> .....	<b>3-1</b>
3.1.1 Configuration File Settings .....	3-1
3.1.2 WGL Conversion Options .....	3-1
3.1.3 Running WGL conversions.....	3-1
<b>3.2 Running the VCD Converter</b> .....	<b>3-2</b>
3.2.1 Configuration File Settings .....	3-2
3.2.2 VCD Conversion Options .....	3-2
3.2.3 Running VCD.....	3-3
3.2.4 Verifying Correct Cyclization, Using the Cycled VCD Output.....	3-4
<b>3.3 Running the VCT Converter</b> .....	<b>3-4</b>
3.3.1 Configuration File Settings .....	3-4
3.3.2 VCT Conversion Options .....	3-4
3.3.3 Running VCT.....	3-4
<b>4.0 RUNNING VELOCITY CAE (GUI-Based)</b> .....	<b>4-1</b>

---

<b>4.1</b>	<b>Using Configuration Files .....</b>	<b>4-1</b>
<b>4.2</b>	<b>Running a Simulation-to-Test Converter .....</b>	<b>4-2</b>
<b>4.3</b>	<b>Simulation-to-Test Conversion Options.....</b>	<b>4-4</b>
4.3.1	Optimize Output.....	4-4
4.3.2	Write Pattern, Timing, etc.....	4-4
4.3.3	Create Tester Setup Files .....	4-4
4.3.4	Create C++ Source.....	4-4
4.3.5	Compile.....	4-4
4.3.6	Normalize Timing.....	4-4
4.3.7	Append.....	4-5
<b>5.0</b>	<b><i>VELOCITY CAE CONVERTERS (GUI-BASED)</i> .....</b>	<b>5-1</b>
<b>5.1</b>	<b>Running the WGL Converter.....</b>	<b>5-1</b>
5.1.1	Configuration File Settings .....	5-1
5.1.2	WGL Conversion Options .....	5-2
5.1.3	Running WGL Conversions.....	5-2
<b>5.2</b>	<b>Running the VCD Converter.....</b>	<b>5-2</b>
5.2.1	Configuration File Settings .....	5-3
5.2.2	VCD Conversion Options .....	5-3
5.2.3	Running VCD Conversions.....	5-4
5.2.4	Verifying Correct Cyclization, Using the Cycled VCD Output.....	5-4
<b>5.3</b>	<b>Running the VCT Converter.....</b>	<b>5-5</b>
5.3.1	Configuration File Settings .....	5-5
5.3.2	VCT Conversion Options .....	5-5
5.3.3	Running VCT Conversions.....	5-6
<b>6.0</b>	<b><i>CONFIGURATION FILES</i> .....</b>	<b>6-2</b>
<b>6.1</b>	<b>Optional) Automatically Generate an Initial Configuration File.....</b>	<b>6-2</b>
<b>6.2</b>	<b>Understanding the Configuration File Structure.....</b>	<b>6-3</b>
<b>6.3</b>	<b>Create (or Edit) the Program Path.....</b>	<b>6-4</b>
<b>6.4</b>	<b>Create (or Edit) the Cyclization Timing .....</b>	<b>6-5</b>
<b>6.5</b>	<b>Create (or Edit) the Pinlist .....</b>	<b>6-6</b>
<b>6.6</b>	<b>Create (or Edit) the DC Levels.....</b>	<b>6-7</b>
<b>6.7</b>	<b>Create (or Edit) Custom Timing.....</b>	<b>6-8</b>
<b>6.8</b>	<b>Customizing Patterns .....</b>	<b>6-9</b>

## **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 Simulation-to-Test Tools.*

### 1.1 What is Velocity CAE?

Velocity CAE (Computer Aided Engineering) is a suite of software tools for automatically generating Credence D10 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 Simulation-to-Test Tools.

### 1.2 Running Velocity CAE?

Velocity CAE can be run from both the command line and/or a Windows/Linux based GUI interface. The GUI interface 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 and batch run their translations.

For 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 consists of the following major components:

Component	Description
ShellConstructor™	Automatically generates all initial source and setup files for a “skeleton” D10 program
GUI	The graphical interface to all Velocity CAE functionality
Core Engine Analyzer	Central processing “hub” of Velocity CAE. Required for all Simulation-to-Test and Tester-to-Tester conversion tools.
Simulation-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 Simulation-to-Test Tools are available?

Velocity CAE offers the following Simulation-to-Test Tools:

Tool	Description
WGLtoD10	Converts patterns and timing from WGL (Waveform Generation Language) files to D10 STIL pattern and timing files
VCDtoD10	Converts patterns and timing from VCD (Verilog Change Dump) files to D10 STIL pattern and timing files. Automatically cyclizes stream of events in VCD into D10 vectors with timing sets.
VCTtoD10	Converts cycled patterns from Verilog .vct output files and timing from RCONFIG files to D10 STIL pattern and timing files.
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 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

### 1.4 What Tester-to-Tester Tools are available?

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

<b>Tool</b>	<b>Description</b>
J750toD10	Converts patterns and timing from Teradyne J750 files to D10 STIL pattern files along with all test, flow, and programming information
FLEXtoD10	Converts patterns and timing from Teradyne UltraFlex files to D10 STIL pattern files along with all test, flow, and programming information
J973toD10	Converts patterns and timing from Teradyne J973 files to D10 STIL pattern and timing files
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
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
AVCtoD10	Converts patterns and timing from Verigy AVC files to D10 STIL pattern and timing files
STILtoD10	Converts patterns from non-Diamond compliant STIL to proper diamond compliant STIL
STILtoAVC	Converts any STIL based pattern representation into Verigy's AVC and DVC format along with files needed for Verigy's ASCII translator



## 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, C++ source, and pattern files required to build and run a fully-functional test program. These generated files include:

### For the Verigy 93000:

#### Setup Files

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

#### Pattern Files

- ⇒ **.avc**
- ⇒ **.dvc**

### For the Credence Diamond:

#### Setup Files

- ⇒ **.job**
- ⇒ **.make**
- ⇒ **.res**
- ⇒ **.sig**
- ⇒ **.bindef**
- ⇒ **.binmap**

#### C++ Source Files

- ⇒ **.cpp**
- ⇒ **.h**

#### Pattern Files

- ⇒ **Levels.stil**
- ⇒ **Signals.stil**
- ⇒ **Specs.stil**
- ⇒ **Timing.stil**
- ⇒ **Macros.stil**
- ⇒ **Pattern.stil** (Contains `patternExecs`, `patternBursts`, and overall usage)
- ⇒ **Vectors.stil** (defines the vector data)

## **2.0 RUNNING VELOCITY CAE (COMMAND LINE)**

## 2.0 RUNNING VELOCITY CAE (COMMAND LINE)

*Information on how to configure and start up Velocity CAE Simulation-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 Running a Simulation-to-Test Converter

At a command line prompt in a terminal window, type the name of the Converter, followed by any required command-line options and input filenames, and press ENTER.

The following example shows typical command-line syntax for executing the WGLtoD10 Converter:

```
C:\AllianceATE\bin>velocity.exe -WGLto10 +o +s +t +p  
ConfigFile.cfg myPatterns.wgl
```

The following example shows typical command-line syntax for executing the WGLtoAVC Converter:

```
C:\AllianceATE\bin>velocity.exe -WGLtoAVC +o +s +t +p  
ConfigFile.cfg myPatterns.wgl
```

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 “Simulation-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.

## **2.3 Simulation-to-Test Conversion Options**

All Velocity tools, including the Simulation-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.

### **2.3.1 Optimize Output**

Enables the performance of various optimizations on the conversion output.

If this option is turned on, 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.)

**Command-line:**

+o    Turn optimization on  
-o    Turn optimization off

### **2.3.2 Write Pattern, Timing, etc.**

Enables the converter to create new STIL files for the D10 program.

**Command-line:**

+s    Turn STIL file creation on  
-s    Turn STIL file creation off

### **2.3.3 Create Tester Setup Files**

Enables the converter to create new test setup files for the D10 program, such as JOB, SIG, MAKE, etc.

**Command-line:**

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

### **2.3.4 Create C++ Source**

Enables the converter to create new test program source files for the D10 program, namely .cpp and .h files.

**Command-line:**

- +p Turn test program source file creation on
- p Turn test program source file creation off

### **2.3.5 Compile**

Enables Velocity to automatically bring up the Credence ITE at the end of conversion, and run a build on the source files.

**Command-line:**

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

### **2.3.6 Normalize Timing**

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

**Command-line:**

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

### **2.3.7 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

## **3.0 VELOCITY CAE CONVERTERS (COMMAND LINE)**

## 3.0 VELOCITY CAE CONVERTERS (COMMAND LINE)

Velocity CAE comes with independent converters for each translation, this way multiple processes can be ran at once. These converters are ran through the Windows Command Line.

### 3.1 Running the WGL Converters

*Information on how to configure and run WGLtoD10 and WGLtoAVC.*



**BACKGROUND:** This section focuses on Velocity settings specific to the WGLtoD10 and WGLtoAVC Converters. For more information on the use of Velocity Simulation-to-Test Converters and their common settings, refer to the previous section of this guide called “Using Velocity Simulation-to-Test Tools.”

#### 3.1.1 Configuration File Settings

There are no Configuration file settings specific to the WGL Converters. 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 WGLtoD10 and WGLtoAVC Conversion Options

There are no conversion options specific to the WGL Converters. You can use any of the common options described in the section of this guide called “Simulation-to-Test Conversion Options,” or in the *Velocity Program Generator User’s Guide*.

#### 3.1.3 Running WGLtoD10 and WGLtoAVC

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

The following are examples of WGLtoD10 and WGLtoAVC command-line syntax:

```
C:\AllianceATE\bin>velocity -WGLtoD10 +o +s +t +p
ConfigFile.cfg myPatterns.wgl myPatterns2.wgl
```

```
C:\AllianceATE\bin>velocity -WGLtoAVC +o +s +t +p
ConfigFile.cfg myPatterns.wgl myPatterns2.wgl
```

These examples illustrate that you can name multiple WGL files for conversion, listed after the required Configuration file name.



## 3.2 Running the VCD/EVCD Converters

*Information on how to configure and run VCDtoD10 and VCDtoAVC.*



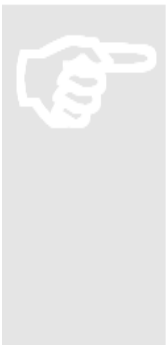
**BACKGROUND:** This section focuses on Velocity settings specific to the VCDtoD10 and VCDtoAVC Converters. For more information on the use of Velocity Simulation-to-Test Converters and their common settings, refer to the previous section of this guide called “Using Velocity Simulation-to-Test Tools.”

### 3.2.1 Configuration File Settings

There are two Configuration file settings specific to the VCD Converters: **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*.

### 3.2.2 VCDtoD10 and VCDtoAVC Conversion Options

There are two conversion options specific to the VCD Converters: **Edge Count** and **Snap Enable**.

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

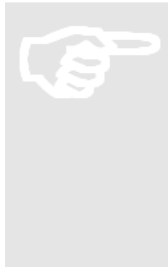
On the command-line, the edge count option is +eN, where N is the number of databits per period.

To set the XMode 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 Converters, another useful option for converting VCD files is **Debug**. 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** option, click on (to put a checkmark in) the **Debug** checkbox in the GUI.

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

### 3.2.3 Running VCDtoD10 and VCDtoAVC

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

The following is an example of VCDtoD10 and VCDtoAVC command-line syntax:

```
C:\AllianceATE\bin>velocity -VCDtoD10 +o +e2 +s +t +p  
ConfigFile.cfg Pattern1.evcd Pattern2.evcd
```

```
C:\AllianceATE\bin>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 VCDtoD10 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 Converters**

*Information on how to configure and run VCTtoD10 and VCDtoAVC.*



**BACKGROUND:** This section focuses on Velocity settings specific to the VCTtoD10 and VCDtoAVC Converter. For more information on the use of Velocity Simulation-to-Test Converters and their common settings, refer to the previous section of this guide called “Using Velocity Simulation-to-Test Tools.”

### **3.3.1 Configuration File Settings**

There are no Configuration file settings specific to the VCT Converters. 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 VCTtoD10 and VCDtoAVC Conversion Options**

There are no conversion options specific to the VCTtoD10 Converter. You can use any of the common options described in the section of this guide called “Simulation-to-Test Conversion Options,” or in the *Velocity Program Generator User’s Guide*.

### **3.3.3 Running VCTtoD10 and VCDtoAVC**

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

The following is an example of VCTtoD10 and VCTtoAVC command-line syntax:

```
C:\AllianceATE\bin>velocity -VCTtoD10 +o +s +t +p  
ConfigFile.cfg myTiming.rconfig myPattern.vct
```

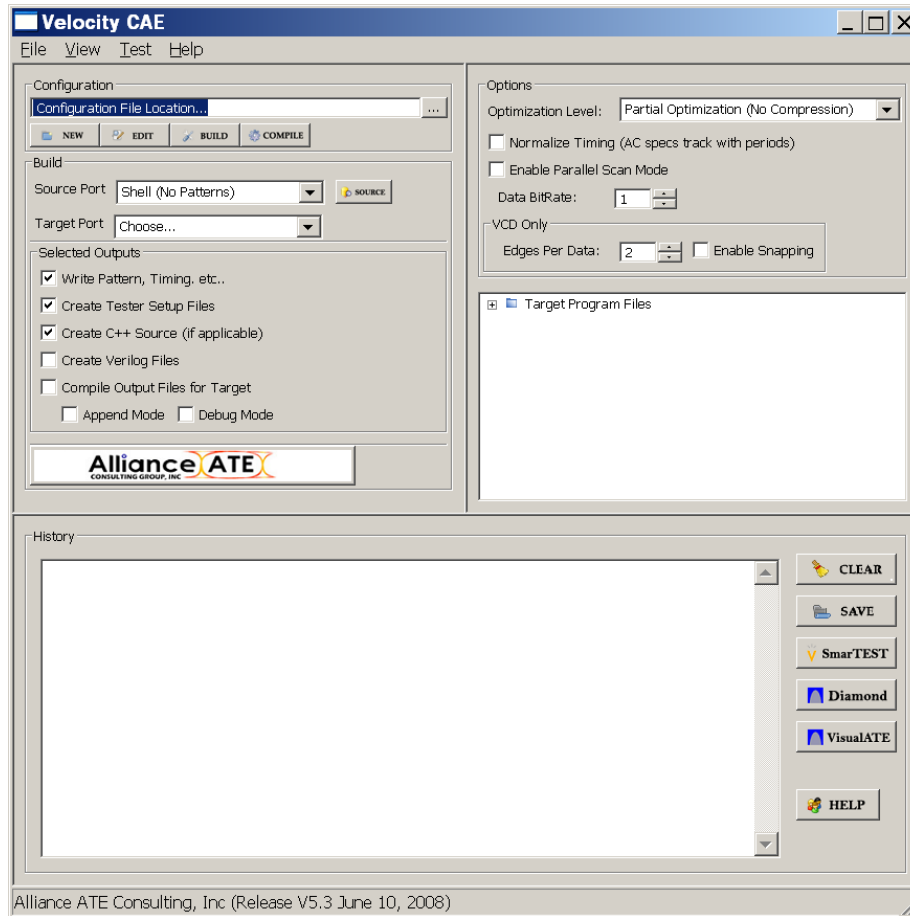
```
C:\AllianceATE\bin>velocity -VCTtoAVC +o +s +t +p  
ConfigFile.cfg myTiming.rconfig myPattern.vct
```

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 Simulation-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*.

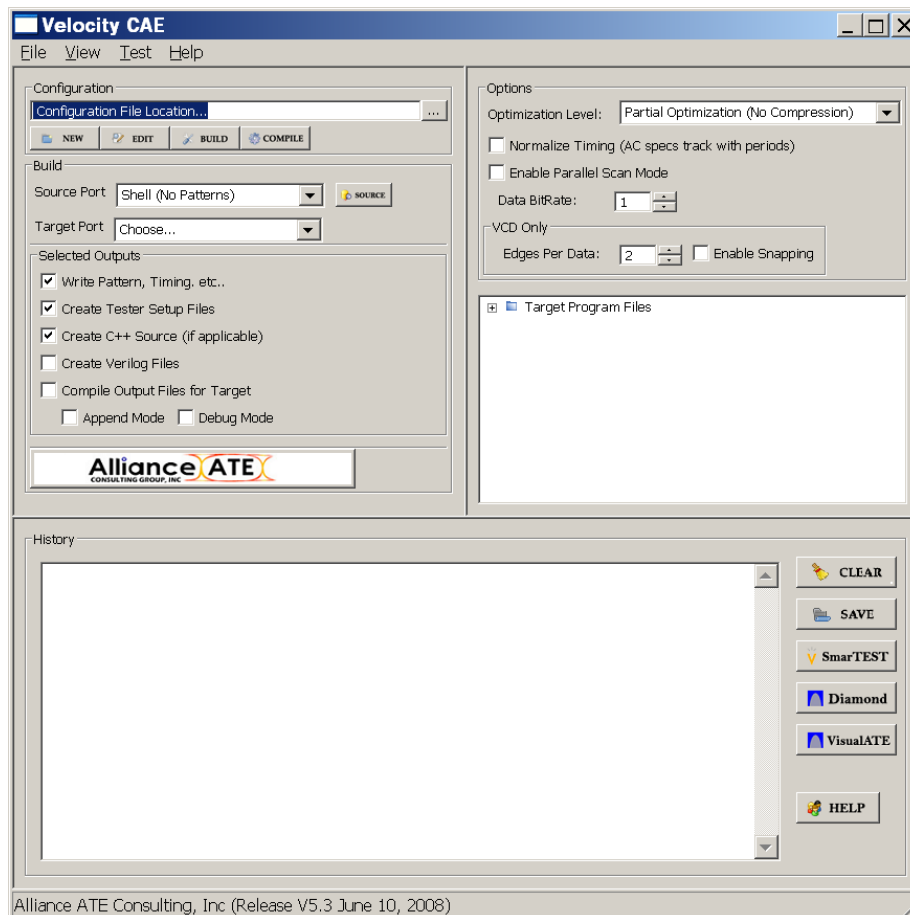
To start a new conversion without customization you can run Velocity CAE with a blank configuration file.

## 4.2 Running a Simulation-to-Test Converter

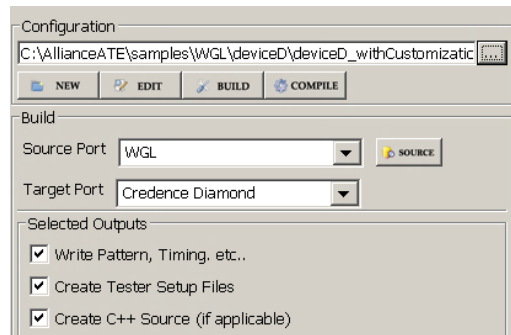
All of the Simulation-to-Test Converters can be accessed with options through the GUI. The following example demonstrates the selection of the WGLtoD10 Converter.

First you can start up *velocity.exe* from the desktop (if a desktop icon was chosen or from the C:\AllianceATE\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.

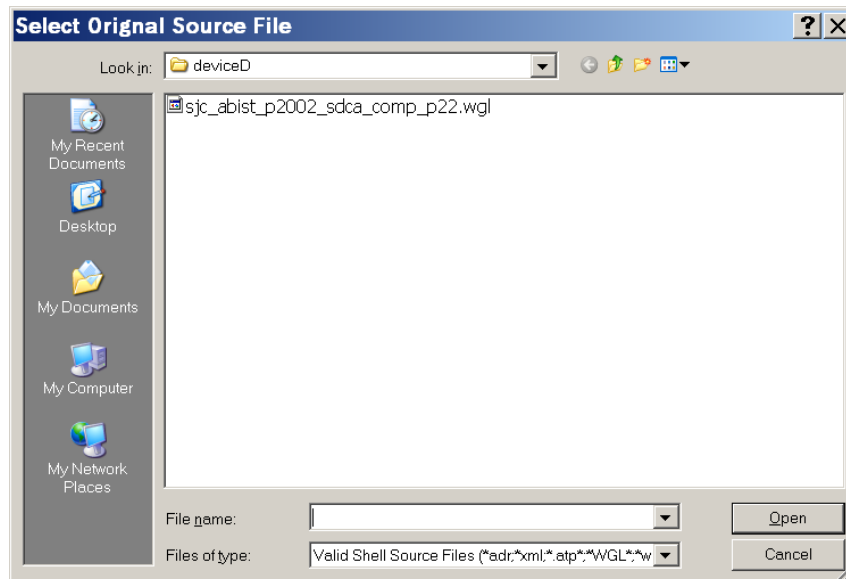
If a correct license file was used you should see the following screen:



To select the WGLtoD10 Converter, first load an appropriate configuration file. For help on creating this file please see the section “**6.0 Configuration Files**” or refer to the *Velocity CAE Configuration Guide*. Select WGL on your Source Port Drag-down. Select Credence Diamond on the Target Port.



Now you can click on the Source Icon to load your source file (WGL) and press Build. **\*Sometimes the WGL file gets stuck too long you must close it to return to the GUI**



Please see the next section in this guide, “Simulation-to-Test Conversion Options,” for more information on the meaning and usage of the various



## **4.3 Simulation-to-Test Conversion Options**

All Velocity tools, including the Simulation-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 Optimize Output**

Enables the performance of various optimizations on the conversion output.

If this option is turned on, 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 new STIL files for the D10 program if Credence Diamond Target is chosen. 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 converter to create new test setup files for the D10 program, such as JOB, SIG, MAKE, etc. For the 93000, this will enable the export of the 93K Pin Configuration file, the AIC file and the other files required to run the ASCII translator (AIT & AIV).

### **4.3.4 Create C++ Source**

Enables the converter to create new test program source files for the D10 program, namely .cpp and .h files. There are no source files created for the 93000 targets to this option would be ignored for all AVC exports

### **4.3.5 Compile**

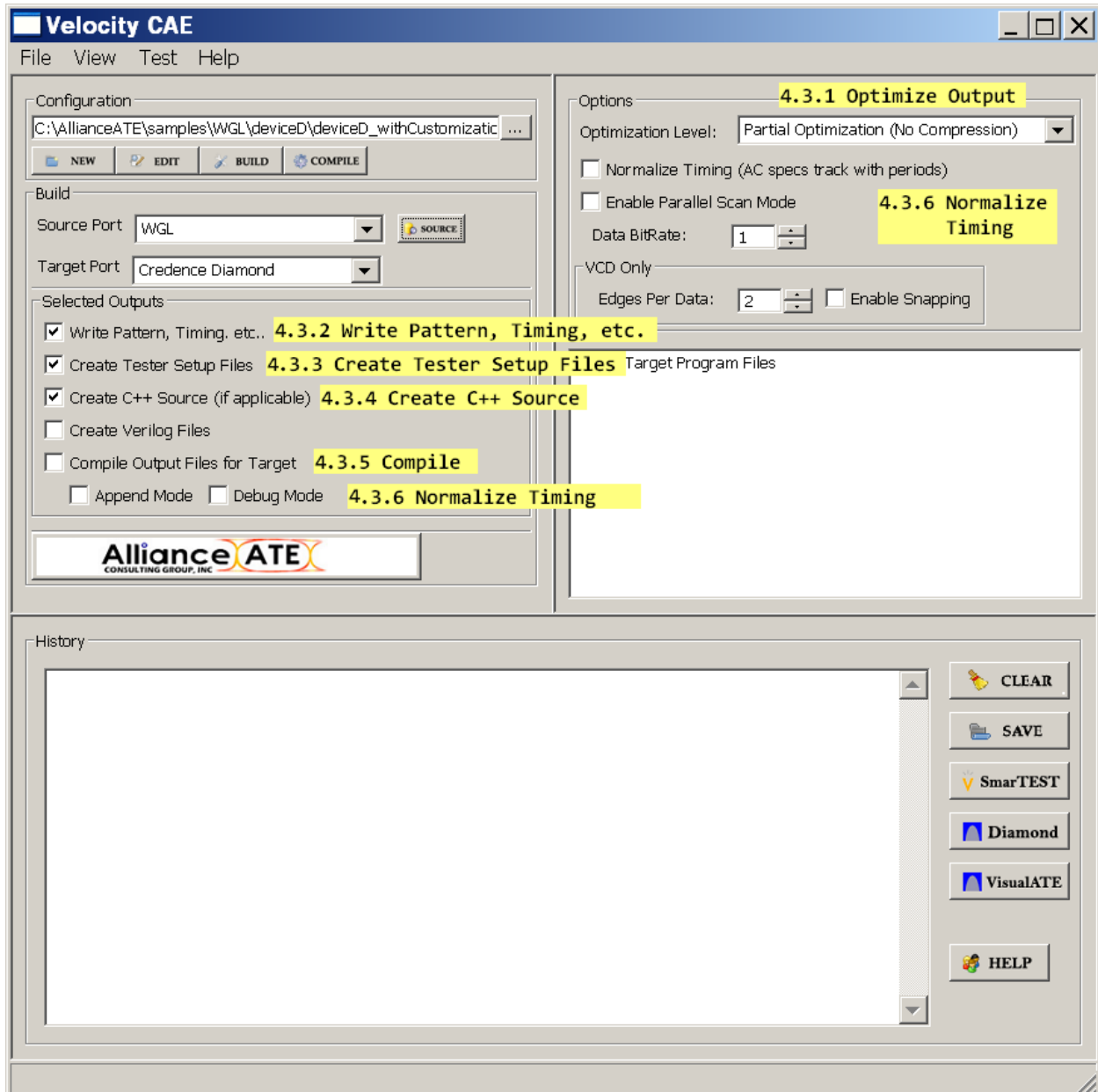
Enables Velocity to automatically run a build on the source files and STIL files if the Credence Diamond Target is chosen. Enables Velocity to automatically run the Verigy ASCII translator (AIT & AIV) if the Verigy 93000 Target is chosen.

### **4.3.6 Normalize Timing**

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

### 4.3.7 Append

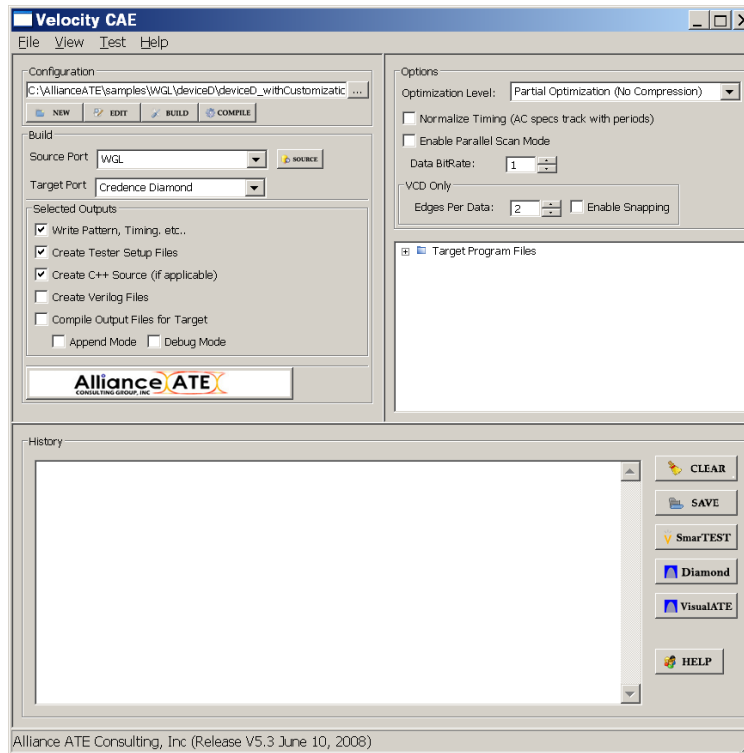
Enables the converter to add new patterns, timing, etc. to an existing program, instead of overwriting the existing data.



## **5.0 VELOCITY CAE CONVERTERS (GUI-BASED)**

## 5.0 VELOCITY CAE CONVERTERS (GUI-BASED)

Velocity CAE comes with independent converters for each translation, this way multiple processes can be ran at once. These converters are run through the Windows GUI Based Application.



### 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 Converters. For more information on the use of Velocity Simulation-to-Test Converters and their common settings, refer to the previous section of this guide called “Using Velocity Simulation-to-Test Tools.”

#### 5.1.1 Configuration File Settings

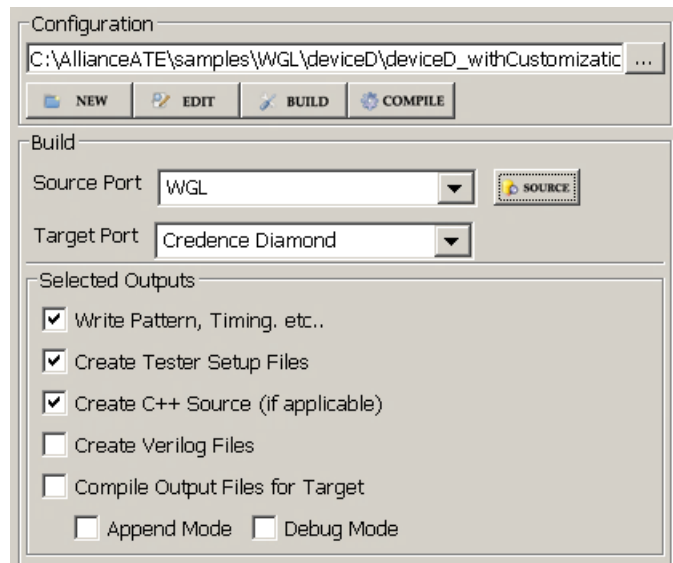
There are no Configuration file settings specific to the WGL Converters. You can use any of the common settings described in the section of this guide called “Creating a Configuration File,” or in the *Velocity Program Generator User’s Guide*. However, you should note that, since WGL is a cyclized format, 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 WGLtoD10 Converter. You can use any of the common options described in the section of this guide called “Simulation-to-Test Conversion Options,” or in the *Velocity Program Generator User’s Guide*.

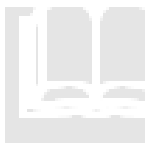
### 5.1.3 Running WGL Conversion

Select WGL under Source Port and choose a Target Port. For this example the Credence Diamond is chosen. To Load the appropriate Configuration and Source Files, Press Build.



## 5.2 Running the VCD/EVCD Converter

*Information on how to configure and run VCD conversions.*



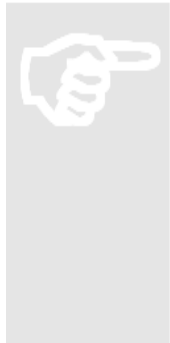
**BACKGROUND:** This section focuses on Velocity settings specific to the VCD Converters. For more information on the use of Velocity Simulation-to-Test Converters and their common settings, refer to the previous section of this guide called “Using Velocity Simulation-to-Test Tools.”

## 5.2.1 Configuration File Settings

There are two Configuration file settings specific to the VCD Converters: **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: **Edge Count** and **Snap Enable**.

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

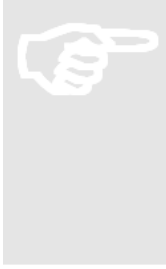
On the command-line, the Edge Count option is +eN, where N is the number of databits per period.

To set the Snap Enable from the GUI, check the snap enable box

The **Snap Resolution** 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 Edge Count option is +eN, where N is the number of edges to use per tester period.

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



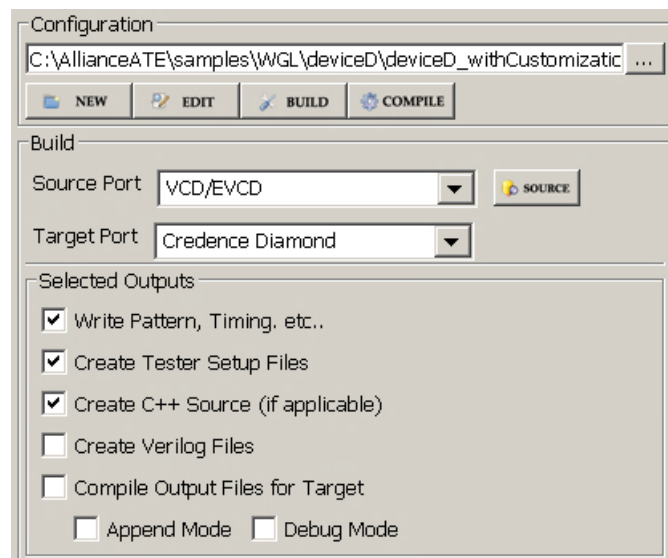
**TIP:** Although not specific to the VCD Converters, another useful option for converting VCD files is **Debug**. This option enables a shortened conversion run, allowing you to examine the initial results for problems before proceeding with the full conversion.

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

You can use any of the common options described in the section of this guide called “Simulation-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 choose Target Port. For the example below the Credence Diamond Target is chosen. To Load the appropriate Configuration and Source Files, Press Build.



### 5.2.4 Verifying Correct Cyclization, Using the Cycled VCD Output

For each input VCD file, the VCDtoD10 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.

## 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 Converters. For more information on the use of Velocity Simulation-to-Test Converters and their common settings, refer to the previous section of this guide called “Using Velocity Simulation-to-Test Tools.”

### 5.3.1 Configuration File Settings

There are no Configuration file settings specific to the VCTtoD10 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.

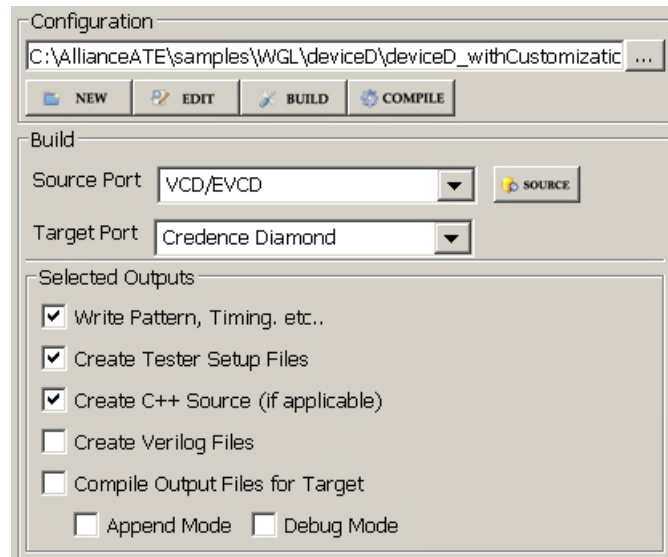
### 5.3.2 VCT Conversion Options

There are no conversion options specific to the VCTtoD10 Converter. You can use any of the common options described in the section of this guide called “Simulation-to-Test Conversion Options,” or in the *Velocity Program Generator User’s Guide*.



### 5.3.3 Running VCT Converter

Select VCD/EVCD under Source Port and Credence Diamond under Target Port. Load the appropriate Configuration and Source Files. Press Build.



## **6.0 CONFIGURATION FILES**

## 6.0 CONFIGURATION FILES

*A brief tutorial on creating a basic Configuration File.*



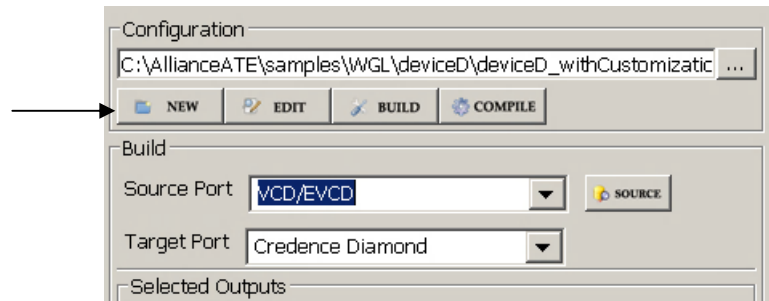
**BACKGROUND:** Velocity requires a Configuration file for every pattern conversion or program generation process. The Configuration file allows you to specify key aspects of the test program creation process in a simple format, including pin mapping, levels, timing, tests, and test flow.



**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. Enter a desired filename into the file selection window and press save. This will generate a initial Velocity CAE Configuration File.



Below is the first part of an example Configuration file automatically generated by Velocity from the Sample1.vcd file listed in the previous File Selection window.

```
#####
## Device and Path Information
#####
PATH      /Documents and Settings/Phil/My Documents/Alliance
ATE/Velocit/Velocity_V3.2.1_Windows/Samples
DEVICE    VCD
PROGRAM   test

PERIOD    5.000ns default
EDGES 1

#####
## Group Definitions
#####
```

```
#####
##  PinList Definition
#####
PINLIST
ANALOG_VDD      default IO    ANALOG_VDD
CVDD             default IO    CVDD
HOLDn           default IO    HOLDn
NOVEA_RVDD      default IO    NOVEA_RVDD
RVDD            default IO    RVDD
WPn             default IO    WPn
anapadext_data_n default IO    anapadext_data_n
```

## 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 name given to this particular target period, or *Clock Domain*).

- 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 D10 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 D10 test program.
- Define the base path using the keyword **PATH**, followed by a directory path specifier. For example,

```
PATH /home/D10programs
```

- 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 D10 Build in the directory /home/D10programs/coolChip/finalTest.

## 6.4 Create (or Edit) the Cyclization Timing



**BACKGROUND:** ATE test systems, such as the Credence D10, 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 D10 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 D10 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. The Period definition also specifies what is known in Velocity as a **Clock Domain**. Optionally, each Period / Clock Domain definition can take a name as a second parameter. This name can be used elsewhere in the Configuration File to reference the Clock Domain.

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 name "domain622" has been assigned to the Clock Domain.

- Use the EDGES Control definition if your VCD pattern contains waveforms that can be translated as RZ or R1 formats on the D10. 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 D10 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:

- Resource Type: **DPIN96**, **DPS16**, **VIS16**, **DIBU**, or **default**. (The word **default** selects DPIN96.)
- 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 D10 pin name.

- The following is an example of a PINLIST definition:

```
#####
##  PinList Definition
#####
PINLIST
ANALOG_VDD          DPS16    POW    0    2
HOLDn               DPIN96   I      1   17   hold_n holdn
WPn                 DPIN96   O      1    8   wp_n
anapadext_data_n   VIS16    ANA    0    3
END PINLIST
```

Note that pin ANALOG\_VDD uses channel 2 of a DPS16 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 D10 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) the DC Levels



**BACKGROUND:** Simulation output files, and even STIL files, do not typically define DC levels for the signals. However, using configuration file structures, Velocity provides you with a way to include levels information with your auto-generated D10 test program.

The LEVELS block allows you to define, for any pin or group of pins, power supply levels, input drive levels, and output threshold levels.

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

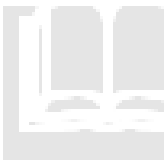
- On the first line, use the keyword **LEVELS** followed by a pin or group name. Optionally, you can use the word **default** for the pin specification to indicate all pins.
- On the next line, use the keyword **POWER** followed by a voltage value. This will be the master power supply voltage level.
- On subsequent lines, use the following keywords followed either by a voltage value or a percentage:

**VIH** – Input voltage for a logic high

**VIL** – Input voltage for a logic low

**VOH** – Output threshold voltage for a logic high

**VOL** – Output threshold voltage for a logic low



**BACKGROUND:** If you specify a level as a percentage, Velocity interprets it as a percentage of the POWER level. This provides a convenient way to scale levels with a device power supply voltage.

- For the last line, use the keywords **END LEVELS**.



- The following is an example of a Levels definition:

```
LEVELS default
POWER 3.0V
VIL 0.8V
VIH 2.0V
VOL 30%
VOH 50%
END LEVELS
```

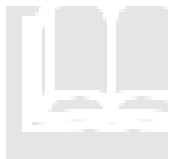
## 6.7 Create (or Edit) Custom Timing



**BACKGROUND:** Although Velocity will create appropriate Time Sets for your D10 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 D10 will be able to use the resulting STIL 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 D10 STIL 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).

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
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.

- 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.