Rapid Prototyping of Application-Specific Signal Processors (RASSP)

Design Tool Encapsulation Document

Version 2.0

Lockheed Martin
Advanced Technology Laboratories
1 Federal Street
Camden, NJ 08102

1.0 Introduction

The purpose of this document is to describe the procedure for encapsulating design tools into the Build 2 version of the RASSP Enterprise Framework. It does not address the installation of the individual design tools.

2.0 Encapsulation Procedure

The encapsulation procedure basically consists of two steps. First the design tools are declared and then they are instantiated for a particular project. Design tools are declared within each task of a workflow, and instantiated by creating a tool definition file. A tool definition file is an ASCII file with a *.tol* extension.

2.1 Tool Declarations

Design tools are declared for each workflow process. A workflow represents the implementation of a design process (e.g. module design, ASIC design, etc.). For RASSP, workflows were developed using Intergraph's DMM (Design Methodology Manager) tool. The DMM Builder provides a GUI (Graphical User Interface) for constructing workflows. Figure 1 shows a portion of a RASSP workflow constructed using DMM Builder.

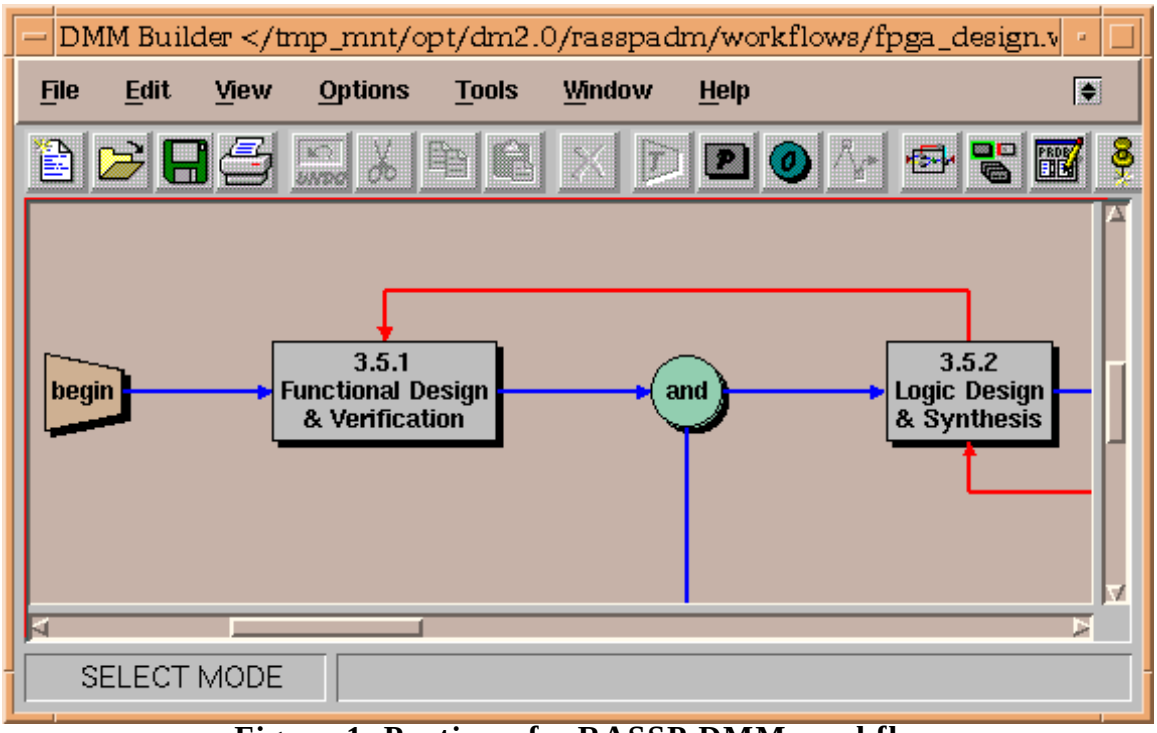

Figure 1. Portion of a RASSP DMM workflow

Design tools are declared in the attributes folder of the definition palette for each workflow process at the time the workflow is constructed in DMM. To access the attributes folder, within DMM Builder, simply double-click on a workflow process with the left mouse button. When the process definition palette is displayed, select the attributes folder. Figure 2 shows the attributes folder for workflow process 3.5.1 from figure 1.

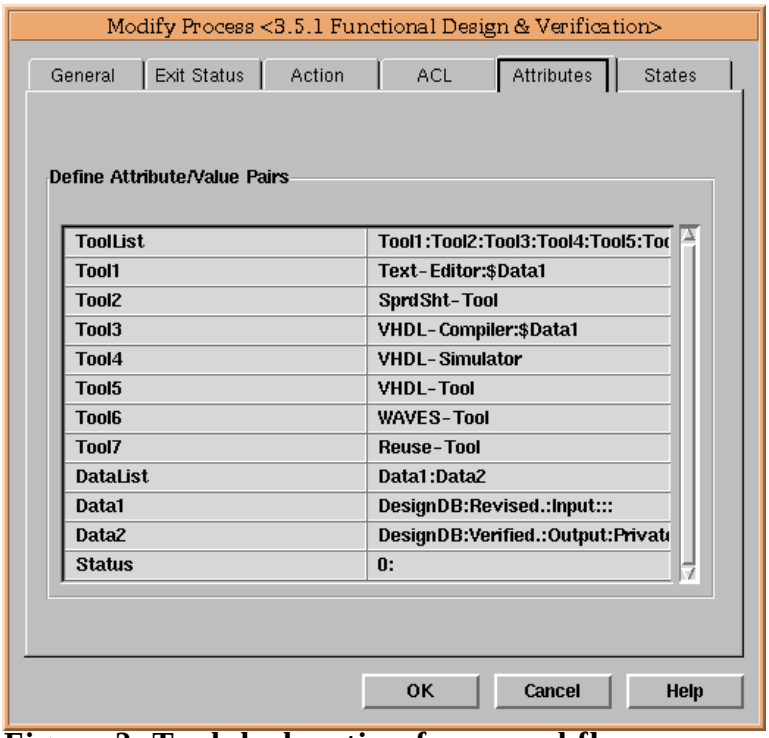

Figure 2. Tool declaration for a workflow process

The attributes folder contains attribute/value pairs. An attribute named *ToolList* is used to specify a list of tools that will be invoked within that process. It's value is a list of attribute names separated by colons.

ToolList *Tool1:Tool2:...:Tooln*

where *Tool1*, *Tool2*,...,*Tooln* are themselves attribute names which will be defined in separate lines. The value for each *Tool1*, *Tool2*,...,*Tooln* attribute is also a list of fields separated by colons. These fields are defined as follows:

Tool1 *generic tool name[:$<arg 1>:$<arg 2>:...:$<arg n>]*

where *generic tool name* is the declaration of a design tool. This field may not contain spaces or underscores. *[:$<arg 1>:$<arg 2>:...:$<arg n>]* are optional fields which would contain command line arguments that are passed to the tool upon invocation. These command line arguments would most likely be input and output files which are attached to business items. More than one argument may be specified, but it must be separated by a colon ":". There is no upper limit on the number of design tools that can be declared for each workflow process.

Another attribute named *DataList* is used to specify the DM2 business items that have been defined as inputs to, outputs from, and controls for, the workflow process. This attribute will not be discussed in this document. The last line which contains the attribute/value pair

Status *0:*

is important and is required for all workflow processes. When the Attributes folder is completed, click on "OK" to close and save it.

Figure 2 shows seven design tools declared for workflow process 3.5.1. They are: *Text-Editor, SprdSht-Tool, VHDL-Compiler, VHDL-Simulator, VHDL-Tool, WAVES-Tool,* and *Reuse-Tool*. These are generic names for the design tools which will be instantiated with specific tool names from the tool definition files. Design tool declarations can be added and deleted from workflow processes after the workflow has been developed by editing the workflow file within the DMM Builder.

2.2 Tool Instantiations

Every design tool that is declared in the DMM workflows must be instantiated. Instantiation is accomplished by creating a separate tool definition file for each design tool. The tool definition file must have a *".tol"* extension. The tool definition files are all stored in a single directory that is defined when the RASSP environment is installed The syntax for the tool definition file is as follows

Tool File Format 2.0.0
<file name>:
$TOOL_ARG_LIST
<arg 1>:<arg value>
<arg 2>:<arg value>
.
.
<arg n>:<arg value>
$END
$TOOL_OPT_LIST
<opt name>:<value>:<required>:<default>:<prompt>:<description>
$END
<toolpad name>:<tool name>:<working dir>:<icon file>:<cmd argument>
$TOOL_ENV_LIST
<env 1>:<env value>
<env 2>:<env value>
.
.
<env n>:<env value>
$END

All lines start in the first column of the file. Fields are separated by colons ":", therefore no colons are permitted in any of the fields. Fields that are enclosed by "<>" are fields that are replaced with tool specific information. Spaces are not permitted in the following fields: *<arg 1..n>*, *<opt name>*, *<value>*, *<required>*, *<default>*, *<tool name>*, *<working dir>*, *<icon file>*, and *<env 1..n>*. Spaces are permitted in the *<arg value>*, *<prompt>*, *<description>*, *<toolpad name>*, *<cmd argument>*, and *<env value>* fields.

The *<file name>* field contains the base name of the tool definition file without the extension (e.g. VHDL-Tool:). The trailing colon is required for this field.

The $TOOL_ARG_LIST section is where command line arguments for the tools are defined. The values of the arguments will be passed in, in the order they are specified, when the tool is invoked. Arguments in this section can be referenced in the *<toolpad name>*, *<cmd argument>*, and *<env value>* fields, as will be explained in the next section. The end of this section is indicated by the $END line.

The $TOOL_OPT_LIST section is where all optional arguments are defined. Optional arguments are usually switches that can be set in the command line when invoking the tool

(e.g. qvcom -nodebug <vhdl file>). A description of the fields in this section is as follows: The <*opt name*> field is the string name of the optional argument. The <*value*> field contains the argument's value. The <*required*> field indicates if this argument is required. If this field contains the string "required", then the argument is required. Otherwise it's optional. The <*default*> field is not currently used and should be left blank. The <*prompt*> field defines the label that can be displayed in the GUI. The <*description*> field contains a descrition of the optional argument. The optional arguments defined in this section are referenced in the <*cmd argument*> field, as will be explained in the next section. The end of this section is indicated by the $END line.

The <*toolpad name*> field is where the name that appears on the DMM toolpad is defined. The <*tool name*> field is the executable path name. Specifying the full path is not necessary if the path is contained in the user's default PATH setting.

The <*working dir*> field specifies the working directory for the tool. The <*icon file*> field contains the name of the file containing the tool icon that is also displayed on the DMM toolpad. The <*cmd argument*> field contains the command line arguments that will be passed to the tool when it is invoked.

The $TOOL_ENV_LIST section contains the settings of any special UNIX environment variables that are needed by the tool. These environment variables are set just before the tool is invoked and remain in existence during execution of the tool. Once the tool is exited, the environment variable settings are removed.

2.3 Examples

The example files in this section are actual tool definition files for design tools that are currently encapsulated within the Build 2 version of the RASSP Enterprise Framework. These examples also correspond to the tools declared for workflow process 3.5.1 from figure 2.

```
Tool File Format 2.0.0
Text-Editor:
$TOOL_ARG_LIST
iFile
$END
$TOOL_OPT_LIST
$END
Emacs editor :emacs::emacs.bmp:$iFile
$TOOL_ENV_LIST
$END
```

Example 1. Text-Editor.tol

Example 1 contains the "Text-Editor.tol" file which instantiates the emacs editor. The <*file name*> field contains the base name of the tool definition file which in this case is "Text-Editor". The $TOOL_ARG_LIST section contains one argument "iFile" which represents the input file to the emacs editor. It's value is passed in from the attributes folder of the definition palette for the DMM workflow process as indicated in Figure 2. The $TOOL_OPT_LIST section is empty since there are no additional arguments to be defined for the emacs editor. The <*toolpad name*> field contains "Emacs editor" which is displayed inside the DMM toolpad when the workflow process is executed. The <*tool name*> field contains the executable name "emacs" for invoking the tool. The <*working dir*> field is not utilized for this tool and is left blank. That is why there are two colons "::" between the

<tool name> and *<icon file>* fields. The *<icon file>* field contains "emacs.bmp" which is the name of the file containing the icon that also gets displayed inside the DMM toolpad when the workflow process is executed. The *<cmd argument>* field contains only one value, "$iFile", which is passed into the emacs editor as a command line argument. This argument contains the name of the file that the editor is invoked on. There are no UNIX environment variables defined for this tool, so the $TOOL_ENV_LIST section is empty.

```
Tool File Format 2.0.0
VHDL-Tool:
$TOOL_ARG_LIST
$END
$TOOL_OPT_LIST
$END
Summit VisualHDL:visual_hdl::vishdl.bmp
$TOOL_ENV_LIST
VISUALHDL:$HOME
$END
```

Example 2. VHDL-Tool.tol

Example 2 contains the "VHDL-Tool.tol" file which instantiates the Summit Visual HDL design tool. The *<file name>* field contains the base name of the tool definition file which in this case is "VHDL-Tool". The $TOOL_ARG_LIST section does not contain any arguments, so it is left blank. Similarly the $TOOL_OPT_LIST section is empty because there are no addition arguments to be defined for the Summit Visual HDL tool. The *<toolpad name>* field contains "Summit VisualHDL" which is displayed inside the DMM toolpad when the workflow process is executed. The *<tool name>* field contains the executable name "visual_hdl" for invoking the tool. The *<working dir>* field is not utilized for this tool and is left blank. That is why there are two colons "::" between the *<tool name>* and *<icon file>* fields. The *<icon file>* field contains "vishdl.bmp" which is the name of the file containing the icon that also gets displayed inside the DMM toolpad when the workflow process is executed. Since the $TOOL_ARG_LIST section is empty, the *<cmd argument>* field is also left empty. The $TOOL_ENV_LIST section contains the UNIX environment variable "VISUALHDL" which is set to the user's home directory when the tool is invoked.

```
Tool File Format 2.0.0
VHDL-Compiler:
$TOOL_ARG_LIST
iFile
$END
$TOOL_OPT_LIST
dSwitch:-nodebug::::optional debug switch
$END
MGC QuickVHDL Compiler:new_qvcom:$HOME/$USER.wl:qvcom.bmp:$dSwitch
$iFile
$TOOL_ENV_LIST
QUICKVHDL:$HOME/quickvhdl.ini
$END
```

Example 3. VHDL-Compiler.tol

Example 3 contains the "VHDL-Compiler.tol" file which instantiates the Mentor Graphics QuickVHDL compiler. The *<file name>* field contains the base name of the tool definition

file which in this case is “VHDL-Compiler”. The $TOOL_ARG_LIST section contains one argument “iFile”. It’s value is passed in from the attributes folder of the definition palette for the DMM workflow process as indicated in Figure 2. The $TOOL_OPT_LIST section contains the optional argument “dSwitch” which represents the optional nodebug switch that can be set for this tool. The <*value*> field contains “-nodebug” which is the value for this switch. The <*required*>, <*default*>, and <*prompt*> fields are all left blank. The <*description*> field contains “optional debug switch” as a brief description of this switch. The <*toolpad name*> field contains “MGC QuickVHDL Compiler” which is displayed inside the DMM toolpad when the workflow process is executed. The <*tool name*> field contains the name “new_qvcom”. This is the name of the UNIX shell script which invokes the tool. The <*working dir*> field contains the tool’s UNIX working directory which is set to “$HOME/$USER.wl”. The <*icon file*> field contains “qvcom.bmp” which is the name of the file containing the icon that also gets displayed inside the DMM toolpad when the workflow process is executed. The <*cmd argument*> field contains two values “$dSwitch” & “$ifile” which are passed into the Mentor Graphics QuickVHDL compiler as command line arguments. There are no UNIX environment variables defined, so the $TOOL_ENV_LIST section is empty.

3.0 Design Tool Invocation

This section describes how design tools are invoked once they have been encapsulated into the DMM workflows. When the RASSP system is executed, the DMM displayer will display the project workflows. Workflow tasks which are startable may be executed by double-clicking on it. Once the workflow task has been started, DMM will display a tool launch pad which contains the tool names and icons of the tools that were declared and instantiated for that particular process. Figure 3 shows an example of the DMM toolpad from which design tools are invoked.

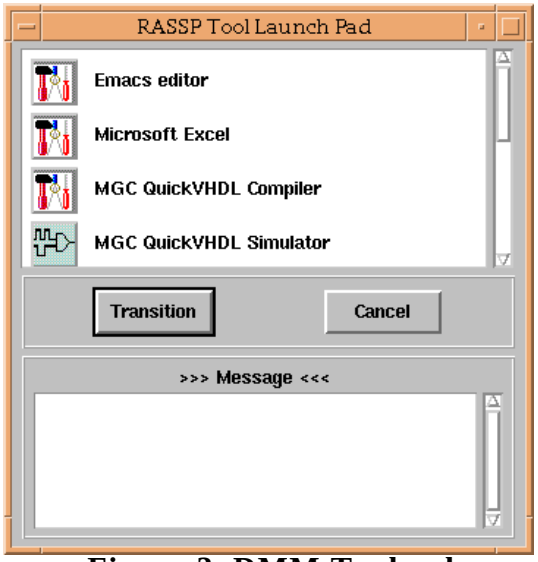

Figure 3. DMM Toolpad

The tool names and icons that appear in the DMM toolpad are all defined in the tool attribute files. To invoke a tool, the user simply clicks the icon next to the tool name. The design tool is then invoked.

4.0 Glossary

ASIC	Application Specific Integrated Circuit
DMM	Design Methodology Manager
DM2	Document Manager
GUI	Graphical User Interface
RASSP	Rapid Prototyping of Application Specific Signal Processors