Chapter 2

2.0 Modeling Conventions

2.1 Introduction

2.1.1 Overview

This chapter will provide some background to the modeling approach developed under the SHARP TIREP project. Additionally, it will address some of the recommended modeling conventions which have been established for use on the SHARP TIREP project. These conventions include file naming conventions and VHDL file header information. The conventions provided herein are recommendations. Their use is not dictated by any of the standards or specifications relating to VHDL.

2.1.2 Selected Modules

For the SHARP TIREP effort, the module selection of Table 2.1.2-1 was made. Module selection was based upon high use modules with current or expected near term obsolescence problems. All modules are the SEM format A configuration.

Key Code	*Description*	*TIREP Team Member*
BDA	*Gate, NAND*	*Lisa Ceder*
BMG	*Counter, Binary/BCD*	*David Broadhead*
DQD	*Standard Serial Output*	*Charles Rogers*
FBG	*Adder, Discrete Sum*	*Charles Rogers*
GDA	*Gates, NAND*	*Lisa Ceder*
KDP	*Shift Register*	*Ed Woods*
MHK	*Encoder, Parity*	*Darin York*
RDH	*Logic Unit, Arithmetic*	*Gary Hout*
WDU	*Gates, 4 8-Input NAND*	*Lisa Ceder*

Table 2.1.2-1. SHARP TIREP Module Selections.

2.2 File Conventions

2.2.1 VHDL File Naming Conventions

The following VHDL file conventions were established to define a format in which a complete VHDL model could be transferred to a manufacturing environment for use within that environment.

2.2.1.1 All VHDL files shall reside in the "VHDL" subdirectory of a specified job. It is recommended that there be no additional directory structure under "VHDL", although this is not prohibited. It has been noted that directory structures can complicate file transfers between computer systems.

2.2.1.2 There shall be no constraint on the number of files present in the "VHDL" directory.

2.2.1.3 A VHDL model file shall be named according to the following convention. The < filename> shall be generated by the design engineer using the guidelines imposed by the design team. These filenames should be consistent with the naming conventions for VHDL identifiers. It is recommended that the < filename> not exceed 8 characters in length in order to support the file use and analysis on PC-DOS based platforms. If filenames are used which exceed 8 characters in length, it is recommended that each filename be unique within the first 8 characters. The < extension> is a 3 character file extension as defined below.

<filename>.<extension>

2.2.1.4 All files contained in the "VHDL" directory shall have one of the file extensions identified in Table 2.2.1.4-1.

2.2.1.5 All VHDL model files shall contain only ASCII text.

Extension	File Type/Description
.VHD	A VHDL source file.
.WAV	A VHDL WAVES source file. It is not required that this extension be used for WAVES files. It is acceptable practice to use .VHD for these files.
.DAT	A model specific data file. Examples of data files include the vector file for a WAVES testbench, a component programming file, etc.
.TXT	An ASCII text file which contains information about the VHDL model, it's use and application.

Table 2.2.1.4-1. VHDL File Extensions.

2.2.1.6 DI-EGDS-80811, paragraph 10.3 dictates the presence of 7 or more text files (.TXT extension) which contain model content information. These files shall be included with all VHDL models. It is recommended that these files be named in accordance with Table 2.2.1.6-1.

Filename	Contents
FILE_1.TXT	Names of files included with delivery
FILE_2.TXT	File overview
FILE_3.TXT	Order of analysis
FILE_4.TXT	List of leaf level modules used
FILE_5.TXT	List of revised VHDL modules
FILE_6.TXT	List of original VHDL modules
FILE_7.TXT	Module/testbench cross reference
etc.	Auxiliary information files

Table 2.2.1.6-1. DID Documentation Files.

2.2.2 Supplemental Recommendations.

2.2.2.1 On UNIX based platforms, file naming conventions can be quite a bit more flexible as the length of the filenames and extensions is not restricted to the extent that PC-DOS naming conventions are. Therefore, it is permissible to take advantage of this additional flexibility as long as each filename used is unique within the first 8 characters. Additionally, the first three letters of the file extension should conform to one of the extensions listed in 2.2.1.4 above.

2.2.2.2 When possible, it is recommended that the filename include one of the suffixes of Table 2.2.2.2-1 to indicate the type of VHDL file it is. In support of PC-DOS platforms, the convention could be adopted where the filename is fixed at 7 characters allowing the eighth character to be selected from the list below.

2.2.2.3 When developing VHDL modules which are compliant with DI-EGDS-80811, the number of files in the module can become quite significant (30 or more). In order to help control the number of files in a VHDL module, it is recommended that in the completed module, an entity and all of it's supporting architectures be provided as a single file. In a similar fashion, it is recommended that package bodies and package declarations be included in the same file. This is a rather loose recommendation as there may be overriding considerations which justify the separation of these elements.

2.3 A VHDL File Header

2.3.1 An Example VHDL Header

Suffix	VHDL File Type
_a	architecture body (may be a combination of structural and behavioral)
_b	behavioral architecture
_c	configuration
_e	entity declaration
_p	package
_s	structural architecture
_t	testbench
none	entity/architecture pair

Table 2.2.2.2-1. File Suffix Conventions.

The code of Figure 2.3.1-1 is a sample VHDL file header. It is recommended that this header be used to provide information about the file. EIA-567A, paragraph 5.2 imposes requirements for the presence of much of this information. Additionally, DI-EGDS-80811, paragraphs 10.2.7 and 10.2.8 require some of the same information. Table 2.3.1-1 lists the header information required by these documents.

Information Requirement	EIA-567A	DI-EGDS-80811
Originator (Author)	5.2.1	10.2.8
DOD Identification Number (if one exists)		10.2.8
Design Unit Name		10.2.8
Revision Identifier	5.2.3	10.2.8
Revision History	5.2.4	10.2.8.1
Design Unit Description	5.2.2	
Analysis Dependencies	5.2.5.1	
Date of Model Completion	5.2.3	
Code Limitations	5.2.5.2	10.2.7
Modeling Approaches		10.2.7
Model Discrepancies	5.2.5.4	
Model Fidelity	5.2.5.3	10.2.7
Development Platform	5.2.6	
VHDL Software Version	5.2.7	
Supplementary Information	5.2.5.5	10.2.7

Table 2.3.1-1. VHDL Header Requirements.

2.3.1.1 The first section of the header provides information on the author and the author's organization. This is normally the organization responsible for the generation of the model. Whenever possible, the author should provide an E-Mail address as this is becoming a favored mechanism for communication on questions and concerns. Additionally, this section will reference the contract/specification and project for which the file was developed. Once the file is complete and validated, this section of the header will only be changed if a new point of contact or organization needs to be added.

2.3.1.2 The second section contains the revision history. Once the file is complete, this section of the header should be changed each time there is a change to the file.

```
--*****************************************************************
--    (c) Copyright 1994 by the
--         Naval Air Warfare Center, Aircraft Division, Indianapolis
--    Source
--         Author(s):          Charles K. Rogers
--         Organization:       NAWC-ADI
--                             Code 306, MS-42
--                             6000 E 21st St
--                             Indianapolis, IN 46219-2189
--                             Phone: 317-353-3579
--                             EMail: ROGERSC1@po2.nawc-ad-indy.navy.mil
--         Reference:          (Contract #, Specification, etc.)
--         Project:            VHDL Style Guide
--    DESC Certification
--         Status:             TBD
--*****************************************************************
--    Revision History
--         Version:            2.0
--              Date:          24 March 1994, 7 April 1994 - Syntax correction
--              Comments:      Model upgraded to implement enhanced control algorithm
--         Version:            1.0
--              Date:          12 September 1993
--              Comments:      Original Release
--*****************************************************************
--    Module Description
--         File:               header.vhd
--         Module Name(s):     (entity/architecture/package/etc.)
--         Constraints:        (critical path delay, etc.)
--         Limitations:        (bus functional only, partial implementation, etc.)none
--         I/O Format(s):      (I/O formats and/or unusual interface characteristics)
--         Purpose and Use:    (text description of module purpose and
--              use. provide as much detail here as reasonable.)
--         Notes:              (supplementary module information)
--*****************************************************************
--    Standard Libraries/Packages
--         std_logic_1164      standard multi-value logic package
--    Associated Packages (order of analysis implied)
--         eia_567_ev          standard electrical view package
--         eia_567_tv          standard timing view package
--    Component Models
--         i_driver            generic input driver
--    Platform:                (486/33 PC, Sparc 2, etc.)
--    Software/Version:        V-System for Windows, Version 3.0
--*****************************************************************
--    Design Specification Elements
--         (Identification of those elements in the subject file which
--         must be tailored for a specific design or application. This
--         allows the file to be employed as a template by other design
--         engineers if it is expedient to do so.)
--*****************************************************************
```

Figure 2.3.1-1. A Sample VHDL Header.

2.3.1.2.1 The version number shall be 1.0 initially. For minor changes, the decimal portion of the version number shall be incremented. For major changes, the integer portion of the version number shall be incremented and the decimal portion shall be set to 0. (Note: At this point in time, it is left to the author to determine the difference between a major change and a minor change.) For simple corrections (such as syntax or spelling), the version does not need to be altered.

2.3.1.2.2 The date provided should be the date on which the indicated version of the file was completed and validated. The only exception to this occurs in the event of a simple correction (see above). In this case, the date of the correction should be appended to the date of the revision release.

2.3.1.2.3 The comments section of this header should detail the changes which were made to reach the version identified.

2.3.1.2.4 The first revision in the historical listing should be the latest revision of the file.

2.3.1.3 The next section of the header contains information about the file. This section details the filename, the module names which are generated by this file when it is analyzed, constraints, limitations, etc. Of particular importance in this section is the "Purpose and Use" description. Under this entry, a detailed text description of the function of the file should be provided. Any algorithms used by the file should be described and, in the absence of other documentation on the algorithm, it's operation should be explained.

2.3.1.4 Supporting files are listed in the next section of the header. These files include libraries, packages, components, etc.) which are required by this file. These supporting files must be analyzed in the order listed in order to support the subject file. Also included in this section is the platform and software which were used in the development of the file.

2.3.1.5 The final section of the header identifies design specification elements. This section provides the design engineer with a quick reference to the elements in the file which are design specific. This helps expedite the use of the file as a template for the development of other VHDL designs

2.4 Coding Conventions

2.4.1 Design Unit Naming Conventions

2.4.1.1 Design unit names are selected by the design engineer and should be coordinated with the design team to insure that proper model interfaces are developed. The length of names along with standard prefixes and/or suffixes should be established by the design team. There are five types of design units which include entity declarations, architecture bodies, package declarations, package bodies and design configurations.

2.4.1.2 Entity names should correspond to integrated circuit hardware device names that are being modeled (i.e. one-to-one relationship with schematic device names, see DI-EGDS-80811, paragraph 10.2.2.4). These names are typically found in the design documentation parts list. If no parts list exists, naming should be coordinated with the appropriate Design Agent and/or System Engineer.

2.4.1.3 Architecture names should indicate the type of design abstraction being modeled. A naming convention is provided in Table 2.4.1.3-1.

Design Abstraction	Architecture Name
Behavioral	arch_<design_name>_beh
Structural	arch_<design_name>_str
Data Flow	arch_<design_name>_dat
Register Transfer Level	arch_<design_name>_rtl
Testbench	arch_<design_name>_tb

Table 2.4.1.3-1. Architecture Naming Conventions.

2.4.1.4 Package names should indicate the category of subprograms that are declared in the package. For example, if a set of functions are created to perform trigonometric operations like Sin, Cos, Tan, etc., then a good name for this package would be "math_trig". If this set of trigonometric functions were tailored to work with IEEE standard logic only, then a good name might be "math_trig_1164".

2.4.1.5 Package body names must always be the same as package names (see paragraph 2.6 of IEEE-STD-1076).

2.4.1.6 Configuration names should indicate which architecture is being configured with a specific entity. For example, if there are two architectures for an entity, one totally behavioral and one built structurally from a library of primitive models, then two configurations are desired. The first configuration might be named "config_<design_name>_beh" and the second configuration might be named "config_<design_name>_str".

2.4.1.7 It is recommended that signals be modeled using the STD_LOGIC resolved type or STD_ULOGIC unresolved type of the STD_LOGIC_1164 package. Attempts to limit the design to a subtype of STD_LOGIC can result in problems in the form of interface inconsistencies as the complete VHDL model is developed.

2.4.2 Other Naming Conventions

2.4.2.1 Port names should correspond directly with the names of the pins of the device that is being modeled.

2.4.2.2 Generic names should reveal how the generic is going to be used in the model. For example, if the generic represents a timing parameter for propagation delay from input to output when the output signal transitions from high to low, then an appropriate name for the generic might be "tphl".

2.4.2.3 Signal names used within a behavioral architecture are named at the discretion of the design engineer. Signal names used to connect primitive component models instantiated in a structural VHDL model, should be coordinated with any existing schematics (see paragraph 10.2.4.1 of DI-EGDS-80811). If a schematic does not exist, then the recommended naming convention is to utilize a prefix (as specified by the design team) and concatenate that prefix onto the signal name or number for signals internal to the model. A suggested naming convention is provided in Table 2.4.2.3-1.

Model	Prefix	Notes
WAVES Testbench	w_<signal_name>	waveform generator stimulus/response
	p_<signal_name>	predicted response
Architecture	n_<signal_name>	for node

Table 2.4.2.3-1. Internal Signal Naming Conventions.

2.4.2.4 Variable names used within a VHDL model are at the discretion of the design engineer. It is strongly recommended that names be used that will add to the readability of the VHDL code, as it pertains to the use of the variable in the model. It is recommended that the purpose of a constant be explained in a comment.

2.4.2.4.1 Pointers are a special case use of variables for indexing into matrices. It is recommended that the suffix "_ptr" be added to a variable that is used as a pointer.

2.4.2.5 Constant names used within a VHDL model are at the discretion of the design engineer. It is strongly recommended that names be used that will add to the readability of the VHDL code, as it pertains to the use of the constant in the model. It is recommended that the purpose of a constant be explained in a comment.

2.4.2.6 Label names used within a VHDL model are at the discretion of the design engineer. It is strongly recommended that names be used that will add to the readability of the VHDL code, as it pertains to the use of the label in the model.

2.4.2.6.1 In structural VHDL models of electronic assemblies, component instantiation labels shall form the reference designator for the components used. These labels shall be of the form

xn

where "x" is a 1 or 2 character class designation letter defined by ANSI Y32.2 and "n" is a sequential integer starting with the number 1 for each class of device.

2.4.2.6.2 In structural VHDL models of components, the component instantiation label shall be of the form

Un

where "U" is that character and "n" is a sequential integer starting with the number 1.

2.4.3 Code Readability

Code readability is the ease with which a competent engineer with a sound working knowledge of VHDL, can read and comprehend the function performed by a VHDL model. Models that are well commented and easily readable provide a cost savings by permitting re-use of the VHDL model and simplifying the life cycle maintenance. There is no measure or test for code readability. It is up to individual design engineers/teams to check themselves by conducting code walk-through(s) in order to assess the quality and readability of their models.

2.4.3.1 Capitalization of either VHDL reserved words or user generated names is necessary for readability. Since the VHDL LRM (IEEE-STD-1076) is case insensitive, it does not make a difference which way capitalization is implemented. During VHDL model development, it is convenient for the VHDL writer to use lower case for everything. Therefore, use of a VHDL model post-processor to implement one of the two capitalization schemes is recommended. Capitalization of VHDL reserved words is the recommended scheme.

2.4.3.2 Comments are necessary for readability. There are two places recommended for comments in VHDL models. One place is in the VHDL file header, as described in paragraph 2.3. The second is in between the lines of code. The extent and detail of the comments is left up to the VHDL design engineer. Too may comments can be a distraction when trying to read and comprehend code. On the other hand, too few or cryptic comments can frustrate someone who is trying to understand the functionality being modeled. Again, there is no measure or test for adequacy of comments. It is up to individual design engineers/teams to check themselves by conducting code walk-throughs.

2.4.3.3 Indentation is strongly recommended to enhance readability. Indentation should be provided for each statement in a VHDL design unit. The amount of indentation (number of spaces or tabs) should be indicative of the location of the statement in the design unit hierarchy.

2.4.3.3.1 It is recommended that two spaces be used for each level of indentation. This provides for good readability without adding excessive numbers of characters to the VHDL code.

2.4.3.3.2 The use of tabs is not recommended due to the varying manners in which tabs are handled between platforms and printers.

2.4.3.4 It is an unacceptable practice to purposely obscure or limit the readability of VHDL code. To do so violates the very intent of VHDL.

2.4.4 Code Walk-Throughs

2.4.4.1 A code walk-through might be equated to the design review of a VHDL model. A code walk-through is conducted by a review team. Its purpose is to identify potential problems and problem areas. A code walk-through is not a forum for fault finding or belittling a colleague.

2.4.4.2 A review team should consist of the following individuals. The total number of individuals involved in a code walk-through should be no more than 7 (including the author and the review leader).

The review leader - chairs the code walk-through and keeps it on track.
The model author - the design engineer who generated the model being reviewed.
The recorder (one of the reviewers) - records issues raised and resolutions made.
The reviewers (3 to 5 people) - identify issues or potential problem areas in the code.

2.4.4.3 In preparation for a code walk-through, the model author is responsible for providing code listings to the review team members at least 1 day before the walk-through. This gives the reviewers an opportunity to review the code prior to the meeting. This preparation effort should not require more than 2 hours of work for any member of the review team.

2.4.4.4 The duration of the code walk-through meeting should be no more than 2 hours.

2.4.4.5 During the code walk-through, the model author presents the VHDL model to the remaining members of the review team by walking them through the code. The model author should describe the modeling approach and identify coding features which strengthen or limit the model. As issues are raised, the recorder should make note of them along with any resolutions reached. Caution: The code walk-through is not a problem solving meeting. The review leader is responsible for keeping the meeting on track. If an issue can not be readily resolved, it should be tabled for appropriate action. The following guidelines apply for a code walk-through.

- Review the model, not the model author. The tone of the meeting should informal and constructive. Comments intended to embarrass or belittle any member of the review team are clearly unacceptable.
- An agenda for the code walk-through should be set and maintained. The review leader is responsible for insuring that the meeting does not drift or become bogged down.
- Debate should be limited. When an issue is raised that cannot be readily resolved, it should be recorded and tabled for further action as appropriate.
- Identify problem areas, but don't try to solve every problem. Once a problem is identified, the solution can be worked out by the model author with the appropriate assistance. It is not incumbent on the review team to resolve the problem during the code walk-through.
- The recorder is responsible for noting issues and resolutions raised during the meeting. A wallboard may be used to help track these items. Following the meeting, the recorder will be responsible for generating a report on the review which identifies all issues raised, the resolutions that were made and remaining action items yet to be resolved.
- The number of participants involved on the review team should be limited. Additionally, all review team members must prepare for the walk-through by reviewing the VHDL models provided by the author. It is an appropriate practice for the review leader to require comments from the review team members prior to the code walk-through as an indication that they have made the necessary preparation.
- A checklist should be provided to insure that all appropriate aspects of the code walk-through are reviewed. As an example, a checklist might include the following items.

    - does the model perform the desired function(s)?
    - are the interfaces (ports) correct (type, polarity, etc)?
    - is the model complexity reasonable?
    - have error handling features been implemented?
    - have built-in-test features been implemented?
    - are there any tool dependent constructs used?
    - is there compliance with the coding standards developed by the design team?
    - is the code adequately commented for readability?
    - are there any incorrect or ambiguous comments?
    - are there misspellings and/or typos?

- Code walk-throughs should be scheduled in order to be effective. They are not spur-of-the-moment design reviews. Time should be scheduled to accomplish this task. Additionally, time should be scheduled to implement model changes resulting in this type of review.
- It is beneficial that all reviewers receive formal training. Such training should stress both review criteria along with the human psychological side of the review process.
- Review of previous reviews can be beneficial in identifying problems with the review process.

2.5 The TIREP Model

2.5.1 The TIREP Model Structure

The diagram of Figure 2.5.1-1 provides an overview of the TIREP model structure.

2.5.1.1 Due to the simplicity of the modules selected for this project, only one core model is required for the entire module. Larger or more complex models may implement several core models in order to realize the function of the module. The core model is implemented in behavioral VHDL which is technology independent and generally synthesizable.

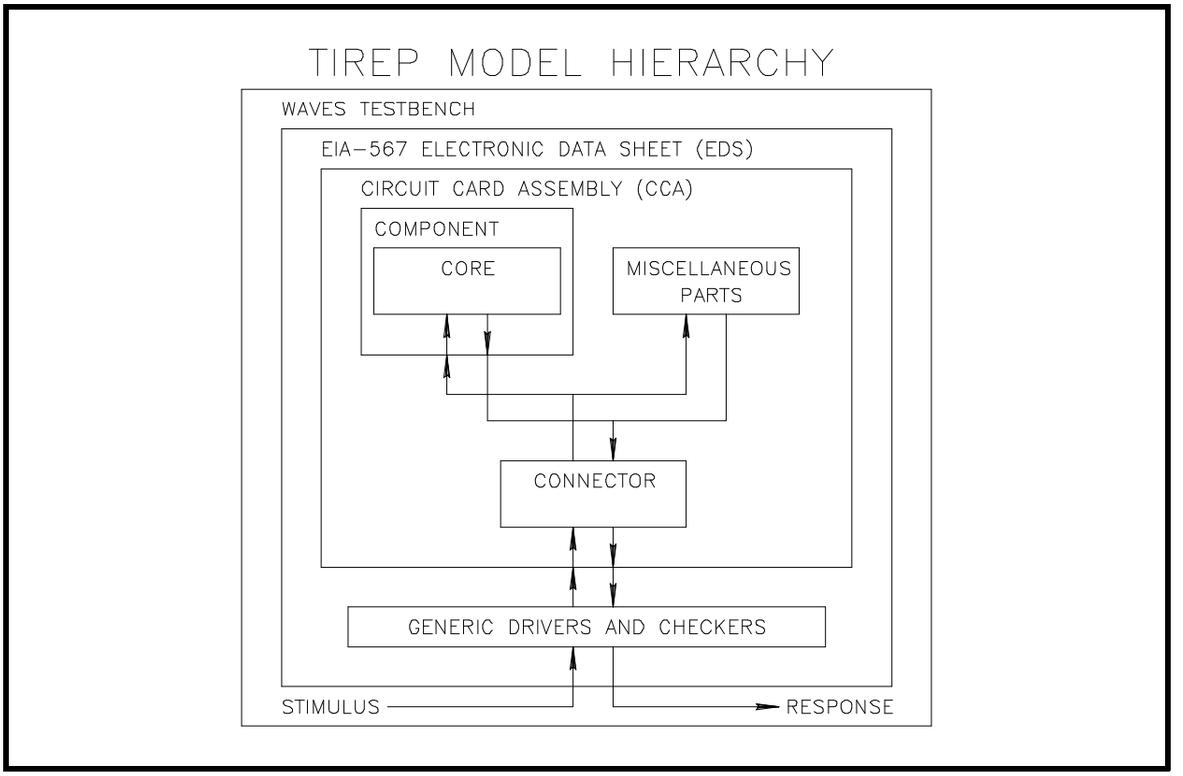

Figure 2.5.1-1. The TIREP Model Structure.

2.5.1.2 Around the core model, a component model is assembled. This model will include timing characteristics, along with interface parameters. Additionally, this model will have it's own WAVES testbench as required by DI-EGDS-80811.

2.5.1.3 The component model along with models for other circuit components are assembled structurally into the circuit card assembly (CCA) model. For the TIREP modeling approach, the miscellaneous components include mechanical parts such as the module frame, rivets, etc. With the exception of the connector, these miscellaneous components will feature either a null or a passive behavior (such as a filter capacitor, a rivet, or a pull-up resistor).

2.5.1.4 The connector can be modeled in a couple of ways. One way would be to model it as an input only component with a null behavior. The alternative would be to model the inputs and outputs of the connector and include a propagation delay through the connector. It is this second option that is depicted in the figure above. While this alternative provides a higher degree of accuracy for the application, it may not work in all cases as bidirectional signals can be difficult to implement in this manner.

2.5.1.5 The miscellaneous components (including the connector) used in the TIREP modeling approach do not fulfill the requirements of the VHDL DID. It was felt that modeling to the level required by the VHDL DID for components with minimal to non-existent behavior is beyond the intent of the DID or the requirements of this project.

2.5.1.6 Asembled around the CCA model is the Electronic Data Sheet (EDS) model. Since this is the level at which the SHARP modules are specified, it is here that the model captures the specification. At this level, a timing shell is built around the CCA model using processes which implement the timing checks and propagation delay characteristics identified under the EIA-567A modeling approach.

2.5.1.7 Finally, a WAVES testbench has been assembled for the EDS module level model. In general, the testbench is assembled around the functional test vector set provided in the module specification. In those instances where a functional vector set is not provided in the specification, an appropriate test vector set was generated for the design.

2.5.2 TIREP Component Model Files

Figure 2.5.2-1 provides a breakout of the files used in the TIREP component model. A note is included to indicate which files require modification for a new design. An indication of the amount of code which must be updated in the file is also provided. To assist in understanding the model structure, a file hierarchy is depicted.

Filename	File Function	Design Modifications
<component>_TSTT.VHD	Testbench	entire file
<component>_HDR.WAV	Waves header file	entire file
<component>_WAVP.WAV	Waveform generator	names/context clauses
<component>_VEC.DAT	Vector file	entire file
WAV_FRMP.WAV	Waves frame package	no change required
<component>_INT.WAV	Waves interface package	context clauses
<component>_OBJ.WAV	Waves objects package	context clauses
<component>_DUTP.WAV	test pins package	TEST_PINS
WAV_INT.WAV	Waves interface package	context clauses
WAV_LOGP.WAV	Value dictionary package	no change required
<component>_EDS.VHD	EDS model	entire file
<component>_COM.VHD	Component models	entire file
<component_COR.VHD	Core models	entire file
I_DRIVER.VHD	Input driver	no change required
O_DRIVER.VHD	Output driver	no change required
B_DRIVER.VHD	Bidirectional driver	no change required
ASYNC_CH.VHD	Asynchronous checker	no change required
SYNC_CHE.VHD	Synchronous checker	no change required
EIA567DS.VHD	EIA design spec package	no change required
<component>_DS_P.VHD	Design spec package	entire file
EIA_567.VHD	EIA standard package	no change required

Figure 2.5.2-1. TIREP Component Model File Structure.

2.5.3 TIREP Assembly Model Files

The structure of the circuit card assembly (CCA) model files is very similar to the component model files. The key difference lies in the fact that the assembly model is built around the CCA file which is a structural VHDL representation of the assembly. The CCA model also features an EDS shell. For the TIREP modules, this forms the module specification since it is from the module specification that this shell is derived. The file hierarchy for the assembly level model is depicted in Figure 2.5.3-1.

2.5.3.1 The < component> _EDS.VHD model is the model described in paragraph 2.5.2. DI-EGDS-80811 requires that each component model and assembly model be provided with a timing shell and a testbench.

2.5.3.2 For the component EDS models, a special operating point was included (tzero). This operating point yields a zero delay version of the component model. This operating point was added due to the manner in which the TIREP modules are specified. In this case, the characteristics of the module or assembly are specified, not the characteristics of the devices used in the module. As a result, it is necessary that the module specifications be carried with the design. Care must be taken when components are selected, however, to insure that they meet the requirements of the module.

Filename	File Function	Design Modifications
<design>_TSTT.VHD	Testbench	entire file
<design>_HDR.WAV	Waves header file	entire file
<design>_WAVP.WAV	Waveform generator	names/context clauses
<design>_VEC.DAT	Vector file	entire file
WAV_FRMP.WAV	Waves frame package	no change required
<design>_INT.WAV	Waves interface package	context clauses
<design>_OBJ.WAV	Waves objects package	context clauses
<design>_DUTP.WAV	Test pins package	TEST_PINS
WAV_INT.WAV	Waves interface package	context clauses
WAV_LOGP.WAV	Value dictionary package	no change required
<design>_EDS.VHD	EDS model	entire file
<design>_CCA.VHD	Circuit card model	entire file
<component_EDS.VHD	Component EDS models	entire file
<component>.VHD	Generic component models	entire file
I_DRIVER.VHD	Input driver	no change required
O_DRIVER.VHD	Output driver	no change required
B_DRIVER.VHD	Bidirectional driver	no change required
ASYNC_CH.VHD	Asynchronous checker	no change required
SYNC_CHE.VHD	Synchronous checker	no change required
EIA567DS.VHD	EIA design spec package	no change required
<design>_DS_P.VHD	Design spec package	entire file
EIA_567.VHD	EIA standard package	no change required

Figure 2.5.3-1. TIREP Assembly Model File Structure.

2.5.3.3 The TIREP team has elected to deviate from DI-EDGS-80811 in one area in that EDS models and testbenches were not required for such elements as resistors, capacitors, connectors or mechanical parts. It was felt that modeling these components to the extent imposed by the DID is beyond the scope of this porject and possibly outside the intent of the DID.

2.6 Tricks and Gimmicks

2.6.1 General

In the sections which follow, some of the more unusual modeling challenges will be presented. It is hoped that these will provide some insight to help deal with those challenging designs which simply do not want to work as desired.

2.6.2 A SN5476 J-K Flip-Flop Example

The first example provided is that of a SN5476 J-K flip-flop. As with many J-K flip-flops, this device features a level sensitive input. The data sheets for these devices will caution the user that the J and K inputs should not change while the clock is high. If this convention is adhered to, the SN5476 will operate like an edge-triggered device. In applications where this condition is violated, the operation of the SN5476 will differ from that of a true edge-triggered device in that the level sensitive nature of the flip-flop input will impact the operation. Figure 2.6.2-1 contains a model of the SN5476 with two architectures. The first architecture is a gate level architecture extracted from a data book. The second architecture is a behavioral VHDL description of this device.

2.6.2.1 The "dataflow" behavior was obtained from the TI data sheet for the SN5476. The double inversion on the "clk" input signal was added to prevent model oscillations under certain input test conditions.

2.6.2.2 In the "behav" architecture, two processes are used to capture the behavior of this device. The first process models the level sensitive latch while the second process implements an edge-triggered flip-flop.

```
--*****************************************************************
--    (c) Copyright 1994 by the
--         Naval Air Warfare Center, Aircraft Division, Indianapolis
--    Source
--         Author(s):              Charles K. Rogers
--         Organization:            NAWC-ADI
--                                  Code 306, MS-42
--                                  6000 E 21st St
--                                  Indianapolis,  IN  46219-2189
--                                  Phone:   317-353-3579
--                                  EMail:   ROGERSC1@po2.nawc-ad-indy.navy.mil
--         Reference:               MIL-M-28787/212
--         Project:                 SHARP   TIREP
--    DESC Certification
--         Status:                  TBD
--*****************************************************************
--    Revision History
--         Version:                 1.0
--              Date:               19 May 1994
--              Comments:           Original Release
--*****************************************************************
--    Module Description
--         File:                    sn5476.vhd
--         Module  Name(s):         sn5476 entity
--                                  arch_sn5476_datarchitecture
--                                  arch_sn5476_beharchitecture
--         Constraints:             none
--         Limitations:             none
--         I/O  Format(s):          UX01Z std_logic subtype
--         Purpose and Use:         This VHDL module contains a behavorial description
--              for the SN5476 dual j-k flip-flop.
--         Notes:                   none
--*****************************************************************
--    Standard Libraries/Packages
--         std_logic_1164           standard multi-value logic package
--    Associated Packages (order of analysis implied)
--         none
--    Component  Models
--         none
--    Platform:                     (486/33 PC, Sparc 2, etc.)
--    Software/Version:             V-System for Windows, Version 3.0
--*****************************************************************
--    Design Specification Elements
--         entire file
--*****************************************************************
LIBRARY ieee;
USE ieee.STD_LOGIC_1164ALL;
ENTITY sn5476 IS
     PORT(j,k,clr,pr,clk: IN UX01Z;
          q,nq: INOUT UX01Z);
```

Figure 2.6.2-1. A Model for the SN5476 J-K Flip-Flop.

```
    END sn5476;

ARCHITECTURE arch_sn5476_dat OF sn5476 IS
    SIGNAL n1,n2,n3,n4,n5,n6,n7,n8,n9:UX01Z;
    BEGIN
--**********************************************************************
 -- Original functional implementation, replaced by behavorial
 --    implementation below.
--**********************************************************************
 -- double inversion required to prevent oscillations in model under certain test conditions
    n9<=  NOT(NOT clk);
    n1<=(clr    AND n2) NOR (j AND n9 AND clr AND n6);
    n2<=(pr    AND n1) NOR (k AND n9 AND pr AND n5);
    n7<=n4    AND clr AND  n2;n8<=n3  AND pr AND n1;
    n5<=  NOT(n6 AND n3 AND    pr);n6<=NOT(n5 AND n4 AND clr);
    n3<=n9    OR NOT  n7;n4<=n9  OR NOT      n8;q<=n5;nq<=n6;
    END arch_sn5476_dat;

ARCHITECTURE arch_sn5476_beh OF sn5476 IS
    SIGNAL ij,ik:UX01Z;
    BEGIN
    PROCESS(j,k,clk,pr,clr,q,nq)
        BEGIN
        IF clr='0' OR pr='0' THEN      ij<=j;ik<=k;
            ELSIF clr='1' AND pr='1' THEN
            IF clk='1' THEN
                IF (j='1' AND nq='1') OR (k='1' AND q='1') THEN      ij<=j;ik<=k;
                    ELSE      ij<=ij;ik<=ik;
                    END IF;
                ELSE      ij<=j;ik<=k;
                END IF;
            ELSE NULL;
            END IF;
        END PROCESS;
    PROCESS(clr,pr,clk)
        BEGIN
        IF clr='0' AND pr='1' THEN      q<='0';nq<='1';
            ELSIF clr='1' AND pr='0' THEN      q<='1';nq<='0';
            ELSIF clr='0' AND pr='0' THEN      q<='1';nq<='1';
            ELSIF clr='1' AND pr='1' THEN
            IF clk'EVENT AND clk='0' THEN
                IF ij='0' AND ik='0' THEN       q<=q;nq<=nq;
                    ELSIF ij='1' AND ik='0' THEN      q<='1';nq<='0';
                    ELSIF ij='0' AND ik='1' THEN      q<='0';nq<='1';
                    ELSIF ij='1' AND ik='1' THEN       q<=nq;nq<=q;
                    END IF;
                END IF;
            END IF;
        END PROCESS;
    END arch_sn5476_beh;
```

Figure 2.6.2-1 continued. A Model for the SN5476.

2.7 Reference Information

2.7.1 Predefined VHDL Data Types

Table 2.7.1-1 contains a list of the predefined VHDL data types. These data types are declared in the package "standard" as defined in the VHDL Language Reference Manual (IEEE-STD-1076).

Type	*Classification*	*Range of Values*	*Delimiter*
Bit	*Enumerated*	*0 or 1*	*apostrophe (')*
Boolean	*Enumerated*	*False or True*	*none*
Character	*Enumerated*	*128 ASCII Character Set*	*apostrophe (')*
Severity_Level	*Enumerated*	*Note, Warning, Error or Failure*	*none*
Integer	*Integer*	*-2147483647 to +2147483647*	*none*
Time	*Physical*	*-2147483647 to +2147483647*	*none*
Real	*Floating Point*	*-1E38 to +1E38*	*none*
String	*Array of Character*	*1 to 2147483647*	*quote (")*
Bit_Vector	*Array of Bit*	*0 to 2147483647*	*quote (")*

Table 2.7.1-1. Predefined VHDL Data Types.

2.7.2 Predefined VHDL Operators

Table 2.7.2-1 contains a list of the predefined VHDL operators.

Operator Type	*Oper*	*Function*	*Operand(s)*	*Result*
logical_operator	*and*	*logical and*	*bit, boolean*	*same type*
	or	*logical or*	*bit, boolean*	*same type*
	nand	*logical nand*	*bit, boolean*	*same type*
	nor	*logical nor*	*bit, boolean*	*same type*
	xor	*logical exclusive or*	*bit, boolean*	*same type*
relational_operator	*=*	*equal to*	*any of same type*	*boolean*
	/=	*not equal to*	*any of same type*	*boolean*
	<	*less than*	*any of same type*	*boolean*
	<=	*less than or equal to*	*any of same type*	*boolean*
	>	*greaterthan*	*any of same type*	*boolean*
	>=	*greater than or equal to*	*any of same type*	*boolean*
adding_operator	*+*	*addition*	*any numeric type*	*same type*
	-	*subtraction*	*any numeric type*	*same type*
	&	*concatenation*	*any one dim array type*	*same type*
sign	*+*	*positive*	*any numeric type*	*same type*
	-	*negative*	*any numeric type*	*same type*
multiplying_operator	***	*multiplication*	*integer, fp, physical*	*same type*
	/	*division*	*integer, fp, physical*	*same type*
	mod	*modulus*	*integer*	*integer*
	rem	*remainder*	*integer*	*integer*
miscellaneous_operator	***	*exponentiation*	*integer, fp*	*same type*
	abs	*absolute value*	*any numeric type*	*same type*
	not	*logicalinverse*	*bit, boolean*	*same type*

Table 2.7.2-1. Predefined VHDL Operators.

2.7.3 Reserved Words

In the sections which follow, the reserved words in VHDL and the standard VHDL packages will be identified. A reserved word may not be used as an identifier or name within VHDL code. It is recommended that reserved words declared by any of the VHDL standard packages not be used in VHDL models, even if a model does not make use of the package in which the word is declared.

2.7.3.1 Table 2.7.3.1-1 contains the reserved words for VHDL. This table is based upon the IEEE-STD-1076 VHDL Language Reference Manual. These words cannot be used as VHDL identifiers.

ABS	ACCESS	AFTER	ALIAS
ALL	AND	ARCHITECTURE	ARRAY
ASSERT	ATTRIBUTE	BEGIN	BLOCK
BODY	BUFFER	BUS	CASE
COMPONENT	CONFIGURATION	CONSTANT	DISCONNECT
DOWNTO	ELSE	ELSIF	END
ENTITY	EXIT	FILE	FOR
FUNCTION	GENERATE	GENERIC	GUARDED
IF	IN	INOUT	IS
LABEL	LIBRARY	LINKAGE	LOOP
MAP	MOD	NAND	NEW
NEXT	NOR	NOT	NULL
OF	ON	OPEN	OR
OTHERS	OUT	PACKAGE	PORT
PROCEDURE	PROCESS	RANGE	RECORD
REGISTER	REM	REPORT	RETURN
SELECT	SEVERITY	SIGNAL	SUBTYPE
THEN	TO	TRANSPORT	TYPE
UNITS	UNTIL	USE	VARIABLE
WAIT	WHEN	WHILE	WITH
XOR			

Table 2.7.3.1-1. VHDL Reserved Words.

2.7.3.2 Table 2.7.3.2-1 contains the reserved words found in the VHDL Standard package. This table is based upon the IEEE-STD-1076 VHDL Language Reference Manual. These words should not be used as VHDL identifiers.

BIT	BIT_VECTOR	BOOLEAN	CHARACTER
ERROR	FAILURE	FALSE	INTEGER
NATURAL	NOTE	NOW	POSITIVE
REAL	SEVERITY_LEVEL	STANDARD	STRING
TIME	TRUE	WARNING	

Table 2.7.3.2-1. VHDL Standard Package Reserved Words.

2.7.3.3 Table 2.7.3.3-1 contains the reserved words declared in the standard TEXTIO package for VHDL. This table is based upon the IEEE-STD-1076 VHDL Language Reference Manual.

ENDFILE	ENDLINE	INPUT	LEFT	LINE	OUTPUT
READ	READLINE	RIGHT	SIDE	TEXT	TEXTIO
WIDTH	WRITE	WRITELINE			

Table 2.7.3.3-1. TEXTIO Package Reserved Words.

2.7.3.4 Table 2.7.3.4-1 contains the reserved words declared under the WAVES standard (IEEE-STD-1029.1. This listing covers all words declared in all of the WAVES standard packages with the exception of the WAVES_SYSTEM package.

ALL_PINS	APPLY	CAPACITIVE
COMPOUND	DELAY	DELAY_TIME
DELAY_TIME_BASIS	DELAY_TIME_LIST	DIRECTION
DIRECTION_LIST	DIRECTION_SUMMARY	DISCONNECTED
DRIVE	ETIME	EVENT
EVENT_LIST	EVENT_TIME	EVENT_TIME_BASIS
EVENT_TIME_LIST	EVENT_VALUE	FILE_SLICE
FILE_SLICE_LIST	FRAME	FRAME_ELIST
FRAME_EVENT	FRAME_LIST	FRAME_SET
FRAME_SET_ARRAY	HIGH	INDEX_SLICE
INDEX_SLICE_LIST	INTEGER_LIST	LOGIC_LIST
LOGIC_MAP	LOGIC_SET	LOGIC_VALUE
LOW	MATCH	MATCH_CONTROL_TYPE
MERGE_ETIME	MERGE_STRING	MIDBAND
NEW_FILE_SLICE	NEW_FRAME_SET	NEW_FRAME_SET_ARRAY
NEW_LOGIC_MAP	NEW_LOGIC_SET	NEW_PINSET
NEW_TIME_DATA	NO_PINS	OBSERVED
PIN_CODE_LIST	PIN_CODE_STRING	PIN_CODES
PIN_DIRECTION	PINSET	PREDICTED
READ_FILE_SLICE	RELEVANCE	REQUIRED
RESISTIVE	RESPONSE	STATE
STIMULUS	STRENGTH	SUPPLY
TAG	TEST_PINS	TEST_PINS_LIST
TIME_DATA	TIME_DATA_LIST	TIME_LIST
TIMED_SLICE	TIMED_SLICE_LIST	UNKNOWN
UNSPECIFIED	VALUE_DICTIONARY	WAVE_TIMING
WAVE_TIMING_LIST	WAVES_INTERFACE	WAVES_MATCH_LIST
WAVES_OBJECTS	WAVES_PORT_LIST	WAVES_STANDARD
WAVES_STD	WAVES_SYSTEM	WTIME_SLICE
WTIME_SLICE_LIST		

Table 2.7.3.4-1. VHDL WAVES Reserved Words.

2.7.3.5 Table 2.7.3.5-1 contains the reserved words declared in the IEEE-STD-1164 multi-value logic package for VHDL.

2.7.3.6 Table 2.7.3.6-1 contains the reserved words identified in the EIA-567 packages. Although these packages are not officially standards, they are recommended for use in the generation of DID compliant models. Hence the reserved words are provided for reference.

FALLING_EDGE	IS_X	RISING_EDGE
STD_LOGIC	STD_LOGIC_1164	STD_LOGIC_VECTOR
STD_ULOGIC	STD_ULOGIC_VECTOR	TO_BIT
TO_BITVECTOR	TO_STDLOGICVECTOR	TO_STDULOGIC
TO_STDULOGICVECTOR	TO_UX01	TO_X01
TO_X01Z	UX01	UX01Z
X01	X01Z	

Table 2.7.3.5-1. IEEE-STD-1164 Reserved Words.

ASYNC	ASYNC_ARRAY
ASYNC_CONSTRAINTS	ASYNCS
CAPACITANCE	CONVERT_TO_UX01Z
CURRENT	CURRENT_VECTOR
DELAY	DELAY_ARRAY
DELAYS	DETECT_EDGE
EDGE_CHECK	EDS
EIA_567	EIA_567_DS
EIA_567_EV	EIA_567_PV
EIA_567_TB	EIA_567_TV
EIA_STRING	ELECTRICAL_PIN_SPEC
ELECTRONIC_DATA_SHEET	ESD_CLASS
ESD_PROTECTION	EV_SIGNAL_LIMIT
EV_SIGNAL_LIMITS	F
GET_TIMING	IMAX
INTERFACE_CONNECTIONS	LOWER_TEMPERATURE_LIMIT
MAX	OPERATING_SELECTION
MAX_ASYNC_CONSTRAINTS_PER_PIN	MAX_DELAY_CONSTRAINTS_PER_PIN
MAX_SYNC_CONSTRAINTS_PER_PIN	MAXIMUM_POWER_DISSIPATION
OPNT	OUTLINE_DRAWING
OUTPUT_DELAYS	PART_NAME
PIN_AND_SIGNAL_CORROLATION	PIN_INDEX
PIN_POINTERS	PIN_RECORD
POINT	POINT_SPECIFICATION
POWER	RESISTANCE
SIGNAL_EDGE	SYNC
SYNC_ARRAY	SYNC_CONSTRAINTS
SYNCS	TEMPERATURE
TMAX	TMIN
TNOM	UPPER_TEMPERATURE_LIMIT
VALID_OPERATING_POINT	VMAX
VMIN	VNOM
VOLTAGE	VOLTAGE_VECTOR

Table 2.7.3.6-1. EIA-567 Reserved Words.

This page intentionally left blank