Chapter 4

4.0 Modeling with EIA-567

4.1 Introduction

4.1.1 Background

In the 1991/2 time frame, Mr. Len Finegold, working on an Air Force contract for the Computer Science Corporation, developed an example DID compliant VHDL model of a Viterbi Decoder. Mr. Finegold based his work upon the EIA-567 Commercial Component Specification. However, the packages and modeling approach he developed have yet to be formally adopted by the EIA.

4.1.2 The Electronic Data Sheet (EDS)

The goal of EIA-567 modeling approach described herein, is to develop an Electronic Data Sheet for a component, sub-assembly, assembly or system using VHDL. Elements modeled in the EDS include input and output pin characteristics (including min/max operating currents, voltages, propagation delays and loads), timing constraints (such as setup and hold times and/or min/max pulse widths), and operating point conditions (such as voltage, power, temperature, package type, radiation hardness, etc.)

4.1.3 The EIA-567 Approach

In the sections which follow, the basic modeling approach as defined by Mr. Len Finegold using his EIA-567 packages will be presented. Under this approach, several standard packages have been defined which when used in conjunction with the appropriate design specification packages provide for the generation of a VHDL model which is compliant with EIA-567 and the VHDL Data Item Description (DI-EGDS-80811). This model will document the input/output pin characteristics and implement the corresponding propagation delay timing at nominal, best and worst case operating conditions. During the course of this discussion, some of the advantages and disadvantages of Mr. Finegold's modeling approach will be discussed in some detail.

4.1.4 Interdependencies

It should be noted that the models presented herein may differ slightly from those originally generated by Mr. Finegold. This is due to the fact that the list of interdependencies (i.e. libraries and/or packages) has been trimmed to only those required by the package or model.

4.1.5 Scope

The modeling approaches described herein are offered as guidelines. These approaches are not dictated for the generation of VHDL models in support of Department of Defense contracts.

4.2 Len Finegold's EIA-567 Approach

4.2.1 Generic Drivers and Checkers

The Electronic Data Sheet (EDS) is developed structurally as shown in Figure 4.2.1-1. The design starts with a core model around which generic drivers and checkers are added to implement timing characteristics and constraints. In support of this design philosophy, three drivers and two checkers have been developed and will be discussed later in this document. Following are the drivers and checkers to be covered.

Component	Component Description
i_driver	Input driver
t_driver	Output driver with tri-state capability
b_driver	Bidirectional driver (code provided is not fully operational)
sync_checker	Synchronous constraint checker (setup and hold times)
async_checker	Asynchronous constraint checker (minimum and maximum pulse widths)

4.2.2 The EIA-567 Packages

In order to support the generic drivers and checkers in the development of the EDS model, there are four standard packages which are analyzed in conjunction with three design packages. The four EIA-567 packages and the three design packages are listed below in the order of analysis. For the design packages, the name "design" has been substituted for the design name. Each of these packages will be discussed in some detail within the following sections.

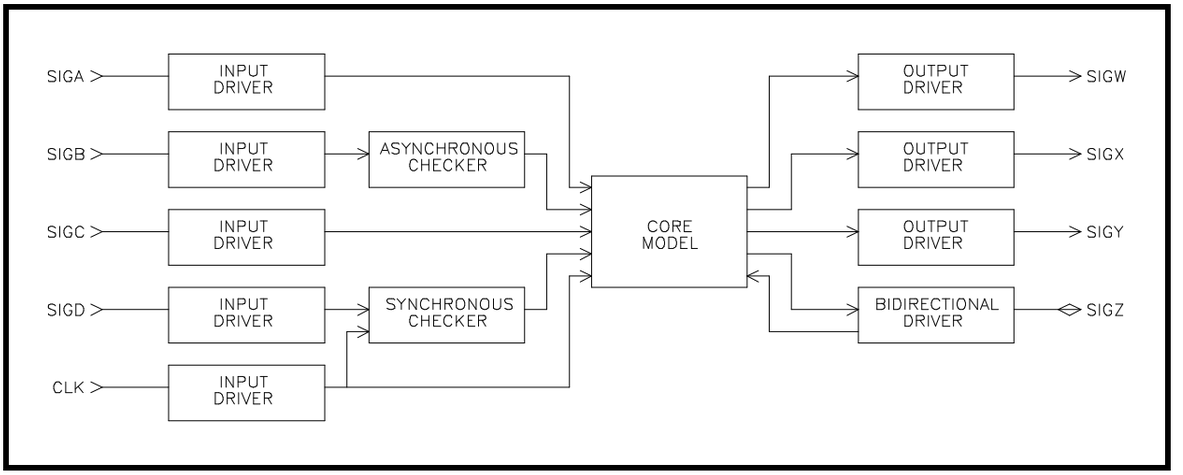

Figure 4.2.1-1. The Electronic Data Sheet Model.

Package	Package Description	Comments
EIA_567_EV	Electrical View Package	Establishes the data types to define pin characteristics
design_EV	Electrical View Package	Defines the pin characteristics for the design
EIA_567_TV	Timing View Package	Establishes the data types to define pin timing
design_TV	Timing View Package	Defines the pin timing for the design
EIA_567_PV	Physical View Package	Establishes the data types which define the electronic data sheet
design_PV	Physical View Package	Defines the electronic data sheet
EIA_567_TB	Toolbox Package	Defines functions used in support of the electronic data sheet

4.2.3 The Electronic Data Sheet

With these elements in hand, the entire EDS model is assembled. All model timing is provided through the input and output drivers. The input drivers may be passed a wiring delay generic timing delay while the output drivers actually model the propagation delays as specified in the "design_TV" package and indexed through the "design_PV" package. In a similar fashion, the synchronous and asynchronous constraints are defined in the "design_TV" package and indexed through the "design_PV" package

4.2.4 Pin Characteristics

Pin characteristics are specified in the "design_EV" package and are loosely indexed through the "design_PV" package. The pin characteristics are carried along with the model as reference information, but they have no functional purpose as far as the model is concerned.

4.3 A Core Model

4.3.1 A 12 Bit Population Counter

In order to develop the EIA-567 modeling approach, we will work with an example design for a 12-bit population counter. This model features 12 inputs. At any point in time, the number of logic ones on these inputs will be summed in order to obtain a 4-bit output between 0 (0000 binary) and 12 (1100 binary) inclusive. The core model to be used is listed in Figure 4.3.1-1. This model is based upon the FBG Standard Electronic Module (SEM) as specified in MIL-M-28787/175, hence the name "fbg_core".

4.3.2 STD_LOGIC_1164

This model is based upon a behavioral architecture. It employs the STD_LOGIC type of the IEEE-STD-1164 multi-value logic system, hence it's dependency on the STD_LOGIC_1164 package. This model is technology independent. This provides the design engineer with the flexibility to evaluate implementation options.

4.3.3 The Uninitialized State

```
--*****************************************************************
--    (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-28787/175
--         Project:            SHARP TIREP
--    DESC Certification
--         Status:             TBD
--*****************************************************************
--    Revision History
--         Version:            1.0
--              Date:          18 May 1994
--              Comments:      Original Release
--*****************************************************************
--    Module Description
--         File:               fbg_cor.vhd
--         Module Name(s):     fbg_coreentity/arch_fbg_beharchitecture
--         Constraints:        none
--         Limitations:        none
--         I/O Format(s):      std_logic
--         Purpose and Use:    This VHDL module contains the behavorial description
--              for the FBG standard hardware module.        This is also the description
--              for the semi-custom microcircuit which is used on this module.
--         Notes:              none
--*****************************************************************
--    Standard Libraries/Packages
--         std_logic_1164      standard multi-value logic package
--    Associated Packages (order of analysis implied)
--         none
--    Component Models
--         none
--    Platform:                486/33MHz PC
--    Software/Version:        V-System for Windows, Version 3.0
--*****************************************************************
--    Design Specification Elements
--         entire file
--*****************************************************************
LIBRARY ieee;
USE ieee.STD_LOGIC_1164ALL;
ENTITY fbg_core IS
    PORT(a0,a1,a2,a3,a4,a5,b0,b1,b2,b3,b4,b5: IN STD_LOGIC;
         f0,f1,f2,f3: OUT STD_LOGIC);
```

Figure 4.3.1-1. A 12-Bit Population Counter.

```
    END fbg_core;
ARCHITECTURE arch_fbg_beh OF fbg_core IS
    FUNCTION to_vector( INPUT,num_bits: INTEGER) RETURN STD_LOGIC_VECTOR IS
        VARIABLE result:STD_LOGIC_VECTOR(num_bits-1 DOWNTO 0);
        VARIABLE weight:INTEGER;
        VARIABLE temp:INTEGER;
        BEGIN
        weight:=2**(num_bits-1);
        temp:= INPUT;
        FOR i IN result'HIGH DOWNTO result'LOW LOOP
            IF temp>=weight THEN
                result(i):='1';
                temp:=temp-weight;
                ELSE result(i) := '0';
                END IF;
            weight:=weight/2;
            END LOOP;
        RETURN result;
        END to_vector;
    SIGNAL din:STD_LOGIC_VECTOR(11 DOWNTO 0);
    SIGNAL dout:STD_LOGIC_VECTOR(3 DOWNTO 0);
    BEGIN
-- All inputs and outputs are single lines (not bussed).
-- Busses are used internally, however.
    din(0)<=a0;din(1)<=a1;din(2)<=a2;din(3)<=a3;din(4)<=a4;din(5)<=a5;
    din(6)<=b0;din(7)<=b1;din(8)<=b2;din(9)<=b3;din(10)<=b4;din(11)<=b5;
    f0<=dout(0);f1<=dout(1);f2<=dout(2);f3<=dout(3);
    PROCESS(din)
        VARIABLE sum:INTEGER RANGE -1 TO 12;
        BEGIN
-- Initialize the sum.
        sum:=0;
-- Count the number of inputs in a logic '1' state.
        FOR i IN 0 TO 11 LOOP
            IF din(i)='1' THEN sum:=sum+1;
                ELSIF din(i)='0' THEN NULL;
-- If an input is undefined, the return -1 in sum.
                ELSE sum:=-1;EXIT;
                END IF;
            END LOOP;
-- Generate output based upon value of sum.
        IF sum=-1 THEN dout<="xxxx";
            ELSE dout<=to_vector(sum,4);
            END IF;
        END PROCESS;
    END arch_fbg_beh;
```

Figure 4.3.1-1 continued. A 12-Bit Population Counter.

All outputs of the model, when simulated, should start in the uninitialized or 'U' state. This is the only time that the model may generate the 'U' state. The occurrence of an event which causes an output in the 'U' state to be reevaluated shall result in the transition of that output to one of the four remaining states (i.e. 'X', '0', '1', 'Z') defined for this model. **Note:** This is a requirement imposed by EIA-567, paragraph 4.4.2.1.3, it is not a requirement under DI-EGDS-80811.

4.3.4 Power

As currently written, EIA-567 and DI-EGDS-80811 do not explicitly address the use of power and ground pins in the model. For the purposes of the TIREP modeling approach, the following guidelines apply.

4.3.4.1 To support logic synthesis, power pins are assumed and should not be modeled in the synthesizable core.

4.3.4.2 To support assembly level models, these pins must be identified as network elements.

4.3.4.3 The requirements for the implementation of power and ground in testbenches has yet to be defined, however knowledge of these functions must be transferred to the test system in some fashion.

4.3.4.4 In the example provided, the functions for Vcc and Gnd are not implemented in the core model. This is done in order to preserve the integrity of the core model for synthesis. The power and ground functions will be picked up in the EDS model as they are required at that level.

4.3.5 Busses or Vectors

It may be noted that this model does not contain any busses in the port statement, but that the inputs and outputs are translated to and from busses within the model architecture. EIA-567, paragraph 4.3.1, requirement 2 prohibits the use of busses at the ports. DI-EGDS-80811 does not impose this limitation.

4.4 The EIA-567 Electrical View Package

4.4.1 Subprogram and Data Type Declarations

In order to develop the EIA-567 modeling approach, we start with the electrical view package. This package is used to define some data types which will be used by the model to describe characteristics of the device/circuit being modeled. Table 4.4.1-1 contains a list of the functions and data types which are defined for this package. Refer to Figure 4.4.1-1 for the code listing of this package.

Declaration	Data Type
RESISTANCE	physical type
CAPACITANCE	physical type
TEMPERATURE	physical type
VOLTAGE	physical type
CURRENT	physical type
POWER	physical type
OPERATING_SELECTION	enumerated type
EV_SIGNAL_LIMIT	record type
EV_SIGNAL_LIMITS	array of ev_signal_limit type
Declarations	Subprogram Type
CONVERT_TO_UX01Z	function (6)

Table 4.4.1-1. EIA_567_EV Package Declarations.

4.4.1.1 This package establishes the physical types of VOLTAGE, CURRENT, POWER, CAPACITANCE, RESISTANCE and TEMPERATURE. In the original version of this package reviewed, this package imposed a range of 0 to 1000 on the capacitance data type. In the listing provided, this range has been extended to 1e8.

```
-------------------------------------------------------------------------
-- eia 567 electrical view standard package
-------------------------------------------------------------------------
-- brief description:          This is the eia 567 package. The contents of
--                             this package should not be changed.
--                             Implementation of the package body may be changed
--                             as long as the specific capabilities are not
--                             changed.
-- version                     1.1
--history:                     broke the original eia_567 package into four smaller
--                             packages (len finegold 12/22/92)
-- annotations:                none
-- analysis dependencies:      std_logic_1164 (ieee standard)
-- limitations:                none
-- supplementary information:  none
-- development platform:       model technologies v-system for ibm pc running
--                             on a softpc emulation of dos on a mac iici and
--                             clsi vhdl learning kit for sun workstations,
--                             synopsys, vantage, intermetrics/valid
-- vhdl software version:      model technologies v-system version 1.3
--                             clsi vhdl learning kit version 1.0
-------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.STD_LOGIC_1164.ALL;
PACKAGE EIA_567_EV IS
-------------------------------------------------------------------------
-- physical types
-------------------------------------------------------------------------
    TYPE RESISTANCE IS RANGE 1 TO 1e8
        UNITS
        ohms;
        kilohms=1000 ohms;
        megohms=1000 kilohms;
        END UNITS;
-- Note: As defined in the example, capacitance has a range to 1000 only.
    TYPE CAPACITANCE IS RANGE 0 TO 1e8
        UNITS
        pf;
        nf=1000 pf;
        uf=1000 nf;
        END UNITS;
    TYPE TEMPERATURE IS RANGE -100 TO 300
        UNITS
        degrees_c;
        END UNITS;
    TYPE VOLTAGE IS RANGE -1e8 TO 1e8
        UNITS
        uv;
        mv=100 uv;
```

Figure 4.4.1-1. The EIA_567_EV Package.

```
        v=1000  mv;
        END UNITS;
    TYPE CURRENT IS RANGE -1e8 TO 1e8
        UNITS
        ua;
        ma=100   ua;
        a=1000   ma;
        END UNITS;
    TYPE POWER IS RANGE -1e8 TO 1e8
        UNITS
        uw;
        mw=100  uw;
        W=1000  mw;
        END UNITS;
-------------------------------------------------------------------------
-- type conversions
-------------------------------------------------------------------------
    FUNCTION CONVERT_TO_UX01Z(s:STD_LOGIC_VECTOR)
        RETURN STD_LOGIC_VECTOR;
    FUNCTION CONVERT_TO_UX01Z(s:STD_ULOGIC_VECTOR)
        RETURN STD_ULOGIC_VECTOR;
    FUNCTION CONVERT_TO_UX01Z(s:STD_ULOGIC) RETURN UX01Z;
    FUNCTION CONVERT_TO_UX01Z(b:BIT_VECTOR)
        RETURN STD_LOGIC_VECTOR;
    FUNCTION CONVERT_TO_UX01Z(b:BIT_VECTOR)
        RETURN STD_ULOGIC_VECTOR;
    FUNCTION CONVERT_TO_UX01Z(b:BIT) RETURN UX01Z;
-------------------------------------------------------------------------
-- electrical view related declarations
--    the following type definition is used to enumerate the three basic
--    operating selections which define the performance envelope of the
--    device.   tmin is the fastest selection bounding the quick portion of
--    the operating envelope.      tnom is the selection which lies somewhere
--    between tmin and tmax.     tmax is the slowest selection bounding the
--    slow portion of the operating envelope.
-------------------------------------------------------------------------
    TYPE OPERATING_SELECTION IS (TMIN,TNOM,TMAX);
-------------------------------------------------------------------------
-- individual electrical characteristics for a pin which defines the
-- limit of it's operation
-------------------------------------------------------------------------
    TYPE EV_SIGNAL_LIMIT IS RECORD
        voh,vol:VOLTAGE;
        ioh,iol:CURRENT;
        test_load: NATURAL;
        vih,vil:VOLTAGE;
        iih,iil:CURRENT;
        pin_load: CAPACITANCE;
        END RECORD;
```

Figure 4.4.1-1 continued. The EIA_567_EV Package.

```
------------------------------------------------------------------------
-- a collection of the individual electrical limits
------------------------------------------------------------------------
    TYPE EV_SIGNAL_LIMITS IS ARRAY (NATURAL RANGE <>)
        OF EV_SIGNAL_LIMIT;
    END EIA_567_EV;

PACKAGE BODY EIA_567_EV IS
------------------------------------------------------------------------
-- table name: cvt_to_ux01z
--parameters:
--    in: std_ulogic
--    returns: ux01z
--    purpose: to convert state-strength to state only
--    example: if (cvt_to_ux01z(input_signal)='1') then ...
------------------------------------------------------------------------
    TYPE logic_ux01z_table IS
        ARRAY (STD_ULOGIC'LOW TO STD_ULOGIC'HIGH) OF UX01Z;
    CONSTANT cvt_to_ux01z: logic_ux01z_table:=
        ('U','X','0','1','Z','X','0','1','X');
    FUNCTION CONVERT_TO_UX01Z (s: STD_LOGIC_VECTOR)
        RETURN STD_LOGIC_VECTOR IS
        VARIABLE sv: STD_LOGIC_VECTOR (1 TO s'length):=s;
        VARIABLE result: STD_LOGIC_VECTOR (1 TO s'length):=(OTHERS=>'X');
        BEGIN
        FOR i IN result'RANGE LOOP
            result(i):=cvt_to_ux01z(sv(i));
            END LOOP;
        RETURN result;
        END CONVERT_TO_UX01Z;
    FUNCTION CONVERT_TO_UX01Z (s: STD_ULOGIC_VECTOR)
        RETURN STD_ULOGIC_VECTOR IS
        VARIABLE sv: STD_ULOGIC_VECTOR (1 TO s'length):=s;
        VARIABLE result: STD_ULOGIC_VECTOR (1 TO s'length):=(OTHERS=>'X');
        BEGIN
        FOR i IN result'RANGE LOOP
            result(i):=cvt_to_ux01z(sv(i));
            END LOOP;
        RETURN result;
        END CONVERT_TO_UX01Z;
    FUNCTION CONVERT_TO_UX01Z (s: STD_ULOGIC) RETURN UX01Z IS
        BEGIN
        RETURN (cvt_to_ux01z(s));
        END CONVERT_TO_UX01Z;
    FUNCTION CONVERT_TO_UX01Z (b: BIT_VECTOR)
        RETURN STD_LOGIC_VECTOR IS
        VARIABLE bv: BIT_VECTOR (1 TO b'length):=b;
        VARIABLE result: STD_LOGIC_VECTOR (1 TO b'length);
        BEGIN
```

Figure 4.4.1-1 continued. The EIA_567_EV Package.

```
        FOR i IN result'RANGE LOOP
            CASE bv(i) IS
                WHEN    '0'=>result(i):='0';
                WHEN    '1'=>result(i):='1';
                END CASE;
            END LOOP;
        RETURN result;
        END CONVERT_TO_UX01Z;
    FUNCTION CONVERT_TO_UX01Z(b:BIT_VECTOR)
        RETURN STD_ULOGIC_VECTOR IS
        VARIABLE bv:BIT_VECTOR(1 TO  b'length):=b;
        VARIABLE result:STD_ULOGIC_VECTOR(1 TO b'length);
        BEGIN
        FOR i IN result'RANGE LOOP
            CASE bv(i) IS
                WHEN    '0'=>result(i):='0';
                WHEN    '1'=>result(i):='1';
                END CASE;
            END LOOP;
        RETURN result;
        END CONVERT_TO_UX01Z;
    FUNCTION CONVERT_TO_UX01Z(b:BIT) RETURN UX01Z IS
        BEGIN
        CASE b IS
            WHEN  '0'=>RETURN('0');
            WHEN  '1'=>RETURN('1');
            END CASE;
        END CONVERT_TO_UX01Z;
    END EIA_567_EV;
```

Figure 4.4.1-1 continued. The EIA_567_EV Package.

4.4.1.2 The data type OPERATING_SELECTION is basically used to select either the minimum, maximum or nominal propagation delays during a simulation. In accordance with DI-EGDS-80811, paragraph 10.2.3.2 and EIA-567, paragraph 3.4.1, the best, worst and nominal propagation delays shall be included in the model.

4.4.1.3 The EV_SIGNAL_LIMIT record is established to define the interface characteristics of the model ports. The record includes input and output parameters since the port may be an input, output or bidirectional. If the port is an input, then the output parameters (vol, voh, iol, ioh, test_load) are set to zero (0). In a similar fashion, if the port is an output, then the input parameters (vil, vih, iil, iih, pin_load) are set to zero (0). The EV_SIGNAL_LIMIT record type declaration is shown below. The EV_SIGNAL_LIMITS data type is then defined as an array of EV_SIGNAL_LIMIT records.

```
TYPE EV_SIGNAL_LIMIT IS RECORD
    voh,vol:VOLTAGE;
    ioh,iol:CURRENT;
    test_load: NATURAL;
    vih,vil:VOLTAGE;
    iih,iil:CURRENT;
    pin_load: CAPACITANCE;
    END RECORD;
```

4.4.1.3.1 It may be noted from the declaration that the "pin_load" is a capacitance. This is the input capacitance of the pin.

4.4.1.3.2 The "test_load" is an integer and is used to specify the signal load condition for timing measurements. The "test_load" shall correspond to the manufacturer's specified test circuit. Use of the "test_load" requires that each load circuit used be explicitly described for the corresponding "test_load" value.

4.4.1.4 Also included in this package are several conversion functions which deal with the UX01Z data type. This appears to be a favored data type for use with EIA-567, although the use of this data type is not imposed by either EIA-567 or DI-EGDS-80811. The function declarations are listed below.

```
FUNCTION CONVERT_TO_UX01Z(s:STD_LOGIC_VECTOR) RETURN STD_LOGIC_VECTOR;
FUNCTION CONVERT_TO_UX01Z(s:STD_ULOGIC_VECTOR)
    RETURN STD_ULOGIC_VECTOR;
FUNCTION CONVERT_TO_UX01Z(s:STD_ULOGIC) RETURN UX01Z;
FUNCTION CONVERT_TO_UX01Z(b:BIT_VECTOR) RETURN STD_LOGIC_VECTOR;
FUNCTION CONVERT_TO_UX01Z(b:BIT_VECTOR) RETURN STD_ULOGIC_VECTOR;
FUNCTION CONVERT_TO_UX01Z(b:BIT) RETURN UX01Z;
```

4.4.2 Interdependencies

Following are the interdependencies for this package. As noted, this package is dependent only on the IEEE-STD-1164 multi-value logic package.

Interdependencies
STD_LOGIC_1164

4.5 The FBG Electrical View Package

4.5.1 Data Type Declarations

The design electrical view package for the FBG module is contained in Figure 4.5.1-1. This package defines the following constants as they pertain to this module.

Declaration	Data Type
VOLTAGE_VECTOR	array type
CURRENT _VECTOR	array type
ELECTRICAL_PIN_SPEC	constant array type
VMIN, VMAX, VNOM	constant voltage_vector types
IMAX	constant current_vector type
LOWER_TEMPERATURE_LIMIT	constant temperature type
UPPER_TEMPERATURE_LIMIT	constant temperature type

Table 4.5.1-1. Design_EV Package Declarations.

4.5.1.1 The VOLTAGE_VECTOR array is set up for the number of power supplies required by the model. In most cases, only one supply is required and therefore, only 0th array element is needed. The CURRENT_VECTOR should be dimensioned to the same size as the VOLTAGE_VECTOR. The CURRENT_VECTOR will provide a current specification for each voltage (or power supply) identified.

4.5.1.2 In this package the key feature is that the characteristics of the pins are defined as classes in the ELECTRICAL_PIN_SPEC constant. This package contains minimum and maximum interface parameters as defined in the specification. The nominal parameters have been entered as the average of the minimum and maximum values. Pin class 0 is always defined as the no-load case and all elements of this class are set to zero (0). This pin class is used for any pins for which parameter data is not specified. Pin class 0 shall be used for power supply pins, when applicable. The following elements are declared in each class of this constant.

```
--*****************************************************************
--    (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-28787/175
--         Project:                 SHARP   TIREP
--    DESC Certification
--         Status:                  TBD
--*****************************************************************
--    Revision History
--         Version:                 1.0
--              Date:               18 May 1994
--              Comments:           Original Release
--*****************************************************************
--    Module Description
--         File:                    fbg_ev_p.vhd
--         Module  Name(s):         fbg_evpackage
--         Constraints:             none
--         Limitations:             none
--         I/O  Format(s):          none
--         Purpose and Use:         This VHDL package defines the electrical pin
--              specifications and the operating point limits for the FBG
--              standard hardware module.
--         Notes:                   none
--*****************************************************************
--    Standard Libraries/Packages
--         none
--    Associated Packages (order of analysis implied)
--         eia_567_ev               standard electrical view package
--    Component  Models
--         none
--    Platform:                     486/33MHz PC
--    Software/Version:             V-System for Windows, Version 3.0
--*****************************************************************
--    Design Specification Elements
--         electrical pin spec           constant declaration
--         vmax,  vmin,  vnom            constant declaration
--         imax                          constant declaration
--         lower_temperature_limit       constant declaration
--         upper_temperature_limit       constant declaration
--*****************************************************************
USE work.EIA_567_EV.ALL;
```

Figure 4.5.1-1. The FBG_EV Package.

```
PACKAGE fbg_ev IS
-------------------------------------------------------------------
-- Declaration of number of voltage supplies included
-------------------------------------------------------------------
    TYPE VOLTAGE_VECTOR IS ARRAY (0 DOWNTO 0) OF VOLTAGE;
    TYPE CURRENT_VECTOR IS ARRAY (0 DOWNTO 0) OF CURRENT;
-------------------------------------------------------------------
-- Declaration of constants which will be detailed in the package body
-------------------------------------------------------------------
    CONSTANT ELECTRICAL_PIN_SPEC : EV_SIGNAL_LIMITS;
    CONSTANT VMAX : VOLTAGE_VECTOR;
    CONSTANT VMIN : VOLTAGE_VECTOR;
    CONSTANT VNOM : VOLTAGE_VECTOR;
    CONSTANT IMAX : CURRENT_VECTOR;
    CONSTANT LOWER_TEMPERATURE_LIMIT : TEMPERATURE;
    CONSTANT UPPER_TEMPERATURE_LIMIT : TEMPERATURE;
    END fbg_ev;

PACKAGE BODY fbg_ev IS
-------------------------------------------------------------------
-- Signal specifications
--*****************************************************************
-- For the design, identify the number of pin classes required. A
-- pin class is defined as an input or an output whose interface
-- characteristics are different from other inputs and outputs. As
-- a minimum, there should be three pin classes: default (class 0),
-- inputs and outputs. Pin classes for nominal (or typical), minimum
-- and/or maximum characteristics should be included when possible.
--*****************************************************************
-- Load circuits
--    The standard load circuit for this model consists of two resistors (r1 and r2),
--    a capacitor (cl), a diode (cr1) and a power supply (p01).  The power supply is
--    connected to one side of r2.     The remaining side of r2 is connected to the
--    annode of cr1.     The cathode of cr1 is connected to the output of the FBG module.
--    Also from the FBG module output, the resistor r1 and capacitor cl are connected
--    in parallel, to ground. Following are the values which apply to this load circuit for
--    each class     specified.     For class 0 and all input classes, load components are not
--    used  (N/U).
-------------------------------------------------------------------
--    class     p01  (1%)     cr1       r1 (1%)       r2 (1%)                    cl
--    1         2.5V          1N916     6.04kohm      93.75ohm      N/U
-------------------------------------------------------------------
    CONSTANT ELECTRICAL_PIN_SPEC : EV_SIGNAL_LIMITS := (
-- pin class #0 - no load
          (voh=>0.0     v,ioh=>0.0     ua,vol=>0.0     v,iol=>0.0     ua,test_load=>0,
          vih=>0.0     v,iih=>0.0     ua,vil=>0.0     v,iil=>0.0     ua,pin_load=>0     pf),
-- pin class #1 - inputs, typical
          (voh=>0.0     v,ioh=>0.0     ua,vol=>0.0     v,iol=>0.0     ua,test_load=>0,
          vih=>2.0     v,iih=>20     ua,vil=>0.8     v,iil=>-850     ua,pin_load=>10     pf),
```

Figure 4.5.1-1 continued. The FBG_EV Package.

```
-- pin class #2 - inputs, minimum
        (voh=>0.0   v,ioh=>0.0   ua,vol=>0.0   v,iol=>0.0   ua,test_load=>0,
        vih=>2.0   v,iih=>0   ua,vil=>0.8   v,iil=>-50   ua,pin_load=>2   pf),
-- pin class #3 - inputs a3, a4 maximum
        (voh=>0.0   v,ioh=>0.0   ua,vol=>0.0   v,iol=>0.0   ua,test_load=>0,
        vih=>2.0   v,iih=>40   ua,vil=>0.8   v,iil=>-1.73   ma,pin_load=>15   pf),
-- pin class #4 - inputs a0, a1, a2, b0, b1, b2, b3, b4, b5 maximum
        (voh=>0.0   v,ioh=>0.0   ua,vol=>0.0   v,iol=>0.0   ua,test_load=>0,
        vih=>2.0   v,iih=>160   ua,vil=>0.8   v,iil=>-6.9   ma,pin_load=>15   pf),
-- pin class #5 - input a5 maximum
        (voh=>0.0   v,ioh=>0.0   ua,vol=>0.0   v,iol=>0.0   ua,test_load=>0,
        vih=>2.0   v,iih=>320   ua,vil=>0.8   v,iil=>-13.8   ma,pin_load=>15   pf),
-- pin class #6 - outputs, maximum
        (voh=>2.4   v,ioh=>-400   ua,vol=>0.4   v,iol=>16.0   ma,test_load=>1,
        vih=>0.0   v,iih=>0.0   ua,vil=>0.0   v,iil=>0.0   ua,pin_load=>0   pf));
--*****************************************************************
--For the "vmax", "vmin", "vnom", "imax", "lower_temperature_limit",
-- and "upper_temperature_limit", enter the appropriate secifications
-- for the design.
--*****************************************************************
-----------------------------------------------------------------
-- Power limits
-----------------------------------------------------------------
    CONSTANT VMAX: VOLTAGE_VECTOR:=(0=>5.5   v);
    CONSTANT VMIN: VOLTAGE_VECTOR:=(0=>4.5   v);
    CONSTANT VNOM: VOLTAGE_VECTOR:=(0=>5.0   v);
    CONSTANT IMAX: CURRENT_VECTOR:=(0=>340   ma);
-----------------------------------------------------------------
-- Temperature limits
-----------------------------------------------------------------
    CONSTANT LOWER_TEMPERATURE_LIMIT: TEMPERATURE:=0   degrees_c;
    CONSTANT UPPER_TEMPERATURE_LIMIT: TEMPERATURE:=60   degrees_c;
    END fbg_ev;
```

Figure 4.5.1-1 continued. The FBG_EV Package.

Element	Parameter
vil	maximum input low voltage
vih	minimum input high voltage
vol	maximum output low voltage
voh	minimum output high voltage
iil	maximum input low current
iih	maximum input high current
iol	minimum output low current
ioh	minimum output high current
test_load	class load applied to an output at which the output parameters are measured
pin_load	capacitive load characteristic of the input pin being modeled

4.5.1.2.1 The "test_load" is an integer value which references the load classes applied to this device. For each load class referenced, an explicit description of the load circuit must be provided in the package. This description must include sufficient detail to allow the load circuit to be replicated without confusion or ambiguity. Once again, class 0 is the no-load case. There shall be no load circuits or elements defined for this class.

4.5.1.3 In addition to the pin specifications and the vector declarations, the operating conditions of the model are also defined. This includes the nominal, minimum and maximum operating voltages (VNOM, VMIN and VMAX), the maximum operating current (IMAX) and the operating temperature range (LOWER_TEMPERATURE_LIMIT to UPPER_TEMPERATURE_LIMIT).

4.5.1.3.1 It should be noted that when a single element array (or vector) is employed, such as the current or voltage vectors in this example, it may be necessary to map the vector element to the value as shown. This is not a requirement of VHDL since the array has only one element and one element is assigned. Nonetheless, some VHDL tools may require explicit mapping in this case.

4.5.2 Requirements

This package provides information to meet some of the requirements of DI-EGDS-80811, paragraphs 10.2.2.2 and 10.2.2.3. In EIA-567, this package primarily addresses the requirements of paragraph 3.3.1, subparagraph 4. It should be noted that these documents do not require minimum, maximum and nominal operating voltages. They only require that the operating conditions for the model be specified and leave it up to the design engineer to interpret the requirement as it pertains to a specific application.

4.5.3 Limitations

With the current VHDL tools, the information provided by this package is carried along with the model as reference information only. The information provided by this package does not provide anything of operational significance to the simulation of the completed model other than to impose operational limits on the selected operating point for the design.

4.5.4 Interdependencies

The interdependencies for this package as follows. As noted, the only interdependency is the EIA-567 standard electrical view package.

Interdependencies
EIA_567_EV

4.6 The EIA-567 Timing View Package

4.6.1 Functions and Data Type Declarations

Continuing with the EIA-567 modeling approach, we will now take a look at the timing view package. This package is used to define some data types which will be used by the design entity to describe the model timing parameters such as propagation delay, setup and hold times, minimum and maximum pulse widths, etc. Table 4.6.1-1 contains a list of the functions and data types which are defined for this package. Refer to Figure 4.6.1-1 for the code listing of this package.

4.6.1.1 The POINT data type is used to establish the operating point for the design entity. This point includes the operating voltage(s), the operating temperature and the appropriate propagation delay selection. Following is the declaration for this record type.

```
TYPE POINT IS RECORD
    selection: OPERATING_SELECTION;
    temp: TEMPERATURE;
    SUPPLY: VOLTAGE_VECTOR;
    END RECORD;
```

Declaration	Data Type
POINT	record type
DELAY	record type
OUTPUT_DELAYS	array of delay type
ASYNC	record type
ASYNC_CONSTRAINTS	array of async type
SIGNAL_EDGE	enumerated type
SYNC	record type
SYNC_CONSTRAINTS	array of sync type

Declaration	Subprogram Type
VALID_OPERATING_POINT	function
DETECT_EDGE	function
"="	function
">="	function
"<="	function
">"	function
"<"	function

Table 4.6.1-1. EIA_567_TV Package Declarations.

4.6.1.2 The DELAY, ASYNC and SYNC record types are defined to provide the user with a mechanism through which model timing parameters can be established. The OUTPUT_DELAYS array will be used for propagation delays for transitions between the high (or '1'), the low (or '0') and the high impedance (or 'Z') logic states. The ASYNC_CONSTRAINTS array will be used to capture minimum and maximum pulse widths while the SYNC_CONSTRAINTS array will be used for setup and hold times. Since setup and hold times must be referenced to a signal transition, the type SIGNAL_EDGE was generated which enumerates all of the possible signal transitions for IEEE-STD-1164 std_logic.

4.6.1.2.1 Following is the declaration for the DELAY record type. Each entry in this record defined by the character "t" followed in order by the "from" state and the "to" state. This record is used to enumerate the propagation delays between the '0' (or 'L'), '1' (or 'H') and 'Z' states.

```
TYPE DELAY IS RECORD
    tlh,thl,tlz,thz,tzl,tzh: TIME;
    END RECORD;
```

4.6.1.2.2 Following is the declaration for the ASYNC record type. This record is used to identify the minimum and maximum pulse widths for a signal. The pulse widths specified can be for either high (1), low (0) or both states of a signal (see Figure 4.6.1.2.2-1).

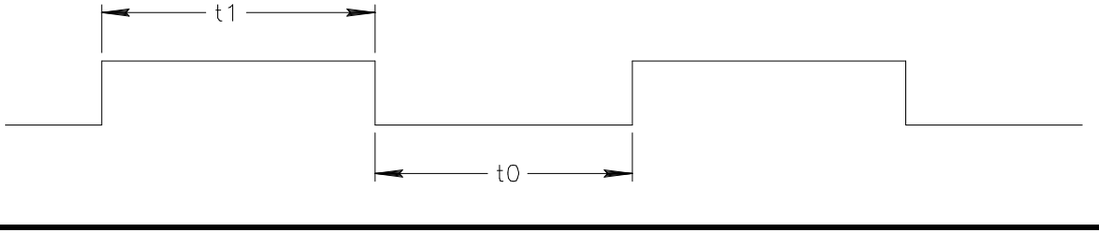

Figure 4.6.1.2.2-1. Definition of T0 and T1.

```
-----------------------------------------------------------------------
-- eia 567 standard package
-----------------------------------------------------------------------
-- brief description:          This is the eia 567 package.  The contents of
--                             this package should not be changed.
--                             Implementation of the package body may be changed
--                             as long as the specific capabilities are not
--                             changed.
-- version                     1.1
--history:                     broke the original eia_567 package into four smaller
--                             packages (len finegold 12/22/92)
-- annotations:                none
-- analysis dependencies:      eia_567_ev
--                             design specific electrical view package
-- limitations:                none
-- supplementary information:  none
-- development platform:       model technologies v-system for ibm pc running
--                             on a softpc emulation of dos on a mac iici and
--                             clsi vhdl learning kit for sun workstations,
--                             synopsys, vantage, intermetrics/valid
-- vhdl software version:      model technologies v-system version 1.3
--                             clsi vhdl learning kit version 1.0
-----------------------------------------------------------------------
LIBRARY ieee;
USE ieee.STD_LOGIC_1164.ALL;
USE work.EIA_567_EV.ALL;
USE work.fbg_ev.ALL;
PACKAGE EIA_567_TV IS
-----------------------------------------------------------------------
-- timing related declarations
--    declaration of an operating point type based on the number of
--    voltage supplies defined by the user
-----------------------------------------------------------------------
    TYPE POINT IS RECORD
        selection: OPERATING_SELECTION;
        temp: TEMPERATURE;
        SUPPLY: VOLTAGE_VECTOR;
        END RECORD;
-----------------------------------------------------------------------
-- declaration of the delay transitions as a record of information
-----------------------------------------------------------------------
    TYPE DELAY IS RECORD
        tlh,thl,tlz,thz,tzl,tzh: TIME;
        END RECORD;
-----------------------------------------------------------------------
-- a collection of individual delays
-----------------------------------------------------------------------
    TYPE OUTPUT_DELAYS IS ARRAY (NATURAL RANGE <>) OF DELAY;
-----------------------------------------------------------------------
```

Figure 4.6.1-1. The EIA_567_TV Package.

```
-- declaration of the asynchronous constraint record of information
--------------------------------------------------------------------
    TYPE ASYNC IS RECORD
        t1min,t0min,t1max,t0max: TIME;
        END RECORD;
--------------------------------------------------------------------
-- a collection of individual asynchronous constraints
--------------------------------------------------------------------
    TYPE ASYNC_CONSTRAINTS IS ARRAY (NATURAL RANGE <>) OF ASYNC;
--------------------------------------------------------------------
-- a definition of all possible edge transitions
--------------------------------------------------------------------
    TYPE SIGNAL_EDGE IS     (euu,eux,eu0,eu1,euz,euw,eul,euh,eud,
                             exu,exx,ex0,ex1,exz,exw,exl,exh,exd,
                             e0u,e0x,e00,e01,e0z,e0w,e0l,e0h,e0d,
                             e1u,e1x,e10,e11,e1z,e1w,e1l,e1h,e1d,
                             ezu,ezx,ez0,ez1,ezz,ezw,ezl,ezh,ezd,
                             ewu,ewx,ew0,ew1,ewz,eww,ewl,ewh,ewd,
                             elu,elx,el0,el1,elz,elw,ell,elh,eld,
                             ehu,ehx,eh0,eh1,ehz,ehw,ehl,ehh,ehd,
                             edu,edx,ed0,ed1,edz,edw,edl,edh,edd);
--------------------------------------------------------------------
-- declaration of the synchronous constraint record of information
--------------------------------------------------------------------
    TYPE SYNC IS RECORD
        setup: TIME;
        hold: TIME;
        edge: SIGNAL_EDGE;
        END RECORD;
--------------------------------------------------------------------
-- a collection of individual synchronous constraints
--------------------------------------------------------------------
    TYPE SYNC_CONSTRAINTS IS ARRAY (NATURAL RANGE <>) OF SYNC;
--------------------------------------------------------------------
-- physical type relational operators
--------------------------------------------------------------------
    FUNCTION  "="(a: VOLTAGE_VECTOR; b: VOLTAGE_VECTOR) RETURN BOOLEAN;
    FUNCTION  ">="(a: VOLTAGE_VECTOR; b: VOLTAGE_VECTOR) RETURN BOOLEAN;
    FUNCTION  "<="(a: VOLTAGE_VECTOR; b: VOLTAGE_VECTOR) RETURN BOOLEAN;
    FUNCTION  ">"(a: VOLTAGE_VECTOR; b: VOLTAGE_VECTOR) RETURN BOOLEAN;
    FUNCTION  "<"(a: VOLTAGE_VECTOR; b: VOLTAGE_VECTOR) RETURN BOOLEAN;
--------------------------------------------------------------------
-- reusable function for checking if the user specified operating point
-- is within the operating limits of the device. requires the
-- definition of the voltage_vector
--------------------------------------------------------------------
    FUNCTION VALID_OPERATING_POINT (phs_point: POINT) RETURN BOOLEAN;
--------------------------------------------------------------------
-- edge detection functions checks if there is a transition of a
```

Figure 4.6.1-1 continued. The EIA_567_TV Package.

```
-- signal_edge type
------------------------------------------------------------------------
    FUNCTION DETECT_EDGE(SIGNAL s:STD_ULOGIC; edge:SIGNAL_EDGE)
        RETURN BOOLEAN;
    END EIA_567_TV;

PACKAGE BODY EIA_567_TV IS
------------------------------------------------------------------------
-- function bodies for some of the physical math
------------------------------------------------------------------------
    FUNCTION "="(a:VOLTAGE_VECTOR; b:VOLTAGE_VECTOR) RETURN BOOLEAN IS
        VARIABLE result:BOOLEAN := TRUE;
        BEGIN
        FOR i IN a'RANGE LOOP
            IF ((result=TRUE) AND (a(i)/=b(i))) THEN result:=FALSE; EXIT;
                END IF;
            END LOOP;
        RETURN result;
        END "=";
    FUNCTION "<="(a:VOLTAGE_VECTOR; b:VOLTAGE_VECTOR)
        RETURN BOOLEAN IS
        VARIABLE result:BOOLEAN := TRUE;
        BEGIN
        FOR i IN a'RANGE LOOP
            IF ((result=TRUE) AND (a(i)>b(i))) THEN result:=FALSE; EXIT;
                END IF;
            END LOOP;
        RETURN result;
        END "<=";
    FUNCTION ">="(a:VOLTAGE_VECTOR; b:VOLTAGE_VECTOR)
        RETURN BOOLEAN IS
        VARIABLE result:BOOLEAN := TRUE;
        BEGIN
        FOR i IN a'RANGE LOOP
            IF ((result=TRUE) AND (a(i)<b(i))) THEN result:=FALSE; EXIT;
                END IF;
            END LOOP;
        RETURN result;
        END ">=";
    FUNCTION "<"(a:VOLTAGE_VECTOR; b:VOLTAGE_VECTOR) RETURN BOOLEAN IS
        VARIABLE result:BOOLEAN := TRUE;
        BEGIN
        FOR i IN a'RANGE LOOP
            IF ((result=TRUE) AND (a(i)>=b(i))) THEN result:=FALSE; EXIT;
                END IF;
            END LOOP;
        RETURN result;
        END "<";
    FUNCTION ">"(a:VOLTAGE_VECTOR; b:VOLTAGE_VECTOR) RETURN BOOLEAN IS
```

Figure 4.6.1-1 continued. The EIA_567_TV Package.

```
        VARIABLE result:BOOLEAN:= TRUE;
        BEGIN
        FOR i IN a'RANGE LOOP
            IF ((result=TRUE) AND (a(i)<=b(i))) THEN result:=FALSE;EXIT;
            END IF;
        END LOOP;
        RETURN result;
        END ">";
-------------------------------------------------------------------
-- function to check if user specified operating point is within limits
-------------------------------------------------------------------
    FUNCTION VALID_OPERATING_POINT(phs_point: POINT) RETURN BOOLEAN IS
        VARIABLE valid:BOOLEAN:= TRUE;
        BEGIN
        IF phs_point.temp<LOWER_TEMPERATURE_LIMIT THEN
            ASSERT FALSE REPORT "TEMPERATURE selected IS too LOW";
            valid:= FALSE;
        ELSIF phs_point.temp>UPPER_TEMPERATURE_LIMIT THEN
            ASSERT FALSE REPORT "TEMPERATURE selected IS too HIGH";
            valid:= FALSE;
        END IF;
        FOR i IN VOLTAGE_VECTOR'RANGE LOOP
            IF phs_point.SUPPLY(i)< VMIN(i) THEN
                ASSERT FALSE REPORT "VOLTAGE selected IS too LOW";
                valid:= FALSE;
            ELSIF phs_point.SUPPLY(i)> VMAX(i) THEN
                ASSERT FALSE REPORT "VOLTAGE selected IS too HIGH";
                valid:= FALSE;
            END IF;
        END LOOP;
        RETURN valid;
        END VALID_OPERATING_POINT;
-------------------------------------------------------------------
-- edge detection
-------------------------------------------------------------------
    FUNCTION DETECT_EDGE(SIGNAL s:STD_ULOGIC;edge:SIGNAL_EDGE)
        RETURN BOOLEAN IS
        BEGIN
        CASE edge IS
            WHEN eux=>RETURN ((s'EVENT AND s='X') AND s'last_value='U');
            WHEN eu0=>RETURN ((s'EVENT AND s='0') AND s'last_value='U');
            WHEN eu1=>RETURN ((s'EVENT AND s='1') AND s'last_value='U');
            WHEN euz=>RETURN ((s'EVENT AND s='Z') AND s'last_value='U');
            WHEN euw=>RETURN ((s'EVENT AND s='W') AND s'last_value='U');
            WHEN eul=>RETURN ((s'EVENT AND s='L') AND s'last_value='U');
            WHEN euh=>RETURN ((s'EVENT AND s='H') AND s'last_value='U');
            WHEN exu=>RETURN ((s'EVENT AND s='U') AND s'last_value='X');
            WHEN ex0=>RETURN ((s'EVENT AND s='0') AND s'last_value='X');
            WHEN ex1=>RETURN ((s'EVENT AND s='1') AND s'last_value='X');
```

Figure 4.6.1-1 continued. The EIA_567_TV Package.

```
            WHEN  exz=>RETURN((s'EVENT AND s='Z') AND s'last_value='X');
            WHEN  exw=>RETURN((s'EVENT AND s='W') AND s'last_value='X');
            WHEN  exl=>RETURN((s'EVENT AND s='L') AND s'last_value='X');
            WHEN  exh=>RETURN((s'EVENT AND s='H') AND s'last_value='X');
            WHEN  e0x=>RETURN((s'EVENT AND s='X') AND s'last_value='0');
            WHEN  e0u=>RETURN((s'EVENT AND s='U') AND s'last_value='0');
            WHEN  e01=>RETURN((s'EVENT AND s='1') AND s'last_value='0');
            WHEN  e0z=>RETURN((s'EVENT AND s='Z') AND s'last_value='0');
            WHEN  e0w=>RETURN((s'EVENT AND s='W') AND s'last_value='0');
            WHEN  e0l=>RETURN((s'EVENT AND s='L') AND s'last_value='0');
            WHEN  e0h=>RETURN((s'EVENT AND s='H') AND s'last_value='0');
            WHEN  e1x=>RETURN((s'EVENT AND s='X') AND s'last_value='1');
            WHEN  e10=>RETURN((s'EVENT AND s='0') AND s'last_value='1');
            WHEN  e1u=>RETURN((s'EVENT AND s='U') AND s'last_value='1');
            WHEN  e1z=>RETURN((s'EVENT AND s='Z') AND s'last_value='1');
            WHEN  e1w=>RETURN((s'EVENT AND s='W') AND s'last_value='1');
            WHEN  e1l=>RETURN((s'EVENT AND s='L') AND s'last_value='1');
            WHEN  e1h=>RETURN((s'EVENT AND s='H') AND s'last_value='1');
            WHEN  ezx=>RETURN((s'EVENT AND s='X') AND s'last_value='Z');
            WHEN  ez0=>RETURN((s'EVENT AND s='0') AND s'last_value='Z');
            WHEN  ez1=>RETURN((s'EVENT AND s='1') AND s'last_value='Z');
            WHEN  ezu=>RETURN((s'EVENT AND s='U') AND s'last_value='Z');
            WHEN  ezw=>RETURN((s'EVENT AND s='W') AND s'last_value='Z');
            WHEN  ezl=>RETURN((s'EVENT AND s='L') AND s'last_value='Z');
            WHEN  ezh=>RETURN((s'EVENT AND s='H') AND s'last_value='Z');
            WHEN  ewx=>RETURN((s'EVENT AND s='X') AND s'last_value='W');
            WHEN  ew0=>RETURN((s'EVENT AND s='0') AND s'last_value='W');
            WHEN  ew1=>RETURN((s'EVENT AND s='1') AND s'last_value='W');
            WHEN  ewz=>RETURN((s'EVENT AND s='Z') AND s'last_value='W');
            WHEN  ewu=>RETURN((s'EVENT AND s='U') AND s'last_value='W');
            WHEN  ewl=>RETURN((s'EVENT AND s='L') AND s'last_value='W');
            WHEN  ewh=>RETURN((s'EVENT AND s='H') AND s'last_value='W');
            WHEN  elx=>RETURN((s'EVENT AND s='X') AND s'last_value='L');
            WHEN  el0=>RETURN((s'EVENT AND s='0') AND s'last_value='L');
            WHEN  el1=>RETURN((s'EVENT AND s='1') AND s'last_value='L');
            WHEN  elz=>RETURN((s'EVENT AND s='Z') AND s'last_value='L');
            WHEN  elw=>RETURN((s'EVENT AND s='W') AND s'last_value='L');
            WHEN  elu=>RETURN((s'EVENT AND s='U') AND s'last_value='L');
            WHEN  elh=>RETURN((s'EVENT AND s='H') AND s'last_value='L');
            WHEN  ehx=>RETURN((s'EVENT AND s='X') AND s'last_value='H');
            WHEN  eh0=>RETURN((s'EVENT AND s='0') AND s'last_value='H');
            WHEN  eh1=>RETURN((s'EVENT AND s='1') AND s'last_value='H');
            WHEN  ehz=>RETURN((s'EVENT AND s='Z') AND s'last_value='H');
            WHEN  ehw=>RETURN((s'EVENT AND s='W') AND s'last_value='H');
            WHEN  ehu=>RETURN((s'EVENT AND s='U') AND s'last_value='H');
            WHEN  ehl=>RETURN((s'EVENT AND s='L') AND s'last_value='H');
            WHEN OTHERS=>  RETURN FALSE;
            END CASE;
        END DETECT_EDGE;
    END EIA_567_TV;
```

Figure 4.6.1-1 continued. The EIA_567_TV Package.

```
TYPEASYNCIS RECORD
    t1min,t0min,t1max,t0max:TIME;
    END RECORD;
```

4.6.1.2.3 Following is the declaration for the SIGNAL_EDGE enumerated type. A signal edge is encoded with the character "e" followed in order by the "from" state and the "to" state. The available state options are shown in Table 4.6.1.2.3-1.

signal_edge	*std_logic*	*Description*
u	*U*	*Uninitialized*
x	*X*	*Don't Know*
0	*0*	*Forcing 0*
1	*1*	*Forcing 1*
z	*Z*	*High Impedance*
w	*W*	*Weak Unknown*
l	*L*	*Weak 0*
h	*H*	*Weak 1*
d	-	*Don't Care*

Table 4.6.1.2.3-1. State Identifiers, "signal_edge" Versus "std_logic".

```
TYPESIGNAL_EDGEIS      (euu,eux,eu0,eu1,euz,euw,eul,euh,eud,
                        exu,exx,ex0,ex1,exz,exw,exl,exh,exd,
                        e0u,e0x,e00,e01,e0z,e0w,e0l,e0h,e0d,
                        e1u,e1x,e10,e11,e1z,e1w,e1l,e1h,e1d,
                        ezu,ezx,ez0,ez1,ezz,ezw,ezl,ezh,ezd,
                        ewu,ewx,ew0,ew1,ewz,eww,ewl,ewh,ewd,
                        elu,elx,el0,el1,elz,elw,ell,elh,eld,
                        ehu,ehx,eh0,eh1,ehz,ehw,ehl,ehh,ehd,
                        edu,edx,ed0,ed1,edz,edw,edl,edh,edd);
```

4.6.1.2.4 Following is the declaration for the SYNC record type. This record defined a setup and hold time with respect to a specified signal edge as shown in Figure 4.6.1.2.4-1.

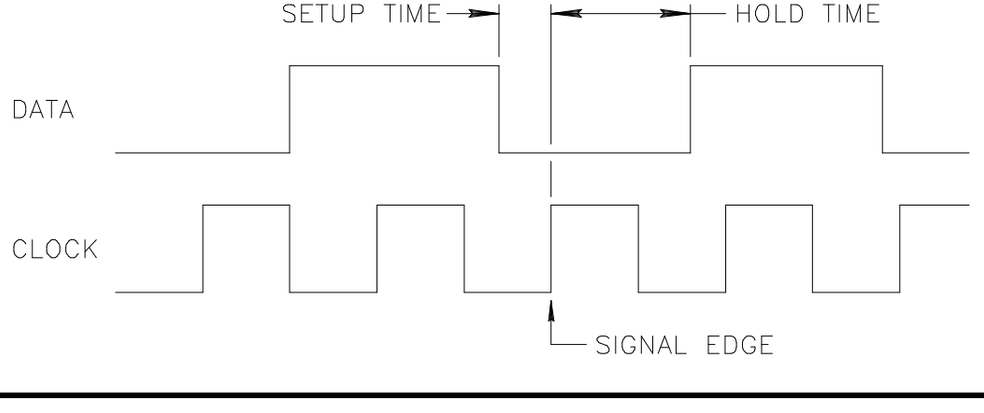

Figure 4.6.1.2.4-1. Definition of Setup and Hold Times.

```
TYPE SYNC IS RECORD
    setup: TIME;
    hold: TIME;
    edge: SIGNAL_EDGE;
    END RECORD;
```

4.6.1.3 The relational functions for VOLTAGE_VECTORs were developed in order to determine that a selected operating point lies within the operating range of the power supplies. In order to define these functions, it was first necessary to establish the dimensions of the VOLTAGE_VECTORs in the "design_EV" package. Following are declarations for the overloaded relational functions for VOLTAGE_VECTORs.

```
FUNCTION  "="(a:VOLTAGE_VECTOR; b:VOLTAGE_VECTOR) RETURN BOOLEAN;
FUNCTION  ">="(a:VOLTAGE_VECTOR; b:VOLTAGE_VECTOR) RETURN BOOLEAN;
FUNCTION  "<="(a:VOLTAGE_VECTOR; b:VOLTAGE_VECTOR) RETURN BOOLEAN;
FUNCTION  ">"(a:VOLTAGE_VECTOR; b:VOLTAGE_VECTOR) RETURN BOOLEAN;
FUNCTION  "<"(a:VOLTAGE_VECTOR; b:VOLTAGE_VECTOR) RETURN BOOLEAN;
```

4.6.1.4 The VALID_OPERATING_POINT function is used to check the user specified temperature and operating voltage to insure that they lie within the specified operating ranges. If a valid operating point is established, this function will return a TRUE. Otherwise, it will return a FALSE. The function declaration for VALID_OPERATING_POINT follows.

```
FUNCTION VALID_OPERATING_POINT (phs_point: POINT) RETURN BOOLEAN;
```

4.6.1.5 The DETECT_EDGE function is used to determine when a specified signal (s) has just had a specified transition (edge). A TRUE is returned when signal (s) has made the specified (edge) transition. Otherwise, a FALSE is returned. This function may be used when evaluating the setup and hold constraints of a model. The function declaration for DETECT_EDGE follows.

```
FUNCTION DETECT_EDGE (SIGNAL s:STD_ULOGIC; edge:SIGNAL_EDGE)
    RETURN BOOLEAN;
```

4.6.2 Interdependencies

The interdependencies for this package are as follows.

Interdependencies
EIA_567_EV
design_EV (FBG_EV for this example)

4.7 The FBG Timing View Package

4.7.1 Data Type Declarations

The timing view package for the FBG module is contained as Figure 4.7.1-1. Table 4.7.1-1 lists the constants as they pertain to this module.

4.7.1.1 The constant ASYNCS contains classes of asynchronous constraints. The default class of zero (0) implies no asynchronous constraint. This class shall always be defined and all entries in this class shall be zero. The asynchronous constraints are used to specify maximum and minimum pulse widths for the design entity. A separate constraint class is defined for each pulse width specification applied to the design. It should be noted that asynchronous constraints may be imposed on both logic one and logic zero states of a signal. Any time a constraint value does not apply in a design, zero should be entered for the value. For example if the t1min pulse width is the only asynchronous constraint specified for a pin, then the remaining elements (t1max, t0min, t0max) should be assigned a value of 0. Each asynchronous constraint contains the following elements. For the FBG module, there are no asynchronous constraints specified.

Declaration	Data Type
ASYNCS	constant async_constraints type
SYNCS	constant sync_constraints type
DELAYS	constant output_delays type
MAX_ASYNC_CONSTRAINTS_PER_PIN	constant natural type
MAX_SYNC_CONSTRAINTS_PER_PIN	constant natural type
MAX_DELAY_CONSTRAINTS_PER_PIN	constant natural type

Table 4.7.1-1. Design_TV Package Declarations.

Element	Pulse width checked
t0min	minimum low
t0max	maximum low
t1min	minimum high
t1max	maximum high

4.7.1.2 The constant SYNCS contains the classes of synchronous constraints. The default class of zero (0) implies no synchronous constraint. This class shall always be defined and the setup and hold time entries for this class shall be zero. The synchronous constraints are used to specify the setup and hold times on one signal (such as a data line) with respect to a specific edge or transition on another signal (such as a clock line). In those cases where either a setup time or a hold time is specified, but not both, the value not specified should be left as a zero. For the FBG module, there are no synchronous constraints specified.

4.7.1.3 The DELAYS constant contains the classes of propagation delays specified for the design entity. The default class of zero (0) implies no propagation delay. This class shall always be defined and all entries in this class shall be zero. Each delay class contains six delay values which cover the following transitions.

Element	Transition
tlh	from low to high
thl	from high to low
tlz	from low to high impedance
tzl	from high impedance to low
thz	from high to high impedance
tzh	from high impedance to high

4.7.2 Requirements

This package provides information to meet some of the requirements of DI-EGDS-80811, paragraphs 10.2.2.2, 10.2.2.3 and 10.2.3.2. In EIA-567, this package primarily addresses requirements in paragraph 3.3.1, sub-paragraphs 3 and 4.

4.7.3 Interdependencies

The interdependencies for this package are as follows. As noted, the only interdependency is the EIA-567 standard timing view package.

Interdependencies
EIA_567_TV

4.8 The EIA-567 Physical View Package

4.8.1 Data Type Declarations

```
--*****************************************************************
--    (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-28787/175
--         Project:                 SHARP   TIREP
--    DESC Certification
--         Status:                  TBD
--*****************************************************************
--    Revision History
--         Version:                 1.0
--              Date:               18 May 1994
--              Comments:           Original Release
--*****************************************************************
--    Module Description
--         File:                    fbg_tv_p.vhd
--         Module  Name(s):         fbg_tvpackage
--         Constraints:             none
--         Limitations:             none
--         I/O  Format(s):          none
--         Purpose and Use:         This VHDL package defines the propagation
--              delays and timing constraints for the FBG standard hardware
--              module.
--         Notes:                   none
--*****************************************************************
--    Standard Libraries/Packages
--         none
--    Associated Packages (order of analysis implied)
--         eia_567_tv               standard timing view package
--    Component  Models
--         none
--    Platform:                     486/33MHz PC
--    Software/Version:             V-System for Windows, Version 3.0
--*****************************************************************
--    Design Specification Elements
--         maximum constraints per pin   constant declarations
--         asyncs, syncs, delays         constant declarations
--*****************************************************************
USE work.EIA_567_TVALL;
PACKAGEfbg_tv IS
-----------------------------------------------------------------------
-- Declaration of input constraints which are appropriate for this model
```

Figure 4.7.1-1. The FBG_TV Package.

```
------------------------------------------------------------------
    CONSTANT ASYNCS: ASYNC_CONSTRAINTS;
    CONSTANT SYNCS: SYNC_CONSTRAINTS;
    CONSTANT DELAYS: OUTPUT_DELAYS;
--*****************************************************************
-- The following constants define the maximum number of constraints
-- which will be associated to any given pin.  An example is a data
-- which is sampled by three different clocks (triple sampling).  The
-- data will have three setups and three holds.      So if this were the
-- largest number of constraints associated with a pin the
-- max_async_constraints_per_pin should be assigned 3.          These constants
-- are used in the eia_567_pv (physical view) package and must be
-- defined beforehand.
--*****************************************************************
    CONSTANT MAX_ASYNC_CONSTRAINTS_PER_PIN: NATURAL:=1;
    CONSTANT MAX_SYNC_CONSTRAINTS_PER_PIN: NATURAL:=1;
    CONSTANT MAX_DELAY_CONSTRAINTS_PER_PIN: NATURAL:=1;
    END fbg_tv;

PACKAGE BODY fbg_tv IS
------------------------------------------------------------------
-- Definition of asynchronous and synchronous constraint classes
--*****************************************************************
-- The asyncs constant is used to declare asynchronous timing constraint
-- classes (i.e. minimum and/or maximum pulse widths). An asynchronous
-- timing constraint class should be declared for any minimum or
-- maximum pulse width specification in the design. As a minimum, the
-- default class 0 shall always be declared.
--*****************************************************************
-- Note: The subject module is completely combinational. There are
-- no synchronous or asynchronous constraints imposed on this
--design.
------------------------------------------------------------------
    CONSTANT ASYNCS: ASYNC_CONSTRAINTS:=(
-- Async constraint class #0 for no constraints
        (0=>(t1min=>0    ns,t0min=>0    ns,t1max=>0    ns,t0max=>0    ns)));
--*****************************************************************
-- The syncs constant is used to declare synchronous timing constraint
-- classes (i.e. setup and hold times). A synchronous timing constraint
-- class should be declared for any setup and/or hold time specification
-- in the design. As a minimum, the default class 0 shall always be
--declared.
--*****************************************************************
    CONSTANT SYNCS: SYNC_CONSTRAINTS:=(
-- Sync constraint class #0 for no constraints
        (0=>(setup=>0      ns,hold=>0      ns,edge=>SIGNAL_EDGE'LEFT)));
------------------------------------------------------------------
-- Definition of output delay classes
--*****************************************************************
```

Figure 4.7.1-1 continued. The FBG_TV Package.

```
-- The delays constant is used to declare propagation delay classes.
-- Propagation delay classes should be included for minimum, maximum
-- and nominal conditions, when available. As a minimum, the default
-- class 0 shall always be declared.
--*****************************************************************
    CONSTANT DELAYS OUTPUT_DELAYS=(
-- Delays defined in eia_567_tv
-- Ouput delay class #0, no delay association
        (tlh=>0   ns,thl=>0   ns,tlz=>0   ns,thz=>0   ns,tzl=>0   ns,tzh=>0   ns),
-- Output delay class #1 (ax/bx to fx nominal)
        (tlh=>70  ns,thl=>60  ns,tlz=>0   ns,thz=>0   ns,tzl=>0   ns,tzh=>0   ns),
-- Output delay class #2 (ax/bx to fx best)
        (tlh=>10  ns,thl=>10  ns,tlz=>0   ns,thz=>0   ns,tzl=>0   ns,tzh=>0   ns),
-- Output delay class #3 (ax/bx to fx worst)
        (tlh=>130 ns,thl=>110 ns,tlz=>0   ns,thz=>0   ns,tzl=>0   ns,tzh=>0   ns));
    END fbg_tv;
```

Figure 4.7.1-1 continued. The FBG_TV Package.

The next package defined in the EIA-567 modeling approach is the physical view package. This package is used to define some of the data types which will be used to establish the electronic data sheet (EDS) for this model. Table 4.8.1-1 contains a list of the data types defined by this package. The content of this package is listed in Figure 4.8.1-1.

Declaration	Data Type
ESD_CLASS	enumerated type
EIA_STRING	character array type
PIN_RECORD	record type
ASYNC_ARRAY	integer array type
SYNC_ARRAY	integer array type
DELAY_ARRAY	integer array type
PIN_POINTERS	record type

Table 4.8.1-1. EIA_567_PV Package Declarations.

4.8.1.1 The ESD_CLASS simply allows the user to define the ElectroStatic Device (ESD) sensitivity in accordance with MIL-STD-1686. If not otherwise specified, this entry will default to "unknown".

4.8.1.2 The EIA_STRING provides a general purpose character string which is used for pin and signal correlation and may find application in other areas such as part numbers, component values and ratings, etc. As defined in the package, the string length for this data type is 27.

4.8.1.3 The PIN_RECORD provides a data type through which pin numbers can be associated with signal names. Due to restrictions imposed on VHDL naming conventions, making such associations through VHDL identifiers can be confusing. Following is the PIN_RECORD declaration.

```
TYPE PIN_RECORD IS RECORD
    pin_id: EIA_STRING;
    signal_name: EIA_STRING;
    END RECORD;
```

```
-------------------------------------------------------------------------
-- eia 567 standard package
-------------------------------------------------------------------------
-- brief description:          This is the eia 567 package.    The contents of
--                             this package should not be changed.
--                             Implementation of the package body may be changed
--                             as long as the specific capabilities are not
--                             changed.
-- version                     1.1
--history:                     broke the original eia_567 package into four smaller
--                             packages (len finegold 12/22/92)
-- annotations:                none
-- analysis dependencies:      design specific timing view package
-- limitations:                none
-- supplementary information:  none
-- development platform:       model technologies v-system for ibm pc running
--                             on a softpc emulation of dos on a mac iici and
--                             clsi vhdl learning kit for sun workstations,
--                             synopsys, vantage, intermetrics/valid
-- vhdl software version:      model technologies v-system version 1.3
--                             clsi vhdl learning kit version 1.0
-------------------------------------------------------------------------
USE work.fbg_tv.ALL;
PACKAGE EIA_567_PVIS
-------------------------------------------------------------------------
-- physical view related declarations
-------------------------------------------------------------------------
-- esd specifications, mil-std-1686
-------------------------------------------------------------------------
    TYPE ESD_CLASS IS (UNKNOWN, i, ii, iii);
-------------------------------------------------------------------------
-- pinout data
-------------------------------------------------------------------------
    TYPE EIA_STRING IS ARRAY (27 DOWNTO 1) OF CHARACTER;
    TYPE PIN_RECORD IS RECORD
        pin_id: EIA_STRING;
        signal_name: EIA_STRING;
        END RECORD;
-------------------------------------------------------------------------
-- single operating point definition
-------------------------------------------------------------------------
    TYPE ASYNC_ARRAY IS ARRAY (MAX_ASYNC_CONSTRAINTS_PER_PIN DOWNTO 1)
        OF INTEGER;
    TYPE SYNC_ARRAY IS ARRAY (MAX_SYNC_CONSTRAINTS_PER_PIN DOWNTO 1)
        OF INTEGER;
    TYPE DELAY_ARRAY IS ARRAY (MAX_DELAY_CONSTRAINTS_PER_PIN DOWNTO 1)
        OF INTEGER;
    TYPE PIN_POINTERS IS RECORD
        async_spec: ASYNC_ARRAY;
```

Figure 4.8.1-1. The EIA_567_PV Package.

```
        sync_spec: SYNC_ARRAY;
        delay_spec: DELAY_ARRAY;
        elec_spec: NATURAL;
        END RECORD;
    END EIA_567_PV;
```

Figure 4.8.1-1 continued. The EIA_567_PV Package.

4.8.1.4 The ASYNC_ARRAY, SYNC_ARRAY and DELAY_ARRAY data types simply establish arrays based upon the maximum number of constraints which may be applied to a single pin. In order to define this data type, it is necessary to have the corresponding maximum constraint counts provided by the "design_TV" package. For the FBG example, each of these constraints is 1.

4.8.1.5 The PIN_POINTERS record type forms the heart of the electronic data sheet. It is through this record type that pointers to the appropriate parameter classes, as defined in the "design_EV" and "design_TV" packages will be mapped. Following is the declaration for this record type.

```
TYPE PIN_POINTERS IS RECORD
    async_spec: ASYNC_ARRAY;
    sync_spec: SYNC_ARRAY;
    delay_spec: DELAY_ARRAY;
    elec_spec: NATURAL;
    END RECORD;
```

4.8.2 Interdependencies

The interdependencies for this package are as follows. As noted, the only interdependency is the design timing view package.

Interdependencies
design_TV (FBG_TV for this example)

4.8.3 Package Body

The EIA_567_PV package does not contain any subprograms or deferred constant declarations, only data type declarations. As a result, it does not require a package body.

4.9 The FBG Physical View Package

4.9.1 Data Type Declarations

The key feature of the design physical view package is that it contains the electronic data sheet. This package ties the elements defined in the electrical and timing views together in order to establish the port specifications for the design entity. The physical view package for the FBG module appears as Figure 4.9.1-1. In this package, the data types of Table 4.9.1-1 are defined.

4.9.1.1 The first element declared in this package is the PIN_INDEX enumerated data type. This data type contains an element for every pin in the design entity. The convention selected for this index is the signal name followed by the "ndx" string. This data type will be used in the pin and signal correlation listing as well as in the EDS. It should be noted that this data type includes an index for power (vccndx) and ground (gndndx).

Declaration	Data Type
PIN_INDEX	enumerated type
PIN_AND_SIGNAL_CORROLATION	record array type
INTERFACE_CONNECTIONS	constant type
POINT_SPECIFICATION	pin_pointer array type
OPNT	record type
ELECTRONIC_DATA_SHEET	opnt array type
PART_NAME	constant string type
ESD_PROTECTION	constant esd_class type
OUTLINE_DRAWING	constant string type
MAXIMUM_POWER_DISSIPATION	constant power type
EDS	electronic data sheet type

Table 4.9.1-1. Design_PV Package Declarations.

4.9.1.2 The INTERFACE_CONNECTIONS constant of the PIN_AND_SIGNAL_CORROLATION record type is used to link the signal names to the pin numbers as defined for the physical unit. Since this constant is comprised of EIA_STRING elements, it is not constrained by the VHDL naming conventions. The user is free to include any valid ASCII character(s) in the signal names and pin numbers do not need to be prefixed with an alphabetic character. Each element of this constant is referenced through the PIN_INDEX data type. When making assignments, the user must specify either the range for each element or fill out the entire 27 character field (spaces recommended) for any entry which is less than 27 characters.

4.9.1.3 The POINT_SPECIFICATION is the data type which is established and used by the OPNT data type to provide the links to characteristic/timing classes which have been defined in the design electrical and timing view packages.

4.9.1.4 The OPNT record type is used to define a single operating point in the EDS. This record type is comprised of two elements. The first element "selpoint" specifies the operating conditions at which the point applies. This includes operating voltage, temperature and propagation delay. The second element "edsnfo" contains the indexes or pointers (for all pins) to the timing and parametric classes defined in the electrical and timing view classes for the design.

4.9.1.5 The ELECTRONIC_DATA_SHEET is defined as a array of 1 or more operating points (OPNT). The VHDL DID requires that operating points be provided for nominal, minimum and maximum output delays (see DI-EGDS-80811, paragraph 10.2.3.2), as a minimum.

4.9.1.5.1 In the FBG physical view package, there are three operating points (OPNT) defined for the EDS of the model. These points are as follows.

Delay	Voltage	Temperature
tmin	5.5 V	0 degrees_c
tnom	5.0 V	25 degrees_c
tmax	4.5 V	60 degrees_c

4.9.1.5.2 At each operating point, a specification is provided for each pin of this design. For all pins in this design, the asynchronous and synchronous classes are assigned 0 since there are no synchronous or asynchronous constraints or specifications for this model. For propagation delays, the appropriate delay class is specified for the minimum, maximum and nominal cases for each output pin. Finally, the electrical specification class is provided for each input and output pin. It is the set of these pin specifications provided at each operating point that forms the ELECTRONIC_DATA_SHEET.

```
--*****************************************************************
--    (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-28787/175
--         Project:                 SHARP   TIREP
--    DESC Certification
--         Status:                  TBD
--*****************************************************************
--    Revision History
--         Version:                 1.0
--              Date:               18 May 1994
--              Comments:           Original Release
--*****************************************************************
--    Module Description
--         File:                    fbg_pv_p.vhd
--         Module  Name(s):         fbg_pv package
--         Constraints:             none
--         Limitations:             none
--         I/O  Format(s):          none
--         Purpose and Use:         This VHDL package defines the pin and signal
--              correlation table and the operating points for the electronic
--              data sheet.
--         Notes:                   none
--*****************************************************************
--    Standard Libraries/Packages
--         none
--    Associated Packages (order of analysis implied)
--         eia_567_ev               standard electrical view package
--         eia_567_tv               standard timing view package
--         eia_567_pv               standard physical view package
--    Component  Models
--         none
--    Platform:                     486/33MHz PC
--    Software/Version:             V-System for Windows, Version 3.0
--*****************************************************************
--    Design Specification Elements
--         pin_index                     enumerated type
--         interface_connections         constant declaration
--         electronic_data_sheet         type declaration
--         miscellaneous                 constant declarations
--         eds                           constant declaration
```

Figure 4.9.1-1. The FBG_PV Package.

```
--*****************************************************************
USE work.EIA_567_EV.ALL;
USE work.EIA_567_TV.ALL;
USE work.EIA_567_PV.ALL;
PACKAGE fbg_pv IS
--*****************************************************************
-- Definition of an enumerated type which is used as an index into most
-- of the data stored in the electronic data sheet, the enumeration uses
-- signal/supply/nc names.
--*****************************************************************
    TYPE PIN_INDEX IS  (a0ndx,a1ndx,a2ndx,a3ndx,a4ndx,a5ndx,b0ndx,b1ndx,
        b2ndx,b3ndx,b4ndx,b5ndx,f0ndx,f1ndx,f2ndx,f3ndx,vccndx,gndndx);
-----------------------------------------------------------------
-- Definition of a type and the signal to pin corrolation
-----------------------------------------------------------------
    TYPE PIN_AND_SIGNAL_CORROLATIONS IS ARRAY (PIN_INDEX) OF PIN_RECORD;
-----------------------------------------------------------------
-- The following constant definition is represented in one of its most
-- expanded forms.
-----------------------------------------------------------------
    CONSTANT INTERFACE_CONNECTIONS: PIN_AND_SIGNAL_CORROLATIONS:=(
--*****************************************************************
-- Begins the array for each connection defined previously with the type
-- pin_index, there is an entry of information. Each entry contains
-- two elementsof information defined by the type. Note that pin names
-- which are considered illegal can be represented in the text string
-- creating an explicit link between the port name used in VHDL and the
-- name actually used in hardware.
--*****************************************************************
        a0ndx=>(pin_id=>"39                                              ",
            signal_name=>"a0                                             "),
        a1ndx=>(pin_id=>"38                                              ",
            signal_name=>"a1                                             "),
        a2ndx=>(pin_id=>"29                                              ",
            signal_name=>"a2                                             "),
        a3ndx=>(pin_id=>"30                                              ",
            signal_name=>"a3                                             "),
        a4ndx=>(pin_id=>"27                                              ",
            signal_name=>"a4                                             "),
        a5ndx=>(pin_id=>"28                                              ",
            signal_name=>"a5                                             "),
        b0ndx=>(pin_id=>"23                                              ",
            signal_name=>"b0                                             "),
        b1ndx=>(pin_id=>"22                                              ",
            signal_name=>"b1                                             "),
        b2ndx=>(pin_id=>"21                                              ",
            signal_name=>"b2                                             "),
        b3ndx=>(pin_id=>"25                                              ",
            signal_name=>"b3                                             "),
```

Figure 4.9.1-1 continued. The FBG_PV Package.

```
            b4ndx=>(pin_id=>"26                                    ",
                 signal_name=>"b4                                  "),
            b5ndx=>(pin_id=>"24                                    ",
                 signal_name=>"b5                                  "),
            f0ndx=>(pin_id=>"34                                    ",
                 signal_name=>"f0                                  "),
            f1ndx=>(pin_id=>"33                                    ",
                 signal_name=>"f1                                  "),
            f2ndx=>(pin_id=>"31                                    ",
                 signal_name=>"f2                                  "),
            f3ndx=>(pin_id=>"35                                    ",
                 signal_name=>"f3                                  "),
            vccndx=>(pin_id=>"1                                   ",
                 signal_name=>"vcc                                 "),
            gndndx=>(pin_id=>"10                                   ",
                 signal_name=>"gnd                                 "));
-----------------------------------------------------------------------
-- Definition of an array of records.  Each record holds the specific
-- indexes for the signal/pin into the stored information in the views
-- of the electronic data sheet.   A single array element of
-- point_specification contains all the information needed to describe
-- the timing characteristics at an operating point.
-----------------------------------------------------------------------
    TYPE POINT_SPECIFICATION IS ARRAY (PIN_INDEX) OF PIN_POINTERS;
    TYPE OPNT IS RECORD
        selpoint: POINT;
        edsnfo: POINT_SPECIFICATION;
        END RECORD;
--*********************************************************************
-- In the following type declaration, the user needs to identify the
-- number of operating points which will be provided in the EDS.
--*********************************************************************
    TYPE ELECTRONIC_DATA_SHEET IS ARRAY (0 TO 2) OF OPNT;
-- The set of all operating points is an electronic data sheet.
--*********************************************************************
-- Definition of name, esd protection, outline drawing, and power for
-- this particular model as well as a few other constant values that
-- need to be visible.
--*********************************************************************
    CONSTANT PART_NAME: STRING:="fbg";
    CONSTANT ESD_PROTECTION: ESD_CLASS:= UNKNOWN;
    CONSTANT OUTLINE_DRAWING: STRING:="tbd";
    CONSTANT MAXIMUM_POWER_DISSIPATION: POWER:=450  mw;
    CONSTANT EDS: ELECTRONIC_DATA_SHEET;
    END fbg_pv;

PACKAGE BODY fbg_pv IS
--*********************************************************************
-- Definition of an array of the indexes. Each array holds the specific
```

Figure 4.9.1-1 continued. The FBG_PV Package.

```
-- index for all of the singal/pins for a specific operating point.
--*****************************************************************
    CONSTANT EDS:ELECTRONIC_DATA_SHEET:=(
-- The best case operating point
        OPERATING_SELECTION(pos(TMIN)=>(
            selpoint=>(selection=>  TMIN,temp=>-55  degrees_c,
                SUPPLY=>(0=>5.5  v)),
            edsnfo=>(
            a0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>2),
            a1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>2),
            a2ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>2),
            a3ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>2),
            a4ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>2),
            a5ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>2),
            b0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>2),
            b1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>2),
            b2ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>2),
            b3ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>2),
            b4ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>2),
            b5ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>2),
            f0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(2)),elec_spec=>6),
            f1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(2)),elec_spec=>6),
            f2ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(2)),elec_spec=>6),
            f3ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(2)),elec_spec=>6),
            vccndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>0),
            gndndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>0))),
-- The nominal or typical operating point
        OPERATING_SELECTION(pos(TNOM)=>(
            selpoint=>(selection=>  TNOM,temp=>27  degrees_c,SUPPLY=>(0=>5  v)),
            edsnfo=>(
            a0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
```

Figure 4.9.1-1 continued. The FBG_PV Package.

```
                delay_spec=>(1=>(0)),elec_spec=>1),
            a1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>1),
            a2ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>1),
            a3ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>1),
            a4ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>1),
            a5ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>1),
            b0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>1),
            b1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>1),
            b2ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>1),
            b3ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>1),
            b4ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>1),
            b5ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>1),
            f0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(1)),elec_spec=>6),
            f1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(1)),elec_spec=>6),
            f2ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(1)),elec_spec=>6),
            f3ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(1)),elec_spec=>6),
            vccndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>0),
            gndndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>0))),
    -- The worst case operating point
        OPERATING_SELECTION(pos(TMAX)=>(
            selpoint=>(selection=>  TMAX,temp=>125  degrees_c,
                SUPPLY=>(0=>4.5  v)),
            edsnfo=>(
            a0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>4),
            a1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>4),
            a2ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>4),
            a3ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>3),
            a4ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
```

Figure 4.9.1-1 continued. The FBG_PV Package.

```
                delay_spec=>(1=>(0)),elec_spec=>3),
            a5ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>5),
            b0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>4),
            b1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>4),
            b2ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>4),
            b3ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>4),
            b4ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>4),
            b5ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>4),
            f0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(3)),elec_spec=>6),
            f1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(3)),elec_spec=>6),
            f2ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(3)),elec_spec=>6),
            f3ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(3)),elec_spec=>6),
            vccndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>0),
            gndndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
                delay_spec=>(1=>(0)),elec_spec=>0))));
    END fbg_pv;
```

Figure 4.9.1-1 continued. The FBG_PV Package.

4.9.1.5.3 As an example, consider the following line of code. The "sync_spec" and the "async_spec" are both mapped to class 0. This means that there are no synchronous or asynchronous constraints associated with this pin. The "delay_spec" is mapped to class 2 (in the FBG_TV package) and the "elec_spec" is mapped to class 6 (in the FBG_EV package). In general, all pins will have an associated "elec_spec" except for power and ground which receive the default class 0. "sync_spec" and "async_spec" are generally associated with inputs while "delay_spec" is associated with outputs.

```
f0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
    delay_spec=>(1=>(2)),elec_spec=>6),
```

4.9.1.6 The PART_NAME, ESD_PROTECTION, OUTLINE_DRAWING and MAXIMUM_POWER_DISSIPATION constants are provided as information about the design. In the case of the FBG module some of these elements have yet to be defined.

4.9.2 Interdependencies

The interdependencies of this package are as follows.

Interdependencies
EIA_567_TV
EIA_567_PV

4.10 The EIA-567 Toolbox Package

4.10.1 Subprogram Declarations

The EIA-567 toolbox package contains a collection of supplemental functions which are useful in developing the electronic data sheet model for the design entity. Table 4.10.1-1 contains a list of the functions provided by this package. The source code listing for this package is contained as Figure 4.10.1-1.

Declaration	Subprogram Type
MAX	function
F	function
GET_TIMING	function
EDGE_CHECK	function

Table 4.10.1-1. EIA_567_TB Package Declarations.

4.10.1.1 The function MAX simply takes two times and returns the larger of the two. This function is employed by the F function. Following is the function declaration for MAX.

```
FUNCTION MAX(a,b: TIME) RETURN TIME;
```

4.10.1.2 The function F is used to determine the appropriate propagation delay based upon a signal transition and the information provided in the design timing view package as referenced in the electronic data sheet. The inputs to this function include the old and new (soon_to_be) states of a given signal along with a delay record. The old and new states are used to determine the type of signal transition involved. The delay record is then used by the function to provide the propagation delay for the specified signal change. The function declaration for F is as follows.

```
FUNCTION F(d: DELAY;SIGNAL old,soon_to_be:STD_LOGIC) RETURN TIME;
```

4.10.1.3 The GET_TIMING function is used to determine if the user selected operating point matches one of the operating points specified in the electronic data sheet. This function requires an exact match of the operating point in order to select the operating point. The manner in which an operating point is selected when an exact match is not found, is left for the modeller to define. As a minimum, it is recommended that the user revert to the TNOM delay point and generate a warning to inform the user that the exact operating point was not found.

```
FUNCTION GET_TIMING (op_point:POINT;xgen:BOOLEAN) RETURN OPNT;
```

4.10.1.4 Finally, the EDGE_CHECK function is a simple function which can be used to determine the type of transition occurring on a signal based upon the IEEE-STD-1164 STD_LOGIC type. This function returns an element of the enumerated type SIGNAL_EDGE as defined in the EIA-567 timing view package.

```
FUNCTION EDGE_CHECK(SIGNAL sig:STD_LOGIC) RETURN SIGNAL_EDGE;
```

4.10.2 Interdependencies

The interdependencies for this package are as follows.

Interdependencies
STD_LOGIC_1164
EIA_567_TV
design_PV (FBG_PV for this example)

4.11 The Input Driver

4.11.1 Operation

```
-- ***************************************************************************
-- eia 567 standard package
-- ***************************************************************************
-- brief description:          this is the eia 567 package. the contents of
--                             the package should not be changed. implementation
--                             of the package body may be changed as long as the
--                             specific capabilities are not changed.
-- version:                    1.0 (first release)
-- history:                    no previous changes to model 11/18/91
-- annotations:                none
-- analysis dependencies:      std_logic_1164
--                             eia_567_tv
--                             fbg_pv
-- limitations:                none
-- supplementary information:  none
-- development platform:       model technologies v-system for ibm pc running
--                             on a softpc emulation of dos on a mac iici.
-- vhdl software version:      model technologies v-system version 1.3
-- ***************************************************************************
LIBRARY ieee;
USE ieee.STD_LOGIC_1164.ALL;
USE work.EIA_567_TV.ALL;
USE work.fbg_tv.ALL;
USE work.fbg_pv.ALL;
PACKAGE EIA_567_TB IS
-------------------------------------------------------------------------
-- function which returns the maximum of two times
-------------------------------------------------------------------------
    FUNCTION MAX(a,b: TIME) RETURN TIME;
-------------------------------------------------------------------------
-- function selects the correct delay transition type and
-- returns the time.
--  a<=aprime  after  f(delays(eds.edsnfo(serialoutndx).delay_spec,a,aprime)
-------------------------------------------------------------------------
    FUNCTION F(d: DELAY; SIGNAL old,soon_to_be:STD_LOGIC) RETURN TIME;
-------------------------------------------------------------------------
-- function which selects the correct info
-- based on the user specified operating point
-- and x generation
-------------------------------------------------------------------------
    FUNCTION GET_TIMING (op_point:POINT;xgen:BOOLEAN) RETURN OPNT;
-------------------------------------------------------------------------
-- function which detects the type of transition
-- a signal makes and returns the value of type
-- signal_edge (found in eia_567_tv)
-------------------------------------------------------------------------
    FUNCTION EDGE_CHECK(SIGNAL sig:STD_LOGIC) RETURN SIGNAL_EDGE;
    END EIA_567_TB;
```

Figure 4.10.1-1. The EIA_567_TB Toolbox Package.

```
PACKAGE BODY EIA_567_TB IS
    FUNCTION GET_TIMING (op_point:POINT;xgen:BOOLEAN) RETURN OPNT IS
        VARIABLE found:BOOLEAN:= FALSE;
        VARIABLE temppoint:OPNT;
        BEGIN
-- first scan all the operating points stored for an exact match.
        loop1:FOR index IN EDS'RANGE LOOP
            IF (EDS(index).selpoint = op_point) THEN
                temppoint:=( EDS(index));
                found:= TRUE;
                EXIT loop1;
                END IF;
            END LOOP loop1;
        IF NOT(found) THEN
-- very optional
-- this is the point to include equations if you have them.
-- otherwise tell the user what the model will do
            ASSERT FALSE
                REPORT "you have chosen an invalid operating POINT"
                SEVERITY ERROR;
            temppoint:= EDS(1);
            END IF;
-- if xgeneration is turned off, no constraint checking should occur so all constraint pointers
-- will be set to point at 0 ns constraints. this will have the effect of turning off checking
        IF NOT(xgen) THEN
            FOR i IN PIN_INDEX LOOP
                FOR j IN MAX_ASYNC_CONSTRAINTS_PER_PIN DOWNTO 1 LOOP
                    temppoint.edsnfo(i).async_spec(j):=0;          -- pointer position 0
                    END LOOP;
                FOR j IN MAX_SYNC_CONSTRAINTS_PER_PIN DOWNTO 1 LOOP
                    temppoint.edsnfo(i).sync_spec(j):=0;
                    END LOOP;
                END LOOP;
            END IF;
            RETURN(temppoint);
        END GET_TIMING;
-------------------------------------------------------------------------
-- function which returns the maximum of two times
-------------------------------------------------------------------------
    FUNCTION MAX(a,b:TIME) RETURN TIME IS
        BEGIN
        IF (a>b) THEN RETURN(a); ELSE RETURN(b); END IF;
        END MAX;
-------------------------------------------------------------------------
-- function selects the correct delay transition type and
-- returns the time.
-- a<=aprime after f(delays(eds.edsnfo(serialoutndx).delay_spec,a,aprime)
-------------------------------------------------------------------------
    FUNCTION F(d:DELAY;SIGNAL old,soon_to_be:STD_LOGIC) RETURN TIME IS
```

Figure 4.10.1-1 continued. The EIA_567_TB Package.

```
        BEGIN
        CASE old IS
        WHEN   '1'|'H'=>
            CASE soon_to_be IS
                WHEN   '0'|'L'=>RETURN (d.thl);
                WHEN   '1'|'H'=>RETURN (0 ns);
                WHEN   'Z'=>RETURN (d.thz);
                WHEN  'U'|'X'|'-'=>   RETURN (MAX(d.thl,d.tlh));
                WHEN   'W'=>RETURN (MAX(d.thz,d.tlz));
                END CASE;
        WHEN   '0'|'L'=>
            CASE soon_to_be IS
                WHEN   '1'|'H'=>RETURN (d.tlh);
                WHEN   '0'|'L'=>RETURN (0 ns);
                WHEN   'Z'=>RETURN (d.tlz);
                WHEN   'U'|'X'|'-'=>RETURN (MAX(d.thl,d.tlh));
                WHEN   'W'=>RETURN (MAX(d.thz,d.tlz));
                END CASE;
        WHEN   'Z'|'W'=>
            CASE soon_to_be IS
            WHEN   '0'|'L'=>RETURN (d.tzl);
                WHEN   '1'|'H'=>RETURN (d.tzh);
                WHEN   'Z'=>RETURN (0 ns);
                WHEN   'U'|'X'|'-'=>RETURN (MAX(d.tzh,d.tzl));
                WHEN   'W'=>RETURN (0 ns);
                END CASE;
        WHEN   'U'|'X'|'-'=>
            CASE soon_to_be IS
                WHEN   '0'|'L'=>RETURN (d.thl);
                WHEN   '1'|'H'=>RETURN (d.tlh);
                WHEN   'Z'=>RETURN (MAX(d.thz,d.tlz));
                WHEN   'U'|'X'|'-'=>RETURN (0 ns);
                WHEN   'W'=>RETURN (MAX(d.thz,d.tlz));
                END CASE;
            END CASE;
        END F;
    FUNCTION EDGE_CHECK (SIGNAL sig:STD_LOGIC) RETURN SIGNAL_EDGE IS
        BEGIN
        RETURN (SIGNAL_EDGE'val((STD_ULOGIC'pos(sig'last_value)*9)+
            (STD_ULOGIC'pos(sig))));
        END EDGE_CHECK;
    END EIA_567_TB;
```

Figure 4.10.1-1 continued. The EIA_567_TB Package.

The input driver simply implements a transfer function with a generic delay. This delay is established to model the routing delays generated when the design is implemented in hardware. Since routing delays are constant regardless of the signal transition, the delay is applied to all signal activity. The generic "pull_value" forms the initialization value for the driver. The source code for this driver is depicted in Figure 4.11.1-1.

```
-----------------------------------------------------------------------
-- model:                    i_driver
--brief description:          this is a general purpose i driver which
--                           when used, will insure all the logic interface
--                           requirements of eia567 are met.
-- model supplier:           len finegold (619) 573-3640/computer sciences corp.
--functionality:              configurable through instantiation as an:
--                           input and      with pullup or tristate with pulldown.
--version:                    1.00
--history:                    initial design (12/25/91)
-- annotations:              this is a model of the behaviour.
-- analysis dependencies:    std_logic_1164.all;
-- limitations:              as designed, this driver will yeild exact
--                           and correct results if is not connected
--                           with a signal which is resolving more than
--                           one signal.    if there are more than one signal
--                           the error will be a bounded by the difference
--                           in time between the shortest wire connection
--                           and the longest wire connection.
--fidelity:                   exact
--discrepancies:             none
-- supplementary information: to instantiate in various configurations:
-- development platform:     model technologies v-system for ibm pc running
--                           on a softpc emulation of dos on a mac iici.
--                           clsi vhdl learning kit for sun 4
-- vhdl software version:     model technologies v-system version 1.3,
--                           clsi version 1.1
LIBRARY ieee;
USE work.STD_LOGIC_1164ALL;
ENTITY i_driver IS
    GENERIC(win_d: TIME;
        pull_value: STD_ULOGIC:='Z');
    PORT(d: OUT STD_LOGIC;
        y: IN STD_LOGIC);
    END i_driver;
ARCHITECTURE general OF i_driver IS
    SIGNAL yprime: STD_LOGIC;
    BEGIN
    PROCESS(y)
        BEGIN
        yprime<=y    AFTER win_d;
        END PROCESS;
    yprime<=pull_value;
    d<=   TO_X01(yprime);
    END general;
```

Figure 4.11.1-1. A Generic Input Driver.

4.11.1.1 This driver employs the TO_X01 conversion function from the STD_LOGIC_1164 package when determining the driver output. As a result, this driver will output only the states 'X', '0' or '1'. The code of Figure 4.11.1-1 defines the generic input driver.

4.11.1.2 It should be noted that EIA-567, paragraph 3.4.2.1.3 dictates the use of the 'U' (uninitialized) state on all output ports upon power up initialization. Since the output of an input driver is an internal node in the complete model, this requirement may not apply to this component.

4.11.2 Interdependencies

The interdependencies for this component are as follows.

Interdependencies
STD_LOGIC_1164

4.12 The Output Driver

4.12.1 Operation

The output driver (also known as the tri-state or t_driver) operates in a manner similar to the input driver, however it implements propagation delays which are sensitive to transitions on the input signal (a). This propagation delay is calculated through the F function provided in the EIA-567 toolbox package, using the "delay_y" generic imported to the component. The source code for this model is provided in Figure 4.12.1-1.

4.12.1.1 The "pull_value" generic is an initialization value for the output of the driver. EIA-567, paragraph 3.4.2.1.3 dictates that this value should be 'U'.

4.12.1.2 The DRIVE input to this component is used to tri-state the output. In the logic '0' state, the output (y) will be driven to the 'Z' state. Otherwise, the output (y) will track the input (a) after the appropriate propagation delay. The purpose of the tri-state capability on this component is not clear. It seems the desirable approach would be to have the core model generate the tri-state condition leaving the output driver to just pass this condition on.

4.12.1.3 The "lo_y" generic is a routing or wire delay, similar to the "wir_d" delay provided with the input driver. This delay is simply added to the delay generated by the F function.

4.12.1.4 The RISING_EDGE and FALLING_EDGE functions used by this component are provided in the STD_LOGIC_1164 package.

4.12.2 Interdependencies

The interdependencies for this component are as follows.

Interdependencies
STD_LOGIC_1164
EIA_567_TV
EIA_567_TB

4.13 The Bidirectional Driver

4.13.1 Operation

The bidirectional driver simply merges the functions of the input and output drivers. On the circuit (internal) side of the driver, there is a separate input and output while on the external side of the driver, there is a single input/output pin. The source code for the bidirectional driver is contained in Figure 4.13.1-1.

4.13.1.1 The author has been informed that this version of the bidirectional driver is not completely operational. It is provided as reference only. Additional work may be needed to bring this model to an operational status.

4.13.1.2 For additional information on the characteristics and parameters associated with this model, refer to paragraphs 4.11 and 4.12 in this document.

4.13.2 Interdependencies

The interdependencies for this component are as follows.

```
-------------------------------------------------------------------------
-- model:                   t_driver
-- brief description:       this is a general purpose tristate output driver which
--                          when used, will insure all the logic interface
--                          requirements of eia567 are met.
-- model supplier:          len finegold (619) 573-3640/computer sciences corp.
-- functionality:           configurable through instantiation as an:
--                          output,tristate,open drain, open collector,
--                          tristate with pullup or tristate with pulldown.
-- version:                 1.00
-- history:      '          initial design (12/25/91)
-- annotations:             this is a model of the behaviour.
-- analysis dependencies:   std_logic_1164.all;
--                          eia_567_tv.all
--                          eia_567_tb.all
-- limitations:             as designed, this driver will yeild exact
--                          and correct results if is not connected
--                          with a signal which is resolving more than
--                          one signal.    if there are more than one signal
--                          the error will be a bounded by the difference
--                          in time between the shortest wire connection
--                          and the longest wire connection.
-- fidelity:                exact
-- discrepancies:           none
-- supplementary
-- development platform:    model technologies v-system for ibm pc running
--                          on a softpc emulation of dos on a mac iici.
--                          clsi vhdl learning kit for sun 4
-- vhdl software version:   model technologies v-system version 1.3,
--                          clsi version 1.1
LIBRARY ieee;
USE ieee.STD_LOGIC_1164.ALL;
USE work.EIA_567_TV.ALL;
USE work.EIA_567_TB.ALL;
ENTITY t_driver IS
    GENERIC (delay_y:DELAY_ -- six value record
        pull_value:STD_ULOGIC:='Z';
        lo_y:TIME:=0  ns);
    PORT (a,DRIVE:IN STD_ULOGIC;
        y:INOUT STD_LOGIC);
    END t_driver;
ARCHITECTURE general OF t_driver IS
    SIGNAL zee:STD_ULOGIC:='Z';
    BEGIN
    PROCESS(DRIVE,a)
        BEGIN
        IF RISING_EDGE(DRIVE) THEN
                y<=a    AFTER (delay_y,zee,a)+lo_y;
            ELSIF FALLING_EDGE(DRIVE) THEN
```

Figure 4.12.1-1. A Generic Output Driver.

```
                y<='Z'   AFTERF(delay_y,y,zee)+lo_y;
            ELSIF (DRIVE='1') THEN
                y<=a   AFTERF(delay_y,y,a)+lo_y;
            ELSIF (DRIVE='0') THEN
                y<='Z'   AFTERF(delay_y,y,zee)+lo_y;
            ELSE   y<='X';
            END IF;
        END PROCESS;
    y<=pull_value;
    END general;
```

Figure 4.12.1-1 continued. A Generic Output Driver.

Interdependencies
STD_LOGIC_1164
EIA_567_TV
EIA_567_TB

4.14 Simulation Control Variables

Before proceeding with the synchronous and asynchronous checkers, it is appropriate to make mention of a couple of simulation control variables which have been defined for use by the model. These are the "x_generation" and "m_generation" variables, a description of which are provided below.

4.14.1 'X_Generation"

The "x_generation" variable or "x_gen" boolean generic as called out in the synchronous and asynchronous checkers is used to control the generation of 'X' or unknown states from the checker components when a timing violation is detected.

4.14.1.1 If "x_gen" is true, then the checker will drive a 'X' to signify that a synchronous or asynchronous constraint violation has occurred. If "x_gen" is false, then 'X' states will not be generated upon the occurrence of a checked violation.

4.14.1.2 This feature is not required under DI-EGDS-80811, however it is required by EIA-567, paragraph 3.4.2.1.3.

4.14.2 'M_Generation"

The "m_generation" variable or "m_gen" severity level generic is used by the checker components to control the generation of assertion messages. This variable may be used to suppress assertion messages generated by the checkers.

4.14.2.1 By definition, the checking of assertions shall be enabled only if the severity of the associated error is greater than "m_gen".

4.14.2.2 In the models provided, if "m_gen" is assigned the value of NOTE then the assertion messages of these checkers will not be generated in the event that a timing constraint is violated. If however, "m_gen" is set to anything other than NOTE (i.e. WARNING, ERROR or FAILURE), assertion messages will be printed when a timing constraint is violated. This is inconsistent with the definition since the severity level of the assertions generated by the checkers is NOTE (default).

4.14.2.3 As with the "x_gen" variable, the use of the "m_gen" variable is required by EIA-567, paragraph 3.4.2.1.3. This variable is not required by DI-EGDS-80811.

4.15 The Asynchronous Checker

4.15.1 Operation

```
-----------------------------------------------------------------------
-- model:                  b_driver
-- brief description:      this is a general purpose io driver which
--                         when used, will insure all the logic interface
--                         requirements of eia567 are met.
-- model supplier:         len finegold (619) 573-3640/computer sciences corp.
-- functionality:          configurable through instantiation as an:
--                         input,output,tristate,open drain, open collector,
--                         tristate with pullup or tristate with pulldown.
-- version:                1.00
-- history:                initial design (12/25/91)
-- annotations:            this is a model of the behaviour.
-- analysis dependencies:  std_logic_1164.all;
--                         eia_567_tv.all
--                         eia_567_tb.all
-- limitations:            as designed, this driver will yeild exact
--                         and correct results if is not connected
--                         with a signal which is resolving more than
--                         one signal.    if there are more than one signal
--                         the error will be a bounded by the difference
--                         in time between the shortest wire connection
--                         and the longest wire connection.
-- fidelity:               exact
-- discrepancies:          none
-- supplementary
-- development platform:   model technologies v-system for ibm pc running
--                         on a softpc emulation of dos on a mac iici.
--                         clsi vhdl learning kit for sun 4
-- vhdl software version:  model technologies v-system version 1.3,
--                         clsi version 1.1
LIBRARY ieee;
USE ieee.STD_LOGIC_1164.ALL;
USE work.EIA_567_TV.ALL;
USE work.EIA_567_TB.ALL;
ENTITY b_driver IS
    GENERIC (delay_y:DELAY_LENGTH; -- six value record
        pull_value:STD_ULOGIC:='Z';
        win_d:TIME:=0 ns;
        lo_y:TIME:=0 ns);
    PORT (a,DRIVE:IN STD_ULOGIC;
        d:OUT STD_ULOGIC;
        y:INOUT STD_LOGIC);
    END b_driver;
ARCHITECTURE general OF b_driver IS
    SIGNAL zee:STD_ULOGIC:='Z';
    BEGIN
        PROCESS(DRIVE,a,y)
        BEGIN
        IF RISING_EDGE(DRIVE) THEN
```

Figure 4.13.1-1. A Generic Bidirectional Driver.

```
            y<=a    AFTERF(delay_y,zee,a)+lo_y;
            d<=   TO_UX01(y) AFTERF(delay_y,zee,a)+lo_y;
        ELSIF FALLING_EDGE(DRIVE) THEN
            y<='Z'    AFTERF(delay_y,y,zee)+lo_y;
            d<=   TO_UX01(y) AFTERF(delay_y,y,zee)+lo_y;
        ELSIF (DRIVE = '1') THEN
            y<=a    AFTERF(delay_y,y,a)+lo_y;
            d<=   TO_UX01(y) AFTERF(delay_y,y,a)+lo_y;
        ELSIF (DRIVE = '0') THEN
            y<='Z'    AFTERF(delay_y,y,zee)+lo_y;
            d<=   TO_X01(y) AFTER win_d;
        ELSE
            y<='X';
            d<='X'    AFTER win_d;
        END IF;
    END PROCESS;
y<=pull_value;
END general;
```

Figure 4.13.1-1 continued. A Bidirectional Driver.

The asynchronous checker is similar to the input and output drivers in that it ultimately passes signals from the input to the output. Along the way, however, the checker does a little bit more by reporting on those signals. Refer to Figure 4.15.1-1 for the code of an asynchronous checker.

4.15.1.1 The key function of the asynchronous checker is to check the minimum and maximum pulse widths of a signal as specified through the "asyncconstraint" generic. In support of this function, the checker makes use of a couple of simulation control generics called "x_gen" and "m_gen".

4.15.1.2 In operation, the asynchronous checker will evaluate the time a signal has been in either a '1' or a '0' state when the signal transitions from that state. If the checker determines that the signal violates one of the asynchronous timing constraints, an assertion error message and/or a 'X' output state will be generated, if enabled.

4.15.1.3 It should be noted that any constraint with a value of zero is not checked. This is the default case for any constraint.

4.15.2 Interdependencies

The interdependencies for this component are as follows.

Interdependencies
STD_LOGIC_1164
EIA_567_TV
EIA_567_TB

4.16 The Synchronous Checker

4.16.1 Operation

Overall, the synchronous checker functions in a manner similar to the asynchronous checker in that it makes use of the "x_gen" and "m_gen" simulation control variables in order define the manner in which it handles detected violations. The key difference in the synchronous checker is that it is used to test for adequate setup and hold times for the model. In order to accomplish this, it is necessary that these times be referenced to a specific transition on another signal (such as a clock). Refer to Figure 4.16.1-1 for the source code of the synchronous checker.

```
--------------------------------------------------------------------
-- model:                    async_checker
-- brief description:        this is an artifice which will check
--                           transition times for pulse width violations.
-- model supplier:           len finegold (619) 573-3640/computer sciences corp.
-- functionality:            if the setup or hold constraints are greater than
--                           0 ns, the constraints will be checked.  no negitive
--                           values will be accepted.  instead, the check specified
--                           should be modified. (a negative setup time is really
--                           a hold time)
-- version:                  1.00
-- history:                  initial design (12/25/91)
-- annotations:              none
-- analysis dependencies:    use work.utilities.all;
--                           use work.std_logic_1164.all;
--                           use work.eia_567_tv.all;
--                           use work.eia_567_tb.all;
-- limitations:              none
-- fidelity:                 this model is a low fidelity model.  it is a fully functional
--                           behavioral model but lacks timing associated with a specific
--                           technology.
-- discrepancies:            none
-- supplementary information: none
-- development platform:     model technologies v-system for ibm pc running
--                           on a softpc emulation of dos on a mac iici.
--                           clsi vhdl learning kit for sun 4
-- vhdl software version:    model technologies v-system version 1.3,
--                           clsi version 1.1
LIBRARY ieee;
USE ieee.STD_LOGIC_1164.ALL;
USE work.EIA_567_TV.ALL;
USE work.EIA_567_TB.ALL;
ENTITY async_checker IS
    GENERIC (x_gen: BOOLEAN;
        m_gen: SEVERITY_LEVEL;
        asyncconstraint: ASYNC);
    PORT (clk: IN STD_LOGIC;
        comes_in_clean: IN STD_LOGIC;
        goes_out_x: OUT STD_LOGIC);
    END async_checker;
ARCHITECTURE behav OF async_checker IS
    BEGIN
    PROCESS (clk)
        VARIABLE generate_error: BOOLEAN := FALSE;
        VARIABLE edge_value: SIGNAL_EDGE;
        BEGIN
        edge_value := EDGE_CHECK (clk);
        IF (asyncconstraint.t1min /= 0 ns) THEN
            IF (edge_value >= e1u) AND (edge_value <= e1h) AND
```

Figure 4.15.1-1. A Generic Asynchronous Checker.

```
                (clk'delayed'last_event<asyncconstraint.t1min)   THEN
                ASSERT  (m_gen=NOTE) REPORT"t1min violation";
                generate_error:=   TRUE;
                END IF;
            END IF;
        IF  asyncconstraint.t1max/=0  ns  THEN
            IF  (edge_value>=e1u)  AND  (edge_value<=e1h)  AND
                (clk'delayed'last_event>asyncconstraint.t1max)   THEN
                ASSERT  (m_gen=NOTE) REPORT"t1max violation";
                generate_error:=   TRUE;
                END IF;
            END IF;
        IF  asyncconstraint.t0min/=0  ns  THEN
            IF  (edge_value>=e0u)  AND  (edge_value<=e0h)  AND
                (clk'delayed'last_event>asyncconstraint.t0min)   THEN
                ASSERT  (m_gen=NOTE) REPORT"t0min violation";
                generate_error:=   TRUE;
                END IF;
            END IF;
        IF  asyncconstraint.t0max/=0  ns  THEN
            IF  (edge_value>=e0u)  AND  (edge_value<=e0h)  AND
                (clk'delayed'last_event>asyncconstraint.t0max)   THEN
                ASSERT  (m_gen=NOTE) REPORT"t0max violation";
                generate_error:=   TRUE;
                END IF;
            END IF;
        IF  generate_error  THEN   goes_out_x<='X';
            ELSE    goes_out_x<=comes_in_clean;
            END IF;
        generate_error:=   FALSE;
        END PROCESS;
    goes_out_x<=comes_in_clean;
    END behav;
```

Figure 4.15.1-1 continued. An Asynchronous Checker.

4.16.1.1 In operation, the synchronous checker will log the occurrence of the reference signal edge as specified in the "syncconstraint" generic. It will then check to insure that the "data_in" signal has been stable for the specified setup and hold times.

4.16.1.2 When a violation is detected, the checker will report the violation if it is enabled to do so through the "m_gen" variable. Additionally, the checker will generate an unknown or 'X' state if the "x_gen" variable is true. In the model provided, assertion messages will be enabled when "m_gen" is either WARNING, ERROR or FAILURE. Assertion messages will be inhibited when "m_gen" is NOTE.

4.16.1.3 In the synchronous checker model provided, there are three inputs declared, namely "clk", "data", and "comes_in_clean". When instantiated, the inputs "data" and "comes_in_clean" are tied together forming the data input to the checker. At this point, it is not understood why this separation has been made on this component.

4.16.1.3 It should be noted that any constraint with a value of zero is not checked. This is the default case for any constraint.

4.16.2 Interdependencies

```
-------------------------------------------------------------------
-- model:                    sync_checker
-- brief description:        this is an artifice which will check setup and
--                           hold times.
-- model supplier:           len finegold (619) 573-3640/computer sciences corp.
-- functionality:            if the setup or hold constraints are greater than
--                           0 ns, the constraints will be checked.  no negitive
--                           values will be accepted.  instead, the check specified
--                           should be modified. (a negative setup time IS really
--                           a hold time)
-- version:                  1.00
-- history:                  initial design (12/25/91)
-- annotations:              none
-- analysis dependencies:    use work.utilities.all;
--                           use work.std_logic_1164.all;
--                           use work.eia_567_tv.all;
--                           use work.eia_567_tb.all;
-- limitations:              none
-- fidelity:                 this model is a low fidelity model.  it is a fully functional
--                           behavioral model but lacks timing associated with a specific
--                           technology.
-- discrepancies:            none
-- supplementary information: none
-- development platform:     model technologies v-system for ibm pc running
--                           on a softpc emulation of dos on a mac iici.
--                           clsi vhdl learning kit for sun 4
-- vhdl software version:    model technologies v-system version 1.3,
--                           clsi version 1.1
LIBRARY ieee;
USE ieee.STD_LOGIC_1164.ALL;
USE work.EIA_567_TV.ALL;
USE work.EIA_567_TB.ALL;
ENTITY sync_checker IS
    GENERIC (x_gen: BOOLEAN;
        m_gen: SEVERITY_LEVEL;
        syncconstraint: SYNC);
    PORT (clk,data: IN STD_LOGIC;
        comes_in_clean: IN STD_LOGIC;
        goes_out_x: OUT STD_LOGIC);
    END sync_checker;
ARCHITECTURE behav OF sync_checker IS
    SIGNAL internal_clk: STD_LOGIC;
    BEGIN
    internal_clk <=  TRANSPORT clk AFTER (syncconstraint.hold);
    PROCESS
        VARIABLE generate_error: BOOLEAN := FALSE;
        BEGIN
        goes_out_x <= 'Z';
        LOOP
```

Figure 4.16.1-1. A Generic Synchronous Checker.

```
            WAITUNTIL (EDGE_CHECK(clk)=syncconstraint.edge);
            IF syncconstraint.setup/= 0 ns THEN
                IF NOT(data'stable(syncconstraint.setup)) THEN
                    ASSERT (m_gen=NOTE)
                        REPORT "setup violation. data NOT stable"
                        SEVERITYNOTE;
                    IF x_gen THEN generate_error:=TRUE;
                        ELSE generate_error:=FALSE;
                        END IF;
                    ELSE generate_error:=FALSE;
                    END IF;
                ELSE goes_out_x<=comes_in_clean;
                END IF;
            IF generate_error THEN goes_out_x<='X';
                ELSE goes_out_x<='Z';
                END IF;
            END LOOP;
        END PROCESS;
    PROCESS
        VARIABLE generate_error: BOOLEAN:= FALSE;
        BEGIN
        goes_out_x<='Z';
        LOOP
            WAITUNTIL (EDGE_CHECK(internal_clk)=syncconstraint.edge);
            IF syncconstraint.hold/= 0 ns THEN
                IF NOT(data'stable(syncconstraint.hold)) THEN
                    ASSERT (m_gen=NOTE)
                        REPORT "hold violation. data NOT stable";
                    IF x_gen THEN generate_error:=TRUE;
                        ELSE generate_error:=FALSE;
                        END IF;
                    ELSE generate_error:=FALSE;
                    END IF;
                ELSE goes_out_x<='Z';
                END IF;
            IF generate_error THEN goes_out_x<='X';
                ELSE goes_out_x<='Z';
                END IF;
            END LOOP;
        END PROCESS;
    goes_out_x<=comes_in_clean;
    END behav;
```

Figure 4.16.1-1 continued. A Generic Synchronous Checker.

The interdependencies for this component are as follows.

Interdependencies
STD_LOGIC_1164
EIA_567_TV
EIA_567_TB

4.17 The Electronic Data Sheet Model

4.17.1 Operation

The time has come to put the complete model together. This is accomplished in the code of Figure 4.17.1-1. Referring back to the FBG module example introduced earlier, the core model has been instantiated with the appropriate input and output drivers. It should be noted that this model does not have any bidirectional pins and is not specified with any synchronous or asynchronous constraints. Therefore, these driver and checker components are not actually required for this example. They have been provided for reference only.

4.17.1.1 In this model, the "wi_xx" generics are used to bring in routing delay information for the input and output drivers. The "user_operating_point" is used to establish the conditions under which the model is simulated. It should be noted that an exact match for the operating point conditions must be found in the EDS in order to avoid the generation of an error. Finally, the "x_generation" and "m_generation" simulation control variables are included to handle the manner in which timing violations are handled.

4.17.1.2 In this model, the GET_TIMING will try to find an exact match of the user selected operating point in the EDS. If an exact match is not found, this function will generate an error message and resort to the nominal operating point. A supplemental operation of this function is to set all "sync_spec" and "async_spec" pointers to class 0 if the "x_gen" simulation control variable is false. This will have the effect of inhibiting all timing constraint checking for the model. The GET_TIMING function resides in the EIA_567_TB package (see paragraph 4.10).

4.17.1.3 The VALID_OPERATING_POINT function checks the user specified operating point to insure that the voltage and temperature selected lie within the specified range for this model. The model reports an error if either of these constraints are violated. The VALID_OPERATING_POINT function resides in the EIA_567_TV package (see paragraph 4.6).

4.17.1.4 The example provided herein includes a process which implements the function of "vcc" and "gnd". This process tests to see that "vcc" is a forcing '1' and that "gnd" is a forcing '0'. If these conditions are not met, the model will output 'U' states.

4.17.2 Interdependencies

The interdependencies for this model are as follows.

Interdependencies
STD_LOGIC_1164
EIA_567_EV
design_EV (FBG_EV for this example)
EIA_567_TV
design_TV (FBG_TV for this example)
EIA_567_PV
design_PV (FBG_PV for this example)
EIA_567_TB

Components
core_model (FBG_CORE for this example)
I_DRIVER
T_DRIVER

4.18 Disclaimer

4.18.1 EIA-567A

At the time of this writing, it is known that version A of EIA-567 is nearing release. Once released, the changes identified in version A will be investigated for possible inclusion in this document

```
--*****************************************************************
--    (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-28787/175
--         Project:            SHARP   TIREP
--    DESC Certification
--         Status:             TBD
--*****************************************************************
--    Revision History
--         Version:            1.0
--              Date:          18 May 1994
--              Comments:      Original Release
--*****************************************************************
--    Module Description
--         File:               fbg_eds.vhd
--         Module  Name(s):    fbg_edsentity/arch_fbg_eds_strarchitecture
--         Constraints:        none
--         Limitations:        none
--         I/O  Format(s):     std_logic
--         Purpose and Use:    This VHDL module describes the FBG standard
--              hardware module in accordance with M28787/175. This module
--              implements a 12-bit population counter. It features 12 inputs
--              and 4 outputs. The number of logic 1's on the 12 inputs are
--              summed and the result is provided (in binary) on the outputs.
--              The outputs range from a binary 0 (0000) to a binary 12 (1100).
--         Notes:              none
--*****************************************************************
--    Standard Libraries/Packages
--         std_logic_1164      standard multi-value logic package
--    Associated Packages (order of analysis implied)
--         eia_567_ev          standard electrical view package
--         fbg_ev              FBG module electrical view package
--         eia_567_tv          standard timing view package
--         fbg_tv              FBG module timing view package
--         eia_567_pv          standard physical view package
--         fbg_pv              FBG module physical view package
--         eia_567_tb          standard toolbox package
--         fbg_core            behavorial VHDL module core
--    Component  Models
--         i_driver            input timing driver
--         t_driver            output timing driver
```

Figure 4.17.1-1. The EDS Model of the FBG Module.

```
--          fbg_core                the fbg core component
--    Platform:                     486/33MHz PC
--    Software/Version:             V-System for Windows, Version 3.0
--****************************************************************
-- Design Specification Elements
--   entire file
--****************************************************************
LIBRARY ieee;
USE ieee.STD_LOGIC_1164.ALL;
USE work.EIA_567_EV.ALL;
USE work.fbg_ev.ALL;
USE work.EIA_567_TV.ALL;
USE work.fbg_tv.ALL;
USE work.EIA_567_PV.ALL;
USE work.fbg_pv.ALL;
USE work.EIA_567_TB.ALL;
ENTITY fbg_eds IS
      GENERIC(wi_a0,wi_a1,wi_a2,wi_a3,wi_a4,wi_a5,
            wi_b0,wi_b1,wi_b2,wi_b3,wi_b4,wi_b5:TIME:=0 ns;
            wi_f0,wi_f1,wi_f2,wi_f3:TIME:=0 ns;
*****************************************************************
 -- Valid operating points are:
 --      selection=tmin, temp=-55 degrees_c, supply=(5.5 v)
 --      selection=tnom, temp=27 degrees_c, supply=(5 v)
 --      selection=tmax, temp=125 degrees_c, supply=(4.5 v)
*****************************************************************
            user_operating_point: POINT:=(selection=> TNOM,temp=>27 degrees_c,
                  SUPPLY=>(0=>5 v));
*****************************************************************
 -- The x_generation global variable controls 'X' state generation by
 -- the "async_checker" and "sync_checker" modules when an
 -- assert condition is violated.  If true, then 'X' states are generated
 -- on assertion violations.  If false, then 'X' states will not be
 -- generated.
*****************************************************************
                  x_generation: BOOLEAN:= TRUE;
*****************************************************************
 -- The m_generation variable is used to control reports provided by
 -- assertion statements in the "sync_checker" and "async_checker
 -- modules.  If the severity_level of the m_generation variable is less
 -- than or equal to the serverity_level of an assertion violation, then
 -- the assertion report will be generated.  Otherwise, the report will
 -- not be provided.      Valid severity_level's are note, warning, error
 -- or failure.
*****************************************************************
                  m_generation: SEVERITY_LEVEL:= WARNING);
            PORT(a0,a1,a2,a3,a4,a5,b0,b1,b2,b3,b4,b5:IN STD_LOGIC;
                  f0,f1,f2,f3:INOUT STD_LOGIC;
                  vcc,gnd:IN STD_LOGIC);
```

Figure 4.17.1-1 continued. The FBG EDS Model.

```
    END fbg_eds;
ARCHITECTURE arch_fbg_eds_str OF fbg_eds IS
    COMPONENT fbg_core
        PORT(a0,a1,a2,a3,a4,a5,b0,b1,b2,b3,b4,b5: IN STD_LOGIC;
            f0,f1,f2,f3: OUT STD_LOGIC);
        END COMPONENT;
    COMPONENT i_driver
        GENERIC(win_d: TIME;
            pull_value: STD_ULOGIC:='U');
        PORT(d: OUT STD_LOGIC;
            y: IN STD_LOGIC);
        END COMPONENT;
    COMPONENT t_driver
        GENERIC(delay_y: DELAY;
            pull_value: STD_ULOGIC:='U';
            lo_y: TIME:=0 ns);
        PORT(a: IN STD_ULOGIC;
            DRIVE: IN STD_ULOGIC:='1';
            y: INOUT STD_LOGIC);
        END COMPONENT;
    SIGNAL ia0,ia1,ia2,ia3,ia4,ia5,ib0,ib1,ib2,ib3,ib4,ib5: STD_LOGIC;
    SIGNAL if0,if1,if2,if3,vf0,vf1,vf2,vf3: STD_LOGIC;
    FOR ALL: fbg_core USE ENTITY work.fbg_core;
    FOR ALL: i_driver USE ENTITY work.i_driver;
    FOR ALL: t_driver USE ENTITY work.t_driver;
    CONSTANT EDS: OPNT := GET_TIMING(user_operating_point,x_generation);
    BEGIN
    ASSERT VALID_OPERATING_POINT(user_operating_point)
        REPORT "operating outside OF the recommended RANGE OF this design"
        SEVERITY WARNING;
-- unit will not work without power
    if0<=vf0   WHEN vcc='1' AND gnd='0' ELSE 'U';
    if1<=vf1   WHEN vcc='1' AND gnd='0' ELSE 'U';
    if2<=vf2   WHEN vcc='1' AND gnd='0' ELSE 'U';
    if3<=vf3   WHEN vcc='1' AND gnd='0' ELSE 'U';
    s0:i_driver GENERIC MAP(win_d=>wi_a0)
        PORT MAP(d=>ia0,y=>a0);
    s1:i_driver GENERIC MAP(win_d=>wi_a1)
        PORT MAP(d=>ia1,y=>a1);
    s2:i_driver GENERIC MAP(win_d=>wi_a2)
        PORT MAP(d=>ia2,y=>a2);
    s3:i_driver GENERIC MAP(win_d=>wi_a3)
        PORT MAP(d=>ia3,y=>a3);
    s4:i_driver GENERIC MAP(win_d=>wi_a4)
        PORT MAP(d=>ia4,y=>a4);
    s5:i_driver GENERIC MAP(win_d=>wi_a5)
        PORT MAP(d=>ia5,y=>a5);
    s6:i_driver GENERIC MAP(win_d=>wi_b0)
        PORT MAP(d=>ib0,y=>b0);
```

Figure 4.17.1-1 continued. The FBG EDS Model.

```
s7:i_driver GENERIC MAP(win_d=>wi_b1)
     PORT MAP(d=>ib1,y=>b1);
s8:i_driver GENERIC MAP(win_d=>wi_b2)
     PORT MAP(d=>ib2,y=>b2);
s9:i_driver GENERIC MAP(win_d=>wi_b3)
     PORT MAP(d=>ib3,y=>b3);
s10:i_driver GENERIC MAP(win_d=>wi_b4)
     PORT MAP(d=>ib4,y=>b4);
s11:i_driver GENERIC MAP(win_d=>wi_b5)
     PORT MAP(d=>ib5,y=>b5);
s12:t_driver
     GENERIC MAP(delay_y=>   DELAYS(EDS.edsnfo(f0ndx).delay_spec(1)),lo_y=>wi_f0)
     PORT MAP(a=>if0,y=>f0);
s13:t_driver
     GENERIC MAP(delay_y=>   DELAYS(EDS.edsnfo(f1ndx).delay_spec(1)),lo_y=>wi_f1)
     PORT MAP(a=>if1,y=>f1);
s14:t_driver
     GENERIC MAP(delay_y=>   DELAYS(EDS.edsnfo(f2ndx).delay_spec(1)),lo_y=>wi_f2)
     PORT MAP(a=>if2,y=>f2);
s15:t_driver
     GENERIC MAP(delay_y=>   DELAYS(EDS.edsnfo(f3ndx).delay_spec(1)),lo_y=>wi_f3)
     PORT MAP(a=>if3,y=>f3);
s16:fbg_core PORT MAP(ia0,ia1,ia2,ia3,ia4,ia5,ib0,ib1,ib2,ib3,ib4,ib5,
     vf0,vf1,vf2,vf3);
END  arch_fbg_eds_str;
```

Figure 4.17.1-1 continued. The FBG EDS Model.

This page intentionally left blank