Chapter 5

5.0 Variations on EIA-567

5.1 Introduction

5.1.1 Approach

This chapter was established to address some of the concerns and limitations in the EIA-567 approach described in chapter 4. In order to keep things from getting too complicated, only sections of code affected by the recommended or proposed change(s) will be shown. Once all such recommendations have been introduced, a complete example model will be presented.

5.1.2 Drivers and Checkers

It is worth noting that the generic driver and checker components provided in this chapter are original developments by the author. During the early stages of the TIREP project, the team did not have access to the variants developed by Mr. Finegold for the F-22 model example. Functional variations between these implementations will be discussed in detail where appropriate.

5.2 Transition Time Modeling

5.2.1 The Output Driver and Transition Time Modeling

In the EIA-567 modeling approach provided previously, propagation delays are modeled using output driver components. These components do not take into account transition time modeling. This is one of the characteristics which it was necessary to capture for the TIREP models. As a result, the capabilities of the timing view packages were expanded to provide this capability.

5.2.1.1 In order to model transition times, it is first necessary to make a couple of assumptions. The first assumption is based upon the placement of the transition with respect to the propagation delay. For simplicity, the model which follows assumes that the transition time is symmetrical about the propagation delay. The second assumption is that the model will deal only with high-to-low and low-to-high transition times. There has been no basis identified to support modeling transition times to or from other states.

5.2.1.1.1 In order to model an output transition, it was decided to employ the ’X’ (unknown) state during the transition time. Based upon the manner in which this time is defined, a potential error condition has been introduced which needs to be guarded against. This condition occurs when the transition time is greater than twice the propagation delay. Since the transition time is placed symmetrically around the propagation delay, a transition time which is greater than twice the propagation delay will result in a condition where an output signal would actually have to anticipate an input signal change.

5.2.1.1.2 Due to the fact that the model developed for this feature will have the capability to generate ’X’ states and it will also perform an associated timing check on the transition time versus the propagation delay, it was decided to employ the "m_gen" and "x_gen" simulation control variables as defined in chapter 4.

5.2.1.2 To develop the transition time modeling capability, the DELAY record declaration from the EIA_567_TV package was expanded to include transition time low-to-high (ttlh) and transition time high-to-low (tthl) as follows.

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

5.2.1.3 Next, when using the new DELAY type to define the delay classes in the design timing view package, "ttlh" and "tthl" also need to be assigned values.

5.2.1.4 Finally, it is necessary to provide the output driver with the capability to utilize these additional times. This was accomplished as shown in Figure 5.2.1.4-1.

```
--*****************************************************************
--  (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:          EIA-567
--      Project:            SHARP   TIREP
--  DESC Certification
--      Status:             TBD
--*****************************************************************
--  Revision History
--      Version:            1.1
--          Date:           23 September 1994
--          Comments:       Corrected bug in transition time model check
--      Version:            1.0
--          Date:           23 May 1994
--          Comments:       Original Release
--*****************************************************************
--  Module Description
--      File:               o_drive.vhd
--      Module  Name(s):    o_driverentity/arch_o_driver_beharchitecture
--      Constraints:        none
--      Limitations:        none
--      I/O  Format(s):     std_logic
--      Purpose and Use:    This VHDL module describes an output driver used
--          to implement propagation delays and transition timing when used
--          in conjunction with the EIA-567 packages.
--      Notes:              none
--*****************************************************************
--  StandardLibraries/Packages
--      std_logic_1164      standard multi-value logic package
--  Associated Packages (order of analysis implied)
--      eia_567             standard eia package
--  Component  Models
--      none
--  Platform:               486/33MHz PC
--  Software/Version:       V-System for Windows, Version 3.0
--*****************************************************************
--  Design Specification Elements
--      none
--*****************************************************************
LIBRARYieee;
LIBRARYeia;
```

Figure 5.2.1.4-1. An Enhanced Output Driver

```
USE ieee.STD_LOGIC_1164.ALL;
USE eia.EIA_567.ALL;
ENTITY o_driver IS
    GENERIC(delay_y:DELAY;
        wir_o:TIME:=0 ns;
        pin_data: EV_SIGNAL_LIMIT;
        m_gen:SEVERITY_LEVEL;
        x_gen:BOOLEAN);
    PORT(a:IN STD_LOGIC;
        y:INOUT STD_LOGIC:='U');
    END o_driver;
ARCHITECTURE arch_o_driver_beh OF o_driver IS
    BEGIN
    PROCESS(a)
        VARIABLE del,del1,del2:TIME;
        BEGIN
        IF y/=a THEN
            del:= F(delay_y,y,a);
-- variables used to check for timing violations
            IF y/='Z' AND a/='Z' THEN
                del1:=del-delay_y.tthl/2;
                del2:=del-delay_y.ttlh/2;
                ELSE del1:=0 ns;del2:=0 ns;
                END IF;
            IF   m_gen<=WARNING THEN
-- check for ttlh and/or tthl 2*del timing violation
                ASSERT NOW=0 ns OR (del1>=0 ns AND del2>=0 ns)
                    REPORT "OUTPUT transition TIME ERROR"
                    SEVERITY WARNING;
                END IF;
            ELSE del:=0 ns;del1:=0 ns;del2:=0 ns;
            END IF;
        IF (x_gen) AND (delay_y.tthl>0 ns) AND (delay_y.ttlh>0 ns) THEN
-- if x_gen is true then perform transition time simulation on
-- low-to-high and high-to-low transitions
            IF (y='1' OR y='H') AND (a='0' OR a='L') THEN
                IF del1<0 ns THEN del1:=0 ns;
                    END IF;
                y<='X'  AFTER del1+wir_o,
                    a AFTER del+delay_y.tthl/2;
                ELSIF (y='0' OR y='L') AND (a='1' OR a='H') THEN
                IF del2<0 ns THEN del2:=0 ns;
                    END IF;
                y<='X'  AFTER del2+wir_o,
                    a AFTER del+delay_y.ttlh/2;
-- for all other transitions, disregard the transition time
                ELSE y<=a AFTER del+wir_o;
                END IF;
-- if x_gen is false then skip transition simulation
```

Figure 5.2.1.4-1 continued. An Enhanced Output Driver.

```
        ELSE  y<=a  AFTER  del+wir_o;
        END IF;
    END PROCESS;
END  arch_o_driver_beh;
```

Figure 5.2.1.4-1 continued. An Enhanced Output Driver.

5.2.1.4.1 The output driver component operation is basically divided into two sections. The first section is based upon the state of the "m_gen" variable. If a timing violation is detected in which a transition time is specified as greater than twice the propagation delay, and the "m_gen" variable is a severity level of WARNING or greater, then a warning message is printed. Note, in the generic drivers and checkers presented in this chapter, "m_gen" is assigned the severity level of the lowest severity messages to be checked. In this case, the severity level of the message is WARNING, so if "m_gen" is assigned to either WARNING or NOTE, this assertion will be tested, and if violated, reported.

5.2.1.4.2 The second section of the driver is based upon the state of "x_gen" and the type of transition which occurred. If "x_gen" is TRUE and there is a low-to-high or a high-to-low transition on the input to the driver, the 'X' state will be generated during the transition time centered at the propagation delay. For any other type of transition or if "x_gen" is FALSE, the output will be assigned the value of the input following the appropriate propagation delay.

5.2.2 Output Driver Comparison

There are a couple of differences between this output driver and the output driver presented in the previous chapter.

5.2.2.1 First of all, this driver does not feature a "pull_value" generic. The "pull_value" was used to provide an initialization value to the output of the driver. Since EIA-567, paragraph 3.4.2.1.3 dictates that this initialization value be 'U', this generic is not deemed necessary for this component.

5.2.2.2 Secondly, the port for this driver does not include a tri-state capability. Since the driver will operate on all STD_LOGIC states, a 'Z' can be passed as easily as a '0' or a '1'. This provides the design engineer with the freedom to incorporate tri-states within the core model. The ability to tri-state the output drivers does offer an advantage which is worthy of consideration. When using the model as a component in a larger model, the tri-state outputs may be engaged thereby providing a quick mechanism for isolating the model from the remainder of the system.

5.2.3 It should be noted that EIA-567, paragraph 3.4.1 does address transition (rise and fall) times indicating that they are required in order to provide a good approximation of hardware performance, however, they are not overriding objectives, especially for behavioral simulations. Therefore, it is left to the design engineer to determine what is appropriate for the model being developed.

5.3 Electrical View Considerations

5.3.1 Linking Electrical View Classes to Pins

A careful examination of the EIA-567 modeling approach provided in chapter 4 will reveal that the link between the electrical view pin classes in the "design_EV" file to the pins on the model, is inferred, not explicit. In order to complete this link, a generic was added to each of the generic driver components of the EV_SIGNAL_LIMIT type. In service, this generic provides no functional benefit to the components other than to make the link between the electrical view class and the pin explicit. Following is the declaration added to the generic clause of the driver components.

```
pin_data: EV_SIGNAL_LIMIT;
```

When the driver is instantiated into the EDS model, the following generic port map element is then included. The "pin_ndx" is the index of the pin for which the driver is being used, as assigned in the PIN_INDEX declaration in the "design_PV" package.

pin_data=> **ELECTRICAL_PIN_SPEC**(**EDS**.edsnfo(pin_ndx).elec_spec)

5.3.1 Load Circuit Considerations

The EIA-567 electrical view package employs the use of "classes" to define load circuits to be applied to a model. While this provides a viable approach to the documentation of these circuits, it is limited by the fact that it will not support machine interpretation of the circuit unless a standard load configuration is adopted. For the purpose of this document, the "load class" approach identified through the EIA-567 packages has been adopted for the time being. Additional investigation is being performed to determine if a more standardized approach is in order.

5.4 Removing State Limitations on the Input Driver

5.4.1 The Original Driver

With a quick look at the input driver of chapter 4, it may be noted that the output of this device is restricted to the X01 subtype. This has several ramifications on the remaining driver and checker components.

5.4.1.1 On one hand, this technique allows the use of the RISING_EDGE and FALLING_EDGE functions (from the STD_LOGIC_1164 package) in the output driver since these functions only work for '1' to '0' and '0' to '1' transitions.

5.4.1.2 On the other hand, this feature does restrict the STD_LOGIC states which can be used at the model inputs.

5.4.2 An Alternative Driver

The input driver of Figure 5.4.2-1 performs the same general function as the driver presented in chapter 4. The differences occur in two areas. First, the driver will follow any STD_LOGIC state applied to it. Second, the driver does not feature an initialization generic ("pull_value"), but rather assigns the output of the model to 'U' as required by EIA-567, paragraph 3.4.2.1.3.

```
--*****************************************************************
--    (c) Copyright 1994 by the
--          Naval Air Warfare Center, Aircraft Division, Indianapolis
--    Source
--          Author(s):              Charles K. Rogers
--          Organization:             NAWC-ADI
--                                  Code 306, MS-42
--                                  6000 E 21st St
--                                  Indianapolis,  IN  46219-2189
--                                  Phone:   317-353-3579
--                                  EMail:   ROGERSC1@po2.nawc-ad-indy.navy.mil
--          Reference:              MIL-M-28787/212
--          Project:                SHARP   TIREP
--    DESC Certification
--          Status:                 TBD
--*****************************************************************
--    Revision History
--          Version:                1.0
--                Date:             23 May 1994
--                Comments:         Original Release
--*****************************************************************
--    Module Description
--          File:                   i_drive.vhd
--          Module  Name(s):        i_driver entity/arch_i_driver_beh architecture
```

Figure 5.4.2-1. A Generic Input Driver.

```
--        Constraints:          none
--        Limitations:          none
--        I/O Format(s):        std_logic
--        Purpose and Use:      This VHDL module describes an input driver used
--              to implement input wire delays when used in conjunction with the
--              EIA-567 packages.
--        Notes:                none
--*****************************************************************
--    Standard Libraries/Packages
--        std_logic_1164        standard multi-value logic package
--    Associated Packages (order of analysis implied)
--        eia_567               standard eia package
--    Component Models
--        none
--    Platform:                 486/33MHz PC
--    Software/Version:         V-System for Windows, Version 3.0
--*****************************************************************
--    Design Specification Elements
--        none
--*****************************************************************
LIBRARY ieee;
LIBRARY eia;
USE ieee.STD_LOGIC_1164.ALL;
USE eia.EIA_567.ALL;
ENTITY i_driver IS
    GENERIC(win_d: TIME;
        pin_data: EV_SIGNAL_LIMIT);
    PORT(a: IN STD_LOGIC;
        y: OUT STD_LOGIC:='U');
    END i_driver;
ARCHITECTURE arch_i_driver_beh OF i_driver IS
    BEGIN
    PROCESS(a)
        BEGIN
        y<=a   AFTER win_d;
        END PROCESS;
    END  arch_i_driver_beh;
```

Figure 5.4.2-1 continued. A Generic Input Driver.

5.5 Dealing with the "m_gen" Variable

As noted in the previous chapter, there is some concern over the manner in which the "m_gen" variable is handled in the synchronous and asynchronous drivers when compared to the manner in which it is defined. For clarification, the following definition is presented for "m_gen". This is the definition which will be used by the synchronous and asynchronous checkers which appear in Figures 5.5-1 and 5.5-2.

> Timing violations tested by assertion statements will be reported whenever the severity level of the associated assertion test is greater than or equal to the "m_gen" variable.

5.5.1 Severity Levels

```
--*****************************************************************
--  (c) Copyright 1994 by the
--      Naval Air Warfare Center, Aircraft Division, Indianapolis
--  Source
--      Author(s):          Charles K. Rogers
--      Organization:         NAWC-ADI
--                          Code 306, MS-42
--                          6000 E 21st St
--                          Indianapolis,  IN  46219-2189
--                          Phone:   317-353-3579
--                          EMail:   ROGERSC1@po2.nawc-ad-indy.navy.mil
--      Reference:          MIL-M-28787/212
--      Project:            SHARP   TIREP
--  DESC Certification
--      Status:             TBD
--*****************************************************************
--  Revision History
--      Version:            1.0
--          Date:           23 May 1994
--          Comments:       Original Release
--*****************************************************************
--  Module Description
--      File:               sync_c.vhd
--      Module  Name(s):    sync_checkerentity/arch_sync_checker_beh
--          architecture
--      Constraints:        none
--      Limitations:        none
--      I/O  Format(s):     std_logic
--      Purpose and Use:    This VHDL module describes a synchronous checker
--          which is used to perform setup and hold timing checks on a data
--          signal with respect to a clock edge.
--      Notes:              none
--*****************************************************************
--  Standard Libraries/Packages
--      std_logic_1164      standard multi-value logic package
--  Associated Packages (order of analysis implied)
--      eia_567              standard timing view package
--  Component  Models
--      none
--  Platform:               486/33MHz PC
--  Software/Version:       V-System for Windows, Version 3.0
--*****************************************************************
--  Design Specification Elements
--      none
--*****************************************************************
LIBRARY ieee;
LIBRARY eia;
USE ieee.STD_LOGIC_1164.ALL;
USE eia.EIA_567.ALL;
```

Figure 5.5-1. A Synchronous Checker.

```
ENTITY sync_checker IS
    GENERIC(x_gen:BOOLEAN:= TRUE;
        m_gen:SEVERITY_LEVEL:= WARNING;
        syncconstraint: SYNC);
    PORT(clk,data_in: IN STD_LOGIC;
        data_out: OUT STD_LOGIC:='U');
    END sync_checker;
ARCHITECTURE arch_sync_checker_beh OF sync_checker IS
    BEGIN
    PROCESS(clk,data_in)
        VARIABLE ERROR:BOOLEAN:= FALSE;
        VARIABLE clkev,dataev:TIME:=0 ns;
        VARIABLE se:SIGNAL_EDGE;
        VARIABLE sevlev:SEVERITY_LEVEL:= WARNING;
        BEGIN
        IF clk'EVENT THEN se:=EDGE_CHECK(clk);
            IF syncconstraint.edge=se THEN
-- If m_gen is greater than sevlev, then skip this check
                IF m_gen<=sevlev THEN
-- Check setup time
                    ASSERT NOW=0 ns OR now-dataev>=syncconstraint.setup
                        REPORT "setup violation"
                        SEVERITY WARNING;
                    END IF;
                IF now-dataev<syncconstraint.setup THEN ERROR:= TRUE;
                    END IF;
                clkev:= NOW;
                END IF;
            END IF;
        IF data_in'EVENT THEN
-- If m_gen is greater than sevlev, then skip this check
            IF m_gen<=sevlev THEN
-- Check hold time
                ASSERT NOW=0 ns OR now-clkev>=syncconstraint.hold
                    REPORT "hold violation"
                    SEVERITY WARNING;
                END IF;
            IF now-clkev<syncconstraint.hold THEN ERROR:= TRUE;
                END IF;
            dataev:= NOW;
            END IF;
-- If x_gen is true then output 'X' when violation occurs.
-- Otherwise, output the input.
        IF ERROR AND x_gen THEN data_out<='X';ERROR:= FALSE;
            ELSE data_out<=data_in;
            END IF;
        END PROCESS;
    END arch_sync_checker_beh;
```

Figure 5.5-1 continued. A Synchronous Checker.

```
--******************************************************************
--   (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:            EIA-567
--        Project:              SHARP   TIREP
--   DESC Certification
--        Status:               TBD
--******************************************************************
--   Revision History
--        Version:              1.1
--             Date:            23 September 1994
--             Comments:        Added cycle time checks
--        Version:              1.0
--             Date:            23 May 1994
--             Comments:        Original Release
--******************************************************************
--   Module Description
--        File:                 async_c.vhd
--        Module  Name(s):      async_checkerentity/arch_async_checker_beh
--             architecture
--        Constraints:          none
--        Limitations:          none
--        I/O  Format(s):       std_logic
--        Purpose and Use:      This VHDL module describes an asynchronous
--             checker which is used to perform minimum and maximum pulse width
--             checks.
--        Notes:                none
--******************************************************************
--   Standard Libraries/Packages
--        std_logic_1164        standard multi-value logic package
--   Associated Packages (order of analysis implied)
--        eia_567               standard eia package
--   Component  Models
--        none
--   Platform:                  486/33MHz PC
--   Software/Version:          V-System for Windows, Version 3.0
--******************************************************************
--   Design Specification Elements
--        none
--******************************************************************
LIBRARY ieee;
```

Figure 5.5-2. An Asynchronous Checker.

```
LIBRARY eia;
USE ieee.STD_LOGIC_1164.ALL;
USE eia.EIA_567.ALL;
ENTITY async_checker IS
    GENERIC(x_gen:BOOLEAN:= TRUE;
        m_gen:SEVERITY_LEVEL:= WARNING;
        asyncconstraint: ASYNC);
    PORT(data_in: IN STD_LOGIC;
        data_out: OUT STD_LOGIC:='U');
    END async_checker;
ARCHITECTURE arch_async_checker_beh OF async_checker IS
    BEGIN
    PROCESS(data_in)
        VARIABLE lstrise,lstfall:TIME:=0 ns;
        VARIABLE se:SIGNAL_EDGE;
        VARIABLE ERROR:BOOLEAN:= FALSE;
        VARIABLE sevlev:SEVERITY_LEVEL:= WARNING;
        BEGIN
        se:= EDGE_CHECK(data_in);
-- If m_gen is greater than sevlev, then skip these checks
        IF m_gen<=sevlev THEN
            IF se=e01 OR se=el1 OR se=elh OR se=e0h THEN
-- Check minimum low pulse width
                IF asyncconstraint.t0min>0 ns THEN
                    ASSERT NOW=0 ns OR now-lstfall>=asyncconstraint.t0min
                        REPORT "minimum LOW pulse WIDTH violation"
                        SEVERITY sevlev;
                    END IF;
-- Check maximum low pulse width
                IF asyncconstraint.t0max>0 ns THEN
                    ASSERT NOW=0 ns OR now-lstfall<=asyncconstraint.t0max
                        REPORT "maximum LOW pulse WIDTH violation"
                        SEVERITY sevlev;
                    END IF;
-- Check minimum cycle time
                IF asyncconstraint.cycmin>0 ns THEN
                    ASSERT NOW=0 ns OR now-lstrise>=asyncconstraint.cycmin
                        REPORT "minimum cycle (maximum frequency) violation"
                        SEVERITY sevlev;
                    END IF;
-- Check maximum cycle time
                IF asyncconstraint.cycmax>0 ns THEN
                    ASSERT NOW=0 ns OR now-lstrise<=asyncconstraint.cycmax
                        REPORT "maximum cycle (minimum frequency) violation"
                        SEVERITY sevlev;
                    END IF;
                ELSIF se=e10 OR se=eh0 OR se=ehl OR se=e11 THEN
-- Check minimum high pulse width
                IF asyncconstraint.t1min>0 ns THEN
```

Figure 5.5-2 continued. An Asynchronous Checker.

```
                    ASSERT NOW=0 ns OR  now-lstrise>=asyncconstraint.t1min
                        REPORT "minimum HIGH pulse WIDTH violation"
                        SEVERITY sevlev;
                    END IF;
-- Check maximum high pulse width
                IF asyncconstraint.t1max>0 ns THEN
                    ASSERT NOW=0 ns OR  now-lstrise<=asyncconstraint.t1max
                        REPORT "maximum HIGH pulse WIDTH violation"
                        SEVERITY sevlev;
                    END IF;
                END IF;
            END IF;
-- Log low pulse width violations
        IF se=e01 OR se=el1 OR se=elh OR se=e0h THEN
            IF asyncconstraint.t0min>0 ns AND
                    now-lstfall<asyncconstraint.t0min  THEN
                    ERROR:= TRUE;
                ELSIF asyncconstraint.t0max>0 ns AND
                    now-lstfall>asyncconstraint.t0max  THEN
                    ERROR:= TRUE;
-- Log cycle time violations
                ELSIF asyncconstraint.cycmin>0 ns AND
                    now-lstrise<asyncconstraint.cycmin  THEN
                    ERROR:= TRUE;
                ELSIF asyncconstraint.cycmax>0 ns AND
                    now-lstrise>asyncconstraint.cycmax  THEN
                    ERROR:= TRUE;
                END IF;
            lstrise:= NOW;
            END IF;
-- Log high pulse width violations
        IF se=e10 OR se=eh0 OR se=ehl OR se=e11 THEN
            IF asyncconstraint.t1min>0 ns AND
                    now-lstrise<asyncconstraint.t1min  THEN
                    ERROR:= TRUE;
                ELSIF asyncconstraint.t1max>0 ns AND
                    now-lstrise>asyncconstraint.t1max  THEN
                    ERROR:= TRUE;
                END IF;
            lstfall:= NOW;
            END IF;
-- If x_gen is true then output 'X' when violation occurs.
-- Otherwise, output the input.
        IF ERROR AND x_gen THEN  data_out<='X';ERROR:= FALSE;
            ELSE  data_out<=data_in;
            END IF;
        END PROCESS;
    END arch_async_checker_beh;
```

Figure 5.5-2 continued. An Asynchronous Checker.

For these checkers, a severity level of WARNING is attached to each timing violation test. Therefore, a "m_gen" value of either NOTE or WARNING will cause timing violations to be reported. A "m_gen" value of either ERROR or FAILURE will cause timing violations to go unreported.

5.5.2 Signal Transitions

In the synchronous checker, a single clock edge or transition type must be specified. As it is currently written, this checker is setup to operate on "e01" and "e10" edges due to the fact that the TO_X01 function is employed on the clock. This means that the checker will view a 'L'-to-'H', '0'-to-'H' or 'L'-to-'1' transition the same as a '0'-to-'1' transition.

5.5.3 Output Limitations

In a similar fashion, the asynchronous checker employs the TO_X01 function on the input data. This permits the checker to view a 'L' state as '0' and a 'H' state as '1' when checking pulse widths. This also means that a "e1h" signal edge on "data_in" will not cause a pulse width test to be performed by the component.

5.5.4 Cycle Checking

In the original asynchronous checker, only minimum and maximum pulse widths were checked. In the model depicted herein, a feature has been added which provides for cycle time (or frequency) checking. In this case, the declaration for the data type ASYNC has been altered to include "cycmin" and "cycmax" for the minimum and maximum cycle times (or maximum and minimum frequencies respectively).

```
TYPE ASYNC IS RECORD
    t1min,t0min,t1max,t0max,cycmin,cycmax:TIME;
    END RECORD;
```

5.6 The GET_TIMING Function

In the model example presented in chapter 4, the associations of Table 5.6-1 were made in the EDS. It is primarily in CMOS technologies where this association is established. In bipolar technologies, the association is not nearly as clear. As a result, concern has been raised over the general application of these associations in the EDS models.

5.6.1 The Original Function

The GET_TIMING function as defined in the EIA-567 toolbox package in chapter 4, looks for an exact match of all operating point parameters (delay, voltage and temperature). If not found, this function will generate an error and default to the nominal operating point (the first point in the EDS).

Delay	*Voltage*	*Temperature*
tmin	*high*	*low*
tnom	*nominal*	*room*
tmax	*low*	*high*

Table 5.6.1-1. EDS Operating Points.

5.6.2 An Alternative Approach

For the TIREP project, it was felt that the delay (operating_point.selection) value would be a better entity to base the operating point selection from the EDS on. As a result, the GET_TIMING function was altered as shown below. Note, this function still looks for an exact match of the operating point. However, if an exact match is not found, the function reverts to the "operating_point.selection" or delay for selection of the operating point. It may be noted that an assertion message is printed when this occurs to notify the user of the selection.

```
----------------------------------------------------------------------
-- Function to find the user selected operating point
----------------------------------------------------------------------
    FUNCTION GET_TIMING(op_point: POINT) RETURN OPNT IS
```

```
        VARIABLE found: BOOLEAN := FALSE;
        VARIABLE temppoint: OPNT;
        BEGIN
---------------------------------------------------------------------
-- first scan operating points stored for an exact match
---------------------------------------------------------------------
        FOR index IN EDS'RANGE LOOP
            IF (EDS(index).selpoint=op_point)     THEN
                temppoint:=  EDS(index);
                found:=  TRUE;
                EXIT;
                END IF;
            END LOOP;
---------------------------------------------------------------------
-- if not found, additional scans for a "close" match can be inserted
-- here
---------------------------------------------------------------------
        IF NOT found THEN
            FOR index IN EDS'RANGE LOOP
                IF (EDS(index).selpoint.selection=op_point.selection)     THEN
                    temppoint:=  EDS(index);
                    EXIT;
                    END IF;
                END LOOP;
            ASSERT FALSE REPORT "DELAYS based upon selection GENERIC."
                SEVERITY NOTE;
            END IF;
        RETURN temppoint;
        END GET_TIMING;
```

5.6.3 Adding "x_gen" Flexibility

It may be noted that the original GET_TIMING function also included a feature which zeroed out the synchronous and asynchronous constraints if the "x_gen" variable was false. This will effectively eliminate all constraint checking whenever "x_gen" is false. The TIREP team felt that the flexibility offered by having independent control over the "x_gen" and "m_gen" functions was desirable. Hence this portion of the GET_TIMING function was deleted.

5.7 Pin and Signal Corrolation

In the EIA-567 modeling approach, the INTERFACE_CONNECTIONS constant declaration was found in the EIA-567 physical view package. For the purposes of printed wiring board layout work, this is not deemed to be the best location for this declaration. By placing this constant declaration in the component model (in this case, the EDS model), the searching required for this information is greatly simplified. Following is the INTERFACE_CONNECTIONS constant which was declared for the DQD module example which will be used later in this chapter.

```
    CONSTANT INTERFACE_CONNECTIONS: PIN_AND_SIGNAL_CORROLATION := (
        andx=> (pin_id=> "34          ",signal_name=> "a          "),
        bndx=> (pin_id=> "8           ",signal_name=> "b          "),
        cndx=> (pin_id=> "13          ",signal_name=> "c          "),
        dndx=> (pin_id=> "12          ",signal_name=> "d          "),
        endx=> (pin_id=> "32          ",signal_name=> "e          "),
        fndx=> (pin_id=> "36          ",signal_name=> "f          "),
        gndx=> (pin_id=> "22          ",signal_name=> "g          "),
        d32ndx=> (pin_id=> "19          ",signal_name=> "d32        "),
        d33ndx=> (pin_id=> "39          ",signal_name=> "d33        "),
        sow_sndx=> (pin_id=> "14          ",signal_name=> "sow s      "),
```

```
p_undx= > (pin_id= > "4              ",signal_name= > "p.u.               "),
ldlndx= > (pin_id= > "7              ",signal_name= > "ldl                "),
not_ovd_sndx= > (pin_id= > "15       ",signal_name= > "not ovd s          "),
dta_clk_sndx= > (pin_id= > "18       ",signal_name= > "dta clk s          "),
clkndx= > (pin_id= > "24             ",signal_name= > "clk                "),
ena_0ndx= > (pin_id= > "25           ",signal_name= > "ena 0              "),
not_eocndx= > (pin_id= > "27         ",signal_name= > "not eoc            "),
not_mclr_sndx= > (pin_id= > "29      ",signal_name= > "not mclr s         "),
cnt_1ndx= > (pin_id= > "33           ",signal_name= > "cnt 2              "),
cnt_2ndx= > (pin_id= > "30           ",signal_name= > "cnt 1              "),
sdindx= > (pin_id= > "40             ",signal_name= > "sdi                "),
not_stb_sndx= > (pin_id= > "9        ",signal_name= > "not stb s          "),
actndx= > (pin_id= > "2              ",signal_name= > "act                "),
not_actndx= > (pin_id= > "37         ",signal_name= > "not act            "),
cnt_0ndx= > (pin_id= > "5            ",signal_name= > "cnt 0              "),
dta_0_sndx= > (pin_id= > "11         ",signal_name= > "dta 0 s            "),
dta_1_sndx= > (pin_id= > "31         ",signal_name= > "dta 1 s            "),
srmndx= > (pin_id= > "16             ",signal_name= > "srm                "),
urc_sndx= > (pin_id= > "20           ",signal_name= > "urc s              "),
not_lrc_sndx= > (pin_id= > "26       ",signal_name= > "not lrc s          "),
clk1ndx= > (pin_id= > "38            ",signal_name= > "clk1               "),
ldl1ndx= > (pin_id= > "3             ",signal_name= > "ldl1               "),
rdy_sndx= > (pin_id= > "6            ",signal_name= > "rdy s              "),
not_dta_0_rzndx= > (pin_id= > "23    ",signal_name= > "not dta 0 rz       "),
not_dta_1_rzndx= > (pin_id= > "17    ",signal_name= > "not dta 1 rz       "),
stb_csndx= > (pin_id= > "21          ",signal_name= > "stb cs             "),
ovr_0ndx= > (pin_id= > "28           ",signal_name= > "ovr 0              "),
not_eoc1ndx= > (pin_id= > "35        ",signal_name= > "not eoc1           "),
vccndx= > (pin_id= > "1              ",signal_name= > "vcc                "),
gndndx= > (pin_id= > "10             ",signal_name= > "gnd                "));
```

5.8 Modeling Hierarchy

5.8.1 Limitations of the EIA-567 Approach

The EIA-567 modeling approach as presented in chapter 4 is very good for developing a DID compliant model for the first time. This approach enables the design engineer to become acquainted with the model development in a timely and cost effective manner by breaking the model into manageable blocks (packages). When it comes to modeling a design hierarchy (i.e. components/modules/systems), this approach results in a tremendous amount of duplicated information for each model developed.

5.8.1.1 The only package developed which can be placed in a library for use by all design units is the EIA_567_EV package. All other packages must be analyzed for each model. Additionally, each of the generic drivers and checkers must be analyzed for each model due to the placement of functions upon which these components are dependent.

5.8.1.2 In the sections which follow, a variation on the EIA-567 modeling approach will be presented which is an attempt to maximize the number of general purpose functions and components which can be placed into a library, thus minimizing the amount of duplicated code for each model.

5.8.2 An Approach to Hierarchy

The first task performed will be to take the four EIA-567 standard packages (i.e. EV, TV, PV and toolbox), and extract all design independent functions and type declarations for insertion into an EIA_567 library. With this done, the package of Figure 5.8.2-1 was derived. This package contains the declarations identified in Table 5.8.2-1. Each of these declarations is described in chapter 4

Data Type	Source Package
RESISTANCE	EIA_567_EV
CAPACITANCE	EIA_567_EV
TEMPERATURE	EIA_567_EV
VOLTAGE	EIA_567_EV
CURRENT	EIA_567_EV
POWER	EIA_567_EV
OPERATING_SELECTION	EIA_567_EV
EV_SIGNAL_LIMIT	EIA_567_EV
EV_SIGNAL_LIMITS	EIA_567_EV
DELAY	EIA_567_TV
OUTPUT_DELAYS	EIA_567_TV
ASYNC	EIA_567_TV
ASYNC_CONSTRAINTS	EIA_567_TV
SIGNAL_EDGE	EIA_567_TV
SYNC	EIA_567_TV
SYNC_CONSTRAINTS	EIA_567_TV
ESD_CLASS	EIA_567_PV
EIA_STRING	EIA_567_PV
PIN_RECORD	EIA_567_PV

Subprogram	Source Package
FUNCTION CONVERT_TO_UX01Z (6)	EIA_567_EV
FUNCTION DETECT_EDGE	EIA_567_TV
FUNCTION MAX	EIA_567_TB
FUNCTION F	EIA_567_TB
FUNCTION EDGE_CHECK	EIA_567_TB

Table 5.8.2-1. EIA-567 Package Declarations.

5.8.2.1 With this accomplished, it can be determined that all of the functions and data types necessary to support the generic driver and checker components are now available in the EIA-567 library. This allows the following drivers and checkers to be included in the EIA-567 library.

i_driver
o_driver
b_driver
asynchronous_checker
synchronous_checker

5.8.2.2 All of these components have been previously declared with the exception of the bidirectional driver (b_driver) which is included as Figure 5.8.2.2-1. It should be noted that this component is simply the combination o f the input and output driver components. At the time of this writing, this component has not been validated.

5.8.3 The Design Specification Package

With an EIA-567 library now established, the task remains to assemble the remaining design specific data types and functions into what have been termed "design specification" (or DS) packages.

5.8.3.1 In order to develop the design specification packages, it is necessary to have a design to work with. Figure 5.8.3.1-1 contains a core model for the SEM DQD module. This module is called a standard digital serial output module. This model is written in an RTL format which was extracted directly from the schematic diagram for the module.

```
--*****************************************************************
--  (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:            EIA-567
--        Project:              SHARP   TIREP
--  DESC Certification
--        Status:               TBD
--*****************************************************************
--  Revision History
--        Version:              1.1
--              Date:           23 September 1994
--              Comments:       Added cycle time values to async record
--        Version:              1.0
--              Date:           23 May 1994
--              Comments:       Original Release
--*****************************************************************
--  Module Description
--        File:                 eia_567p.vhd
--        Module  Name(s):      eia_567package
--        Constraints:          none
--        Limitations:          none
--        I/O  Format(s):       none
--        Purpose and Use:      This package is based upon elements of the
--              original eia_567_ev, pv, tv and toolbox packages. Design
--              independent declarations and functions have been combined in
--              this package to allow this code to be compiled into a library
--              and used in component/assembly models throughout a design.  This
--              package is based upon the packages developed by Len Finegold
--              dated 12/22/92.
--        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
```

Figure 5.8.2-1. An EIA-567 Library.

```
--        none
--*************************************************************
LIBRARY ieee;
USE ieee.STD_LOGIC_1164.ALL;
PACKAGE EIA_567 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;
        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 EIA_STRING IS ARRAY (27 DOWNTO 1) OF CHARACTER;
----------------------------------------------------------------------
-- type conversions
----------------------------------------------------------------------
    FUNCTION CONVERT_TO_UX01Z (s:STD_LOGIC_VECTOR)
        RETURN STD_LOGIC_VECTOR;
```

Figure 5.8.2-1 continued. An EIA-567 Library.

```
    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 o fthe operating envelope.
----------------------------------------------------------------------
    TYPE OPERATING_SELECTION IS (TMIN,TNOM,TMAX,tzero);
----------------------------------------------------------------------
-- 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;
----------------------------------------------------------------------
-- a collection of the individual electrical limits
----------------------------------------------------------------------
    TYPE EV_SIGNAL_LIMITS IS ARRAY(NATURAL RANGE    <>)
        OF EV_SIGNAL_LIMIT;
----------------------------------------------------------------------
--  declaration of the delay transitions as a record of information
----------------------------------------------------------------------
    TYPE DELAY IS RECORD
        tlh,thl,tlz,thz,tzl,tzh,ttlh,tthl: TIME;
        END RECORD;
----------------------------------------------------------------------
--  a collection of individual delays
----------------------------------------------------------------------
    TYPE OUTPUT_DELAYS IS ARRAY(NATURAL RANGE  <>)  OF DELAY;
----------------------------------------------------------------------
--  declaration of the asynchronous constraint record of information
----------------------------------------------------------------------
    TYPE ASYNC IS RECORD
```

Figure 5.8.2-1 continued. An EIA-567 Library.

```
        t1min,t0min,t1max,t0max,cycmin,cycmax: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;
----------------------------------------------------------------------
-- edge detection functions checks if there is a transition of a
-- signal_edge type
----------------------------------------------------------------------
    FUNCTION DETECT_EDGE (SIGNAL s:STD_ULOGIC; edge: SIGNAL_EDGE)
        RETURN BOOLEAN;
----------------------------------------------------------------------
-- esd specifications, mil-std-1686
----------------------------------------------------------------------
    TYPE ESD_CLASS IS (UNKNOWN, i, ii, iii);
----------------------------------------------------------------------
--   pinout data
----------------------------------------------------------------------
    TYPE PIN_RECORD IS RECORD
        pin_id: EIA_STRING;
        signal_name: EIA_STRING;
        END RECORD;
----------------------------------------------------------------------
-- function which returns the maximum of two times
```

Figure 5.8.2-1 continued. An EIA-567 Library.

```
---------------------------------------------------------------------
    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 detects the type of transition a signal makes and
-- returns the value of type signal_edge
---------------------------------------------------------------------
    FUNCTION EDGE_CHECK(SIGNAL s:STD_LOGIC) RETURN SIGNAL_EDGE;
    END EIA_567;

PACKAGE BODY EIA_567 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
```

Figure 5.8.2-1 continued. An EIA-567 Library.

```
        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
        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;
------------------------------------------------------------------------
-- 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');
```

Figure 5.8.2-1 continued. An EIA-567 Library.

```
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');
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');
```

Figure 5.8.2-1 continued. An EIA-567 Library.

```
            WHEN    ehl=>RETURN ((s'EVENT AND s='L') AND s'last_value='H');
            WHEN OTHERS => RETURN FALSE;
            END CASE;
        END DETECT_EDGE;
---------------------------------------------------------------------
-- Function to return the maximum of two times
---------------------------------------------------------------------
    FUNCTION MAX(a,b: TIME) RETURN TIME IS
        BEGIN
        IF b>a THEN RETURN b;
            ELSE RETURN a;
            END IF;
        END MAX;
---------------------------------------------------------------------
-- Function to return the appropriate delay based upon the identified
-- signal edge
---------------------------------------------------------------------
FUNCTION F(d: DELAY;SIGNAL old, soon_to_be:STD_LOGIC) RETURN TIME IS
        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 '0' | 'L' => RETURN (0 ns);
                WHEN '1' | 'H' => RETURN (d.tlh);
                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 '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.tzl,d.tzh));
                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));
```

Figure 5.8.2-1 continued. An EIA-567 Library.

```
            WHEN 'U' | 'X' | '-' => RETURN (0 ns);
            WHEN 'W' => RETURN (MAX(d.thz,d.tlz));
            END CASE;
        END CASE;
    END F;
-----------------------------------------------------------------------
-- Function to determine the type of a signal edge
-----------------------------------------------------------------------
FUNCTION EDGE_CHECK(SIGNAL s:STD_LOGIC) RETURN SIGNAL_EDGE IS
    BEGIN
    RETURN (SIGNAL_EDGE'val((STD_ULOGIC'pos(s'last_value)*9)
        +( STD_ULOGIC'pos(s))));
    END EDGE_CHECK;
END EIA_567;
```

Figure 5.8.2-1 continued. An EIA-567 Library.

5.8.3.2 With this done, the design specification package for the module is developed. This package contains the design characteristics which were originally defined in the "design_EV", "design_TV" and the "design_PV" packages. Table 5.8.3.2-1 contains a list of declarations which are now made in the "design_DS" package along with a reference to the package in which they were originally found. Additional information on these declarations can be found in chapter 4. The resulting package for the DQD module example is provided as Figure 5.8.3.2-1.

5.8.3.2.1 In this package, it may be noted that there is much more diversity among the input and output pins than was evidenced in the FBG design example. The load circuits described in this package are inferred as they are not actually imposed in the specification (MIL-M-28787/212).

5.8.3.2.2 Additionally, this model does feature some synchronous and asynchronous timing constraints which are specified within this package.

5.8.3.2.3 It may be noted from this listing that there are several elements which were originally declared in the standard EIA-567 packages (rather than the design specific packages). Inclusion of these elements in this package was necessary in order to maintain the proper order of analysis without adding another package.

```
--**************************************************************
--  (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:          EIA-567
--      Project:            SHARP  TIREP
--  DESC Certification
```

Figure 5.8.2.2-1. A Generic Bidirectional Driver.

```
--      Status:              TBD
--*****************************************************************
--   Revision History
--      Version:             1.1
--         Date:             23 September 1994
--         Comments:         Corrected bug in transition time model check
--      Version:             1.0
--         Date:             23 May 1994
--         Comments:         Original Release
--*****************************************************************
--   Module Description
--      File:                b_drive.vhd
--      Module  Name(s):     b_driver entity/arch_b_driver_beh architecture
--      Constraints:         none
--      Limitations:         none
--      I/O  Format(s):      std_logic
--      Purpose and Use:     This VHDL module describes a bidirectional
--         driver used to implement propagation and input wire delays along
--         with output transition times when used in conjunction with the
--         EIA-567 packages.
--      Notes:               none
--*****************************************************************
--   Standard Libraries/Packages
--      std_logic_1164       standard multi-value logic package
--   Associated Packages (order of analysis implied)
--      eia_567              standard eia package
--   Component  Models
--      none
--   Platform:               486/33MHz PC
--   Software/Version:       V-System for Windows, Version 3.0
--*****************************************************************
--   Design Specification Elements
--      none
--*****************************************************************
LIBRARY ieee;
LIBRARY eia;
USE ieee.STD_LOGIC_1164.ALL;
USE eia.EIA_567.ALL;
ENTITY b_driver IS
   GENERIC(delay_y:DELAY;
      win_d,wir_o:TIME;
      pin_data: EV_SIGNAL_LIMIT;
      m_gen:SEVERITY_LEVEL;
      x_gen:BOOLEAN);
   PORT(a:IN STD_LOGIC;
      d:OUT STD_LOGIC;
      y:INOUT STD_LOGIC:='U');
   END b_driver;
-- Note:  This architecture does not provide a resolution function
```

Figure 5.8.2.2-1 continued. A Bidirectional Driver.

```
-- for this device if it is also being driven externally.
ARCHITECTURE arch_b_driver_beh OF b_driver IS
    BEGIN
-- Output state determination
    PROCESS(a)
        VARIABLE del,del1,del2:TIME;
        BEGIN
        IF y/=a THEN
            del:= F(delay_y,y,a);
-- variables used to check for timing violations
            IF y/='Z' AND a/='Z' THEN
                del1:=del-delay_y.tthl/2;
                del2:=del-delay_y.ttlh/2;
                ELSE del1:=0 ns;del2:=0 ns;
                END IF;
            IF   m_gen<=WARNING THEN
-- check for ttlh and/or tthl 2*del timing violation
                ASSERT NOW=0 ns OR (del1>=0 ns AND del2>=0 ns)
                    REPORT "OUTPUT transition TIME ERROR"
                    SEVERITY WARNING;
                END IF;
            ELSE del:=0 ns;del1:=0 ns;del2:=0 ns;
            END IF;
        IF x_gen AND (delay_y.tthl>0 ns) AND (delay_y.ttlh>0 ns) THEN
-- if x_gen is true then perform transition time simulation on
-- low-to-high and high-to-low transitions
            IF (y='1' OR y='H') AND (a='0' OR a='L') THEN
                IF del1<0 ns THEN del1:=0 ns;
                    END IF;
                y<='X'  AFTER del1+wir_o,
                    a AFTER del+delay_y.tthl/2;
                ELSIF (y='0' OR y='L') AND (a='1' OR a='H') THEN
                IF del2<0 ns THEN del2:=0 ns;
                    END IF;
                y<='X'  AFTER del2+wir_o,
                    a AFTER del+delay_y.ttlh/2;
-- for all other transitions, disregard the transition time
                ELSE y<=a AFTER del+wir_o;
                END IF;
-- if x_gen is false then skip transition simulation
            ELSE y<=a AFTER del+wir_o;
            END IF;
        END PROCESS;
-- Input state determination
    PROCESS(y)
        BEGIN
        d<=y   AFTER win_d;
        END PROCESS;
    END arch_b_driver_beh;
```

Figure 5.8.2.2-1 continued. A Bidirectional Driver.

```
--*****************************************************************
--  (c) Copyright 1994 by the
--        Naval Air Warfare Center, Aircraft Division, Indianapolis
--  Source
--        Author(s):            Charles K. Rogers
--        Organization:           NAWC-ADI
--                              Code 306, MS-42
--                              6000 E 21st St
--                              Indianapolis,  IN  46219-2189
--                              Phone:   317-353-3579
--                              EMail:   ROGERSC1@po2.nawc-ad-indy.navy.mil
--        Reference:            MIL-M-28787/212
--        Project:              SHARP   TIREP
--  DESC Certification
--        Status:               TBD
--*****************************************************************
--  Revision History
--        Version:              1.0
--              Date:           20 May 1994
--              Comments:       Original Release
--*****************************************************************
--  Module Description
--        File:                 dqd_cor.vhd
--        Module  Name(s):      dqd_core entity/arch_dqd_core_beh architecture
--        Constraints:          none
--        Limitations:          none
--        I/O  Format(s):       std_logic
--        Purpose and Use:      This VHDL module contains the behavorial
--              description for the DQD standard hardware module.          This is also
--              the description for the semi-custom microcircuit which is used
--              in this module.   This is a synthesizable model.
--        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
--        none
--*****************************************************************
LIBRARY ieee;
USE ieee.STD_LOGIC_1164.ALL;
ENTITY dqd_core  IS
    PORT(a,b,c,d,e,f,g,d32,d33,sow_s,p_u,ldl,not_ovd_s,dta_clk_s:  IN STD_LOGIC;
```

Figure 5.8.3.1-1. The DQD Core Model.

```
            clk,ena_0,not_eoc,not_mclr_s,cnt_1,cnt_2,sdi,not_stb_s:  IN STD_LOGIC;
            act,not_act,cnt_0,dta_0_s,dta_1_s,srm,urc_s,not_lrc_s,clk1:  OUT STD_LOGIC;
            ldl1,rdy_s,not_dta_0_rz,not_dta_1_rz,stb_cs,ovr_0,not_eoc1:  OUT STD_LOGIC);
        END  dqd_core;
    ARCHITECTURE arch_dqd_core_beh  OF  dqd_core  IS
        FUNCTION  to_integer(INPUT:STD_LOGIC_VECTOR) RETURN INTEGER IS
            VARIABLE result:INTEGER;
            VARIABLE weight:INTEGER;
            BEGIN
            weight:=1;result:=0;
            FOR i IN INPUT'LOW TO INPUT'HIGH LOOP
                IF INPUT(i)='1'  THEN
                    result:=result+weight;
                    ELSIF INPUT(i)/='0'  THEN RETURN -1;
                    END IF;
                weight := weight * 2;
                END LOOP;
            RETURN result;
            END  to_integer;
        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 n0,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15:STD_LOGIC;
        SIGNAL n16,n17,n18,n19,n20,n21,n22,n23,n24,n25:STD_LOGIC;
        SIGNAL n26,n27,n28,n29,n30,n40,n50:STD_LOGIC;
        SIGNAL ij1,ij2,ij3,ij4,ij5,ij6,ij7,ij8:STD_LOGIC;
        SIGNAL ik1,ik2,ik3,ik4,ik5,ik6,ik7,ik8:STD_LOGIC;
        SIGNAL   iact,inot_act,icnt_0,idta_0_s,idta_1_s,isrm,iurc_s,inot_lrc_s,iclk1:STD_LOGIC;
        SIGNAL buf1,buf2:INTEGER RANGE -1 TO 15;
        SIGNAL  din1,din2,dout1,dout2:STD_LOGIC_VECTOR(3 DOWNTO 0);
        BEGIN
    -- output signal assignment
        act<=iact;not_act<=inot_act;cnt_0<=icnt_0;dta_0_s<=idta_0_s;
```

Figure 5.8.3.1-1 continued. The DQD Core Model.

```
        clk1<=iclk1;dta_1_s<=idta_1_s;srm<=isrm;urc_s<=iurc_s;
        not_lrc_s<=inot_lrc_s;
-- combinational logic implementations
        n0<='0';
        stb_cs<=(n1 NAND n2) NAND ((n3 AND n4) NAND iclk1);
        not_eoc1<=(n5 AND n6) NAND n5;
        not_dta_1_rz<=n1 NAND idta_1_s;
        not_dta_0_rz<=n1 NAND idta_0_s;
        idta_0_s<=n7 NAND n8;
        idta_1_s<=n7 NAND NOT n8;
        iclk1<=n9 AND dta_clk_s;
        inot_lrc_s<=icnt_0 AND (iclk1 NAND ldl);
        icnt_0<=iclk1 NAND (n7 AND n10);
        ovr_0<=ldl AND iclk1 AND n11;
        isrm<= NOT ldl;
        iurc_s<= NOT not_stb_s;
        ldl1<=n14 AND n15 AND n16;
        rdy_s<=n13 AND n17;
        n1<=iact AND iclk1;
        n9<=not_mclr_s AND not_ovd_s;
        n14<=n12 NAND n13;
        n18<=n3 AND n4;
        n19<= NOT n14 NAND NOT not_eoc;
        n20<=n9 AND not_eoc;
        n22<= NOT n21;
        n21<=sow_s NAND ldl;
        n23<=n17 AND inot_act;
        n24<=n16 AND NOT ldl;
        n25<= NOT ena_0;
        din1(0)<=a;din1(1)<=b;din1(2)<=c;din1(3)<=d;
        din2(0)<=e;din2(1)<=f;din2(2)<=g;din2(3)<=p_u;
-- j-k flip-flop implementations
        PROCESS(ena_0,n25,clk,p_u,not_mclr_s,n3,n12,ij1,ik1)
            BEGIN
            IF not_mclr_s='0' OR p_u='0' THEN ij1<=ena_0;ik1<=n25;
                ELSIF not_mclr_s='1' AND p_u='1' THEN
                IF clk='1' THEN
                    IF (ena_0='1' AND n12='1') OR (n25='1' AND n3='1') THEN
                            ij1<=ena_0;ik1<=n25;
                        ELSE ij1<=ij1;ik1<=ik1;
                        END IF;
                    ELSE ij1<=ena_0;ik1<=n25;
                    END IF;
                ELSE NULL;
                END IF;
            END PROCESS;
        PROCESS(not_mclr_s,p_u,clk)
            BEGIN
            IF not_mclr_s='0' AND p_u='1' THEN n3<='0';n12<='1';
```

Figure 5.8.3.1-1 continued. The DQD Core Model.

```
            ELSIF not_mclr_s='1' AND p_u='0' THEN    n3<='1';n12<='0';
            ELSIF not_mclr_s='0' AND p_u='0' THEN    n3<='1';n12<='1';
            ELSIF not_mclr_s='1' AND p_u='1' THEN
            IF clk'EVENT AND clk='0' THEN
                IF ij1='0' AND ik1='0' THEN     n3<=n3;n12<=n12;
                    ELSIF ij1='1' AND ik1='0' THEN    n3<='1';n12<='0';
                    ELSIF ij1='0' AND ik1='1' THEN    n3<='0';n12<='1';
                    ELSIF ij1='1' AND ik1='1' THEN     n3<=n12;n12<=n3;
                    END IF;
                END IF;
            END IF;
        END PROCESS;
    PROCESS(n0,n23,clk,not_ovd_s,not_mclr_s,n11,n13,ij2,ik2)
        BEGIN
        IF not_mclr_s='0' OR not_ovd_s='0' THEN     ij2<=n0;ik2<=n23;
            ELSIF not_mclr_s='1' AND not_ovd_s='1' THEN
            IF clk='1' THEN
                IF (n0='1' AND n13='1') OR (n23='1' AND n11='1') THEN
                        ij2<=n0;ik2<=n23;
                    ELSE     ij2<=ij2;ik2<=ik2;
                    END IF;
                ELSE     ij2<=n0;ik2<=n23;
                END IF;
            ELSE NULL;
            END IF;
        END PROCESS;
    PROCESS(not_mclr_s,not_ovd_s,clk)
        BEGIN
        IF not_mclr_s='0' AND not_ovd_s='1' THEN    n11<='0';n13<='1';
            ELSIF not_mclr_s='1' AND not_ovd_s='0' THEN    n11<='1';n13<='0';
            ELSIF not_mclr_s='0' AND not_ovd_s='0' THEN    n11<='1';n13<='1';
            ELSIF not_mclr_s='1' AND not_ovd_s='1' THEN
            IF clk'EVENT AND clk='0' THEN
                IF ij2='0' AND ik2='0' THEN     n11<=n11;n13<=n13;
                    ELSIF ij2='1' AND ik2='0' THEN    n11<='1';n13<='0';
                    ELSIF ij2='0' AND ik2='1' THEN    n11<='0';n13<='1';
                    ELSIF ij2='1' AND ik2='1' THEN     n11<=n13;n13<=n11;
                    END IF;
                END IF;
            END IF;
        END PROCESS;
    PROCESS(ldl,n0,clk,not_mclr_s,not_stb_s,n17,n26,ij3,ik3)
        BEGIN
        IF not_stb_s='0' OR not_mclr_s='0' THEN     ij3<=ldl;ik3<=n0;
            ELSIF not_stb_s='1' AND not_mclr_s='1' THEN
            IF clk='1' THEN
                IF (ldl='1' AND n26='1') OR (n0='1' AND n17='1') THEN
                        ij3<=ldl;ik3<=n0;
                    ELSE     ij3<=ij3;ik3<=ik3;
```

Figure 5.8.3.1-1 continued. The DQD Core Model.

```
                    END IF;
                ELSE      ij3<=ldl;ik3<=n0;
                END IF;
            ELSE NULL;
            END IF;
        END PROCESS;
    PROCESS(not_stb_s,not_mclr_s,clk)
        BEGIN
        IF not_stb_s='0' AND not_mclr_s='1' THEN    n17<='0';n26<='1';
            ELSIF not_stb_s='1' AND not_mclr_s='0' THEN    n17<='1';n26<='0';
            ELSIF not_stb_s='0' AND not_mclr_s='0' THEN    n17<='1';n26<='1';
            ELSIF not_stb_s='1' AND not_mclr_s='1' THEN
            IF clk'EVENT AND clk='0' THEN
                IF ij3='0' AND ik3='0' THEN     n17<=n17;n26<=n26;
                    ELSIF ij3='1' AND ik3='0' THEN    n17<='1';n26<='0';
                    ELSIF ij3='0' AND ik3='1' THEN    n17<='0';n26<='1';
                    ELSIF ij3='1' AND ik3='1' THEN     n17<=n26;n26<=n17;
                    END IF;
                END IF;
            END IF;
        END PROCESS;
    PROCESS(ldl,n26,clk,n9,p_u,n27,n15,ij4,ik4)
        BEGIN
        IF p_u='0' OR n9='0' THEN     ij4<=ldl;ik4<=n26;
            ELSIF p_u='1' AND n9='1' THEN
            IF clk='1' THEN
                IF (ldl='1' AND n15='1') OR (n26='1' AND n27='1') THEN
                        ij4<=ldl;ik4<=n26;
                    ELSE      ij4<=ij4;ik4<=ik4;
                    END IF;
                ELSE      ij4<=ldl;ik4<=n26;
                END IF;
            ELSE NULL;
            END IF;
        END PROCESS;
    PROCESS(p_u,n9,clk)
        BEGIN
        IF p_u='0' AND n9='1' THEN    n27<='0';n15<='1';
            ELSIF p_u='1' AND n9='0' THEN    n27<='1';n15<='0';
            ELSIF p_u='0' AND n9='0' THEN    n27<='1';n15<='1';
            ELSIF p_u='1' AND n9='1' THEN
            IF clk'EVENT AND clk='0' THEN
                IF ij4='0' AND ik4='0' THEN     n27<=n27;n15<=n15;
                    ELSIF ij4='1' AND ik4='0' THEN    n27<='1';n15<='0';
                    ELSIF ij4='0' AND ik4='1' THEN    n27<='0';n15<='1';
                    ELSIF ij4='1' AND ik4='1' THEN     n27<=n15;n15<=n27;
                    END IF;
                END IF;
            END IF;
```

Figure 5.8.3.1-1 continued. The DQD Core Model.

```
    END PROCESS;
PROCESS(n0,n18,clk,n19,n9,n4,n2,ij5,ik5)
    BEGIN
    IF n9='0' OR n19='0' THEN     ij5<=n0;ik5<=n18;
        ELSIF n9='1' AND n19='1' THEN
        IF clk='1' THEN
            IF (n0='1' AND n2='1') OR (n18='1' AND n4='1') THEN
                    ij5<=n0;ik5<=n18;
                ELSE     ij5<=ij5;ik5<=ik5;
                END IF;
            ELSE     ij5<=n0;ik5<=n18;
            END IF;
        ELSE NULL;
        END IF;
    END PROCESS;
PROCESS(n9,n19,clk)
    BEGIN
    IF n9='0' AND n19='1' THEN     n4<='0';n2<='1';
        ELSIF n9='1' AND n19='0' THEN     n4<='1';n2<='0';
        ELSIF n9='0' AND n19='0' THEN     n4<='1';n2<='1';
        ELSIF n9='1' AND n19='1' THEN
        IF clk'EVENT AND clk='0' THEN
            IF ij5='0' AND ik5='0' THEN     n4<=n4;n2<=n2;
                ELSIF ij5='1' AND ik5='0' THEN     n4<='1';n2<='0';
                ELSIF ij5='0' AND ik5='1' THEN     n4<='0';n2<='1';
                ELSIF ij5='1' AND ik5='1' THEN     n4<=n2;n2<=n4;
                END IF;
            END IF;
        END IF;
    END PROCESS;
PROCESS(ldl,n24,clk,p_u,n9,iact,inot_act,ij6,ik6)
    BEGIN
    IF n9='0' OR p_u='0' THEN     ij6<=ldl;ik6<=n24;
        ELSIF n9='1' AND p_u='1' THEN
        IF clk='1' THEN
            IF (ldl='1' AND inot_act='1') OR (n24='1' AND iact='1') THEN
                    ij6<=ldl;ik6<=n24;
                ELSE     ij6<=ij6;ik6<=ik6;
                END IF;
            ELSE     ij6<=ldl;ik6<=n24;
            END IF;
        ELSE NULL;
        END IF;
    END PROCESS;
PROCESS(n9,p_u,clk)
    BEGIN
    IF n9='0' AND p_u='1' THEN     iact<='0';inot_act<='1';
        ELSIF n9='1' AND p_u='0' THEN     iact<='1';inot_act<='0';
        ELSIF n9='0' AND p_u='0' THEN     iact<='1';inot_act<='1';
```

Figure 5.8.3.1-1 continued. The DQD Core Model.

```
            ELSIF n9='1' AND p_u='1' THEN
            IF clk'EVENT AND clk='0' THEN
                IF ij6='0' AND ik6='0' THEN      iact<=iact;inot_act<=inot_act;
                    ELSIF ij6='1' AND ik6='0' THEN     iact<='1';inot_act<='0';
                    ELSIF ij6='0' AND ik6='1' THEN     iact<='0';inot_act<='1';
                    ELSIF ij6='1' AND ik6='1' THEN      iact<=inot_act;inot_act<=iact;
                    END IF;
                END IF;
            END IF;
        END PROCESS;
    PROCESS(n22,n21,clk,p_u,not_mclr_s,n28,n7,ij7,ik7)
        BEGIN
        IF not_mclr_s='0' OR p_u='0' THEN     ij7<=n22;ik7<=n21;
            ELSIF not_mclr_s='1' AND p_u='1' THEN
            IF clk='1' THEN
                IF (n22='1' AND n7='1') OR (n21='1' AND n28='1') THEN
                        ij7<=n22;ik7<=n21;
                    ELSE     ij7<=ij7;ik7<=ik7;
                    END IF;
                ELSE     ij7<=n22;ik7<=n21;
                END IF;
            ELSE NULL;
            END IF;
        END PROCESS;
    PROCESS(not_mclr_s,p_u,clk)
        BEGIN
        IF not_mclr_s='0' AND p_u='1' THEN    n28<='0';n7<='1';
            ELSIF not_mclr_s='1' AND p_u='0' THEN    n28<='1';n7<='0';
            ELSIF not_mclr_s='0' AND p_u='0' THEN    n28<='1';n7<='1';
            ELSIF not_mclr_s='1' AND p_u='1' THEN
            IF clk'EVENT AND clk='0' THEN
                IF ij7='0' AND ik7='0' THEN     n28<=n28;n7<=n7;
                    ELSIF ij7='1' AND ik7='0' THEN    n28<='1';n7<='0';
                    ELSIF ij7='0' AND ik7='1' THEN    n28<='0';n7<='1';
                    ELSIF ij7='1' AND ik7='1' THEN     n28<=n7;n7<=n28;
                    END IF;
                END IF;
            END IF;
        END PROCESS;
    PROCESS(ldl,n0,clk,p_u,n20,n10,n16,ij8,ik8)
        BEGIN
        IF n20='0' OR p_u='0' THEN     ij8<=ldl;ik8<=n0;
            ELSIF n20='1' AND p_u='1' THEN
            IF clk='1' THEN
                IF (ldl='1' AND n16='1') OR (n0='1' AND n10='1') THEN
                        ij8<=ldl;ik8<=n0;
                    ELSE     ij8<=ij8;ik8<=ik8;
                    END IF;
                ELSE     ij8<=ldl;ik8<=n0;
```

Figure 5.8.3.1-1 continued. The DQD Core Model.

```
                END IF;
            ELSE NULL;
            END IF;
        END PROCESS;
    PROCESS(n20,p_u,clk)
        BEGIN
        IF n20='0' AND p_u='1' THEN     n10<='0';n16<='1';
            ELSIF n20='1' AND p_u='0' THEN     n10<='1';n16<='0';
            ELSIF n20='0' AND p_u='0' THEN     n10<='1';n16<='1';
            ELSIF n20='1' AND p_u='1' THEN
            IF clk'EVENT AND clk='0' THEN
                IF ij8='0' AND ik8='0' THEN     n10<=n10;n16<=n16;
                    ELSIF ij8='1' AND ik8='0' THEN     n10<='1';n16<='0';
                    ELSIF ij8='0' AND ik8='1' THEN     n10<='0';n16<='1';
                    ELSIF ij8='1' AND ik8='1' THEN     n10<=n16;n16<=n10;
                    END IF;
                END IF;
            END IF;
        END PROCESS;
-- counter implementations
    PROCESS(n10,n0,cnt_1,din1)
        BEGIN
        IF n10='0' THEN     buf1<=to_integer(din1);
            ELSE
            IF cnt_1'EVENT AND cnt_1='1' THEN
                IF buf1=15 THEN   buf1<=0;
                    ELSE     buf1<=buf1+1;
                    END IF;
                END IF;
            END IF;
        END PROCESS;
    dout1<=to_vector(buf1,4);
    n5<='1'   WHEN dout1="1111" ELSE
            '0';
    n29<='0'   WHEN dout1="1111" AND cnt_1='0' ELSE
            '1';
    PROCESS(n10,n29,cnt_2,din2)
        BEGIN
        IF n10='0' THEN     buf2<=to_integer(din2);
            ELSIF n29='0' THEN
            IF cnt_2'EVENT AND cnt_2='1' THEN
                IF buf2=15 THEN   buf2<=0;
                    ELSE     buf2<=buf2+1;
                    END IF;
                END IF;
            END IF;
        END PROCESS;
    dout2<=to_vector(buf2,4);
    n6<='1'   WHEN dout2="1111" ELSE
```

Figure 5.8.3.1-1 continued. The DQD Core Model.

```
            '0';
-- shift register implementations
    PROCESS(p_u,inot_lrc_s,n30,n40)
        BEGIN
        IF p_u='0' THEN     n50<='0';n8<='0';
            ELSIF inot_lrc_s'EVENT AND inot_lrc_s='1' THEN
            IF isrm='0' THEN      n50<=n30;n8<=n40;
                ELSE   n8<=n50;
                IF sdi='0' THEN  n50<='0';
                    ELSIF sdi='1' THEN  n50<='1';
                    END IF;
                END IF;
            END IF;
        END PROCESS;
    PROCESS(p_u,iurc_s,d32,d33)
        BEGIN
        IF p_u='0' THEN     n30<='0';n40<='0';
            ELSIF iurc_s'EVENT AND iurc_s='1' THEN      n30<=d32;n40<=d33;
            END IF;
        END PROCESS;
    END  arch_dqd_core_beh;
```

Figure 5.8.3.1-1 continued. The DQD Core Model.

5.8.4 The EIA_567_DS Package

Next, we can generate the EIA_567_DS package which is the collection of all remaining functions. Figure 5.8.4-1 contains this package for the DQD module. Table 5.8.4-1 contains a listing of these functions. For a description of these functions, refer to chapter 4.

5.8.5 The Electronic Data Sheet Model

Finally, the EDS model of the DQD module can be assembled as shown in Figure 5.8.5-1. As noted earlier, this model contains synchronous and asynchronous checkers in addition to the input and output drivers.

5.9 Dealing with Multiple Timing Constraints per Pin

5.9.1 Constraint Selection through Generics

There are two options which have been identified for dealing with multiple timing constraints on a single pin. The first option makes use of some of the features of the EIA-567 package approach. The second options makes use of additional output drivers to implement this capability.

5.9.1.1 It may be noted that the "design_TV" or "design_DS" package allows multiple constraints to be declared for a given I/O pin. It does this through the MAX_ASYNC_CONSTRAINTS_PER_PIN, MAX_SYNC_CONSTRAINTS_PER_PIN, and MAX_DELAY_CONSTRAINTS_PER_PIN constants. Based upon these constants, corresponding elements in the electronic data sheet of the "design_PV" or "design_DS" package may reference multiple constraints for each I/O pin.

5.9.1.2 For a case in which multiple delay constraints are imposed on a design, separate delay constraints may be declared in the "design_TV" or "design_DS" package. For example, one delay constraint would be declared for clock-to-data valid and another constraint would be declared for reset-to-data valid. Within the eds, each pin index entry would contain a constraint map to the "max_delay_constraints_per_pin" value (in this case, two). It is recommended that the first constraint be the most commonly used one (i.e. clock-to-data valid). For pins affected by a single constraint, all map entries would go to the same constraint for that pin.

Data Type	Source Package
VOLTAGE_VECTOR	design_EV
CURRENT_VECTOR	design_EV
ELECTRICAL_PIN_SPEC	design_EV
VMAX	design_EV
VMIN	design_EV
VNOM	design_EV
IMAX	design_EV
LOWER_TEMPERATURE_LIMIT	design_EV
UPPER_TEMPERATURE_LIMIT	design_EV
ASYNCS	design_TV
SYNCS	design_TV
DELAYS	design_TV
MAX_ASYNC_CONSTRAINTS_PER_PIN	design_TV
MAX_SYNC_CONSTRAINTS_PER_PIN	design_TV
MAX_DELAY_CONSTRAINTS_PER_PIN	design_TV
PIN_INDEX	design_PV
PIN_AND_SIGNAL_CORROLATION	design_PV
POINT	EIA_567_TV
ASYNC_ARRAY	EIA_567_PV
SYNC_ARRAY	EIA_567_PV
DELAY_ARRAY	EIA_567_PV
PIN_POINTERS	EIA_567_PV
POINT_SPECIFICATION	design_PV
OPNT	design_PV
ELECTRONIC_DATA_SHEET	design_PV
PART_NAME	design_PV
ESD_PROTECTION	design_PV
OUTLINE_DRAWING	design_PV
MAXIMUM_POWER_DISSIPATION	design_PV
EDS	design_PV

Table 5.8.3.2-1. Design_DS Package Declarations.

Subprogram	Source Package
"="	EIA_567_TV
">="	EIA_567_TV
"<="	EIA_567_TV
">"	EIA_567_TV
"<"	EIA_567_TV
VALID_OPERATING_POINT	EIA_567_TV
GET_TIMING	EIA_567_TB

Table 5.8.4-1. EIA-567 DS Package Declarations.

5.9.1.3 At this point, the appropriate drivers and/or checkers used in the VHDL model would need to implement a variable in the generic map for selecting the constraint to be used. For example, an output driver instantiation might index to "delay_spec(delay_select)" instead of "delay_spec(1)".

```
--*****************************************************************
--   (c) Copyright 1994 by the
--         Naval Air Warfare Center, Aircraft Division, Indianapolis
--   Source
--         Author(s):                Charles K. Rogers
--         Organization:            NAWC-ADI
--                                                    Code 306, MS-42
--                                                    6000 E 21st St
--                                                    Indianapolis,  IN           46219-2189
--                                                    Phone:        317-353-3579
--                                                    EMail:        ROGERSC1@po2.nawc-ad-indy.navy.mil
--         Reference:                MIL-M-28787/212
--         Project:                   SHARP   TIREP
--   DESC Certification
--         Status:                      TBD
--*****************************************************************
--   Revision History
--         Version:                  1.1
--               Date:                      26 September 1994
--               Comments:            Added cycle time checks
--         Version:                  1.0
--               Date:                      23 May 1994
--               Comments:            Original Release
--*****************************************************************
--   Module Description
--         File:                               dqd_ds_p.vhd
--         Module  Name(s):       dqd_ds package
--         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 DQD
--               standard hardware module.
--         Notes:                              none
--*****************************************************************
--   Standard Libraries/Packages
--         none
--   Associated Packages (order of analysis implied)
--         eia_567                    standard eia package
--   Component  Models
--         none
--   Platform:                         486/33MHz PC
--   Software/Version: V-System for Windows, Version 3.0
--*****************************************************************
--   Design Specification Elements
--         entire file
--*****************************************************************
LIBRARY eia;
USE eia.EIA_567.ALL;
```

Figure 5.8.3.2-1. The DQD Design Specification Package.

```
PACKAGE dqd_ds IS
--*****************************************************************
-- Declaration of number of voltage supplies included in the design
--*****************************************************************
   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;
---------------------------------------------------------------------
-- Declaration of input constraints which are appropriate for this model
---------------------------------------------------------------------
   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_ds (design specification) 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;
--*****************************************************************
-- 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  (andx,bndx,cndx,dndx,endx,fndx,gndx,d32ndx,d33ndx,
      sow_sndx,p_undx,ldlndx,not_ovd_sndx,dta_clk_sndx,clkndx,ena_0ndx,
      not_eocndx,not_mclr_sndx,cnt_1ndx,cnt_2ndx,sdindx,not_stb_sndx,
      actndx,not_actndx,cnt_0ndx,dta_0_sndx,dta_1_sndx,srmndx,urc_sndx,
      not_lrc_sndx,clk1ndx,ldl1ndx,rdy_sndx,not_dta_0_rzndx,not_dta_1_rzndx,
      stb_csndx,ovr_0ndx,not_eoc1ndx,vccndx,gndndx);
---------------------------------------------------------------------
-- Definition of a type and the signal to pin corrolation
---------------------------------------------------------------------
```

Figure 5.8.3.2-1 continued. The DQD DS Package.

```
   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.   In this constant definition, each signal is
-- explictly tied to a pin.
--****************************************************************
   CONSTANT INTERFACE_CONNECTIONS:PIN_AND_SIGNAL_CORROLATION:=(
-- 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.
      andx=>(pin_id=>"34                          ",
         signal_name=>"a                          "),
      bndx=>(pin_id=>"8                           ",
         signal_name=>"b                          "),
      cndx=>(pin_id=>"13                          ",
         signal_name=>"c                          "),
      dndx=>(pin_id=>"12                          ",
         signal_name=>"d                          "),
      endx=>(pin_id=>"32                          ",
         signal_name=>"e                          "),
      fndx=>(pin_id=>"36                          ",
         signal_name=>"   F                       "),
      gndx=>(pin_id=>"22                          ",
         signal_name=>"g                          "),
      d32ndx=>(pin_id=>"19                        ",
         signal_name=>"d32                        "),
      d33ndx=>(pin_id=>"39                        ",
         signal_name=>"d33                        "),
      sow_sndx=>(pin_id=>"14                      ",
         signal_name=>"sow    s                   "),
      p_undx=>(pin_id=>"4                         ",
         signal_name=>"p.U.                       "),
      ldlndx=>(pin_id=>"7                         ",
         signal_name=>"ldl                        "),
      not_ovd_sndx=>(pin_id=>"15                  ",
         signal_name=>"   NOT ovd s               "),
      dta_clk_sndx=>(pin_id=>"18                  ",
         signal_name=>"dta   clk   s              "),
      clkndx=>(pin_id=>"24                        ",
         signal_name=>"clk                        "),
      ena_0ndx=>(pin_id=>"25                      ",
         signal_name=>"ena    0                   "),
      not_eocndx=>(pin_id=>"27                    ",
         signal_name=>"   NOT eoc                 "),
      not_mclr_sndx=>(pin_id=>"29                 ",
         signal_name=>"   NOT mclr s              "),
```

Figure 5.8.3.2-1 continued. The DQD DS Package.

```
          cnt_1ndx=>(pin_id=>"33                                  ",
                signal_name=>"cnt   2                             "),
          cnt_2ndx=>(pin_id=>"30                                  ",
                signal_name=>"cnt   1                             "),
          sdindx=>(pin_id=>"40                                    ",
                signal_name=>"sdi                                 "),
          not_stb_sndx=>(pin_id=>"9                               ",
                signal_name=>"   NOT stb s                        "),
          actndx=>(pin_id=>"2                                     ",
                signal_name=>"act                                 "),
          not_actndx=>(pin_id=>"37                                ",
                signal_name=>"   NOT act                          "),
          cnt_0ndx=>(pin_id=>"5                                   ",
                signal_name=>"cnt   0                             "),
          dta_0_sndx=>(pin_id=>"11                                ",
                signal_name=>"dta  0  s                           "),
          dta_1_sndx=>(pin_id=>"31                                ",
                signal_name=>"dta  1  s                           "),
          srmndx=>(pin_id=>"16                                    ",
                signal_name=>"srm                                 "),
          urc_sndx=>(pin_id=>"20                                  ",
                signal_name=>"urc   s                             "),
          not_lrc_sndx=>(pin_id=>"26                              ",
                signal_name=>"   NOT lrc s                        "),
          clk1ndx=>(pin_id=>"38                                   ",
                signal_name=>"clk1                                "),
          ldl1ndx=>(pin_id=>"3                                    ",
                signal_name=>"ldl1                                "),
          rdy_sndx=>(pin_id=>"6                                   ",
                signal_name=>"rdy   s                             "),
          not_dta_0_rzndx=>(pin_id=>"23                           ",
                signal_name=>"   NOT dta 0 rz                     "),
          not_dta_1_rzndx=>(pin_id=>"17                           ",
                signal_name=>"   NOT dta 1 rz                     "),
          stb_csndx=>(pin_id=>"21                                 ",
                signal_name=>"stb   cs                            "),
          ovr_0ndx=>(pin_id=>"28                                  ",
                signal_name=>"ovr   0                             "),
          not_eoc1ndx=>(pin_id=>"35                               ",
                signal_name=>"   NOT eoc1                         "),
          vccndx=>(pin_id=>"1                                     ",
                signal_name=>"vcc                                 "),
          gndndx=>(pin_id=>"10                                    ",
                signal_name=>"gnd                                 "));
--------------------------------------------------------------------
-- timing related declarations
--    declaration of an operating point type based on the number of
--    voltage supplies defined by the user
--------------------------------------------------------------------
     TYPE POINT IS RECORD
```

Figure 5.8.3.2-1 continued. The DQD DS Package.

```
        selection: OPERATING_SELECTION;
        temp: TEMPERATURE;
        SUPPLY: VOLTAGE_VECTOR;
        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;
        sync_spec: SYNC_ARRAY;
        delay_spec: DELAY_ARRAY;
        elec_spec: NATURAL;
        END RECORD;
---------------------------------------------------------------------
-- 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;
--*****************************************************************
-- The array range for the ELECTRONIC_DATA_SHEET type may require
-- updating.
--*****************************************************************
    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:="dqd";
    CONSTANT ESD_PROTECTION: ESD_CLASS:= UNKNOWN;
    CONSTANT OUTLINE_DRAWING: STRING:="tbd";
    CONSTANT MAXIMUM_POWER_DISSIPATION: POWER:=2860  mw;
    CONSTANT EDS: ELECTRONIC_DATA_SHEET;
    END  dqd_ds;

PACKAGE BODY dqd_ds  IS
```

Figure 5.8.3.2-1 continued. The DQD DS Package.

```
--*****************************************************************
-- 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                5.0V            1N916 1.74kohm165ohm50 pf
--        2                5.0V            1N916 6.04kohm247ohm50 pf
--        3                5.0V            1N916 3.01kohm247ohm50 pf
--        4                5.0V            1N916 3.09kohm253ohm50 pf
--        5                5.0V            1N916 6.34kohm253ohm50 pf
--        6                5.0V            1N916 6.65kohm274ohm50 pf
-------------------------------------------------------------------
   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 - all inputs excluding d32, d33, sdi, clk, p_u 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=>-100  ua,  pin_load=>  2  pf),
-- pin class #2 - inputs d32, d33, sdi 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=>0.7  v,  iil=>-100  ua,  pin_load=>  2  pf),
-- pin class #3 - inputs clk, p_u minimum
-- Note:  The p_u input is not specified.  It has been equated to the clk input for
--   minimum conditions and to clk+5 ma for maximum conditions (class #17)
-- Note:  The iih of -18,000 ua in the specification is not understood.
      (voh=>0.0  v,  ioh=>0.0  ua,  vol=>0.0  v,  iol=>0.0  ua,test_load=>0,
      vih=>2.0  v,  iih=>-18.0  ma,  vil=>0.8  v,  iil=>-100  ua,  pin_load=>  2  pf),
-- pin class #4 - inputs d32, d33 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=>20  ua,  vil=>0.8  v,  iil=>-0.389  ma,  pin_load=>  15  pf),
-- pin class #5 - input sdi 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=>-0.778  ma,  pin_load=>  25  pf),
-- pin class #6 - inputs a, b, c, d, e, f, g, cnt_1, cnt_2, sow_s, not_mclr_s maximum
      (voh=>0.0  v,  ioh=>0.0  ua,  vol=>0.0  v,  iol=>0.0  ua,test_load=>0,
```

Figure 5.8.3.2-1 continued. The DQD DS Package.

```
        vih=>2.0  v,  iih=>40  ua,  vil=>0.8  v,  iil=>-1.73  ma,  pin_load=>  15  pf),
-- pin class #7 - inputs not_stb_s, not_ovd_s 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=>120  ua,  vil=>0.8  v,  iil=>-4.80  ma,  pin_load=>  25  pf),
-- pin class #8 - inputs dta_clk_s, ena_0, not_eoc 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=>80  ua,  vil=>0.8  v,  iil=>-3.46  ma,  pin_load=>  25  pf),
-- pin class #9 - input ldl 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=>360  ua,  vil=>0.8  v,  iil=>-15.6  ma,  pin_load=>  95  pf),
-- pin class #10 - input clk 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=>640  ua,  vil=>0.8  v,  iil=>-25.6  ma,  pin_load=>  120  pf),
-- pin class #11 - output clk1 maximum
        (voh=>2.4   v,   ioh=>-1400   ua,   vol=>0.4   v,   iol=>24.0   ma,test_load=>1,
        vih=>0.0  v,  iih=>0.0  ua,  vil=>0.0  v,  iil=>0.0  ua,  pin_load=>  0  pf),
-- pin class #12 - outputs ovr_0, not_dta_0_rz, not_dta_1_rz, stb_cs, not_eoc1, rdy_s maximum
        (voh=>2.4   v,   ioh=>-400   ua,   vol=>0.4   v,   iol=>16.0   ma,test_load=>2,
        vih=>0.0  v,  iih=>0.0  ua,  vil=>0.0  v,  iil=>0.0  ua,  pin_load=>  0  pf),
-- pin class #13 - output ldl1 maximum
        (voh=>2.4   v,   ioh=>-800   ua,   vol=>0.4   v,   iol=>16.0   ma,test_load=>3,
        vih=>0.0  v,  iih=>0.0  ua,  vil=>0.0  v,  iil=>0.0  ua,  pin_load=>  0  pf),
-- pin class #14 - output not_lrc_s maximum
        (voh=>2.4   v,   ioh=>-780   ua,   vol=>0.4   v,   iol=>15.6   ma,test_load=>4,
        vih=>0.0  v,  iih=>0.0  ua,  vil=>0.0  v,  iil=>0.0  ua,  pin_load=>  0  pf),
-- pin class #15 - output urc_s, srm maximum
        (voh=>2.4   v,   ioh=>-380   ua,   vol=>0.4   v,   iol=>15.6   ma,test_load=>5,
        vih=>0.0  v,  iih=>0.0  ua,  vil=>0.0  v,  iil=>0.0  ua,  pin_load=>  0  pf),
-- pin class #16 - outputs dta_0_s, dta_1_s, act, not_act, cnt_0 maximum
        (voh=>2.4   v,   ioh=>-360   ua,   vol=>0.4   v,   iol=>14.4   ma,test_load=>6,
        vih=>0.0  v,  iih=>0.0  ua,  vil=>0.0  v,  iil=>0.0  ua,  pin_load=>  0  pf),
-- pin class #17 - input clk 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=>640  ua,  vil=>0.8  v,  iil=>-31.0  ma,  pin_load=>  120  pf));
--*************************************************************
-- 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=>(650    ma));
--*************************************************************
-- Temperature limits
--*************************************************************
    CONSTANT LOWER_TEMPERATURE_LIMIT : TEMPERATURE := 0   degrees_c;
    CONSTANT UPPER_TEMPERATURE_LIMIT : TEMPERATURE := 60   degrees_c;
--*************************************************************
-- Definition of asynchronous constraint classes
--*************************************************************
    CONSTANT ASYNCS : ASYNC_CONSTRAINTS := (
```

Figure 5.8.3.2-1 continued. The DQD DS Package.

```
 -- Async constraint class #0 for no constraints
         (t1min=>0   ns,t0min=>0   ns,t1max=>0   ns,t0max=>0   ns,
         cycmin=>0   ns,cycmax=>0   ns),
 -- Async constraint class #1 for dta_clk_s
         (t1min=>220   ns,t0min=>280   ns,t1max=>0   ns,t0max=>0   ns,
         cycmin=>5   us,cycmax=>0   ns),
 -- Async constraint class #2 for not_mclr_s
         (t1min=>0   ns,t0min=>50   ns,t1max=>0   ns,t0max=>0   ns,
         cycmin=>0   ns,cycmax=>0   ns),
 -- Async constraint class #3 for not_stb_s
 -- Note:  t0max is period of dta_clk_s minus 20 ns.     t0max is not set
 -- Note:  for the testbench.
         (t1min=>0   ns,t0min=>50   ns,t1max=>0   ns,t0max=>0   ns,
         cycmin=>0   ns,cycmax=>0   ns));
--***************************************************************
 -- Definition of synchronous constraint classes
--***************************************************************
     CONSTANT SYNCS:SYNC_CONSTRAINTS:=(
 -- Sync constraint class #0 for no constraints
         (setup=>0     ns,hold=>0     ns,edge=>SIGNAL_EDGELEFT),
 -- Sync constraint class #1 for not_ovd_s with respect to not_stb_s
 --    The hold time for this constraint should be 60 ns, but this will
 --    result in a testbench failure.
         (setup=>0     ns,hold=>0     ns,edge=>e10));
--***************************************************************
 -- Definition of output delay classes
--***************************************************************
     CONSTANT DELAYS:OUTPUT_DELAYS:=(
 -- 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,
         ttlh=>0   ns,tthl=>0   ns),
 -- Output delay class #1 (dta_clk_s to not_dta_1_rz/not_dta_0_rz nominal)
         (tlh=>38   ns,thl=>42   ns,tlz=>0   ns,thz=>0   ns,tzl=>0   ns,tzh=>0   ns,
         ttlh=>17   ns,tthl=>12   ns),
 -- Output delay class #2 (dta_clk_s to not_dta_1_rz/not_dta_0_rz best)
         (tlh=>12   ns,thl=>12   ns,tlz=>0   ns,thz=>0   ns,tzl=>0   ns,tzh=>0   ns,
         ttlh=>4   ns,tthl=>4   ns),
 -- Output delay class #3 (dta_clk_s to not_dta_1_rz/not_dta_0_rz worst)
         (tlh=>64   ns,thl=>72   ns,tlz=>0   ns,thz=>0   ns,tzl=>0   ns,tzh=>0   ns,
         ttlh=>30   ns,tthl=>20   ns),
 -- Output delay class #4 (dta_clk_s to dta_0_s nominal)
         (tlh=>71   ns,thl=>64   ns,tlz=>0   ns,thz=>0   ns,tzl=>0   ns,tzh=>0   ns,
         ttlh=>17   ns,tthl=>12   ns),
 -- Output delay class #5 (dta_clk_s to dta_0_s best)
         (tlh=>20   ns,thl=>20   ns,tlz=>0   ns,thz=>0   ns,tzl=>0   ns,tzh=>0   ns,
         ttlh=>4   ns,tthl=>4   ns),
 -- Output delay class #6 (dta_clk_s to dta_0_s worst)
         (tlh=>122   ns,thl=>108   ns,tlz=>0   ns,thz=>0   ns,tzl=>0   ns,tzh=>0   ns,
         ttlh=>30   ns,tthl=>20   ns),
 -- Output delay class #7 (dta_clk_s to dta_1_s nominal)
```

Figure 5.8.3.2-1 continued. The DQD DS Package.

```
        (tlh=>77 ns,thl=>80 ns,tlz=>0 ns,thz=>0 ns,tzl=>0 ns,tzh=>0 ns,
        ttlh=>17 ns,tthl=>12 ns),
  -- Output delay class #8 (dta_clk_s to dta_1_s best)
        (tlh=>24 ns,thl=>24 ns,tlz=>0 ns,thz=>0 ns,tzl=>0 ns,tzh=>0 ns,
        ttlh=>4 ns,tthl=>4 ns),
  -- Output delay class #9 (dta_clk_s to dta_1_s worst)
        (tlh=>131 ns,thl=>136 ns,tlz=>0 ns,thz=>0 ns,tzl=>0 ns,tzh=>0 ns,
        ttlh=>30 ns,tthl=>20 ns),
  -- Output delay class #10 (dta_clk_s to stb_cs nominal)
        (tlh=>55 ns,thl=>50 ns,tlz=>0 ns,thz=>0 ns,tzl=>0 ns,tzh=>0 ns,
        ttlh=>17 ns,tthl=>12 ns),
  -- Output delay class #11 (dta_clk_s to stb_cs best)
        (tlh=>16 ns,thl=>16 ns,tlz=>0 ns,thz=>0 ns,tzl=>0 ns,tzh=>0 ns,
        ttlh=>4 ns,tthl=>4 ns),
  -- Output delay class #12 (dta_clk_s to stb_cs worst)
        (tlh=>95 ns,thl=>85 ns,tlz=>0 ns,thz=>0 ns,tzl=>0 ns,tzh=>0 ns,
        ttlh=>30 ns,tthl=>20 ns),
  -- Output delay class #13 (dta_clk_s to not_lrc_s nominal)
        (tlh=>42 ns,thl=>38 ns,tlz=>0 ns,thz=>0 ns,tzl=>0 ns,tzh=>0 ns,
        ttlh=>17 ns,tthl=>12 ns),
  -- Output delay class #14 (dta_clk_s to not_lrc_s best)
        (tlh=>12 ns,thl=>12 ns,tlz=>0 ns,thz=>0 ns,tzl=>0 ns,tzh=>0 ns,
        ttlh=>4 ns,tthl=>4 ns),
  -- Output delay class #15 (dta_clk_s to not_lrc_s worst)
        (tlh=>72 ns,thl=>64 ns,tlz=>0 ns,thz=>0 ns,tzl=>0 ns,tzh=>0 ns,
        ttlh=>30 ns,tthl=>20 ns),
  -- Output delay class #16 (general outputs nominal)
        (tlh=>17 ns,thl=>12 ns,tlz=>0 ns,thz=>0 ns,tzl=>0 ns,tzh=>0 ns,
        ttlh=>17 ns,tthl=>12 ns),
  -- Output delay class #17 (general outputs best)
        (tlh=>4 ns,thl=>4 ns,tlz=>0 ns,thz=>0 ns,tzl=>0 ns,tzh=>0 ns,
        ttlh=>4 ns,tthl=>4 ns),
  -- Output delay class #18 (general outputs worst)
        (tlh=>30 ns,thl=>20 ns,tlz=>0 ns,thz=>0 ns,tzl=>0 ns,tzh=>0 ns,
        ttlh=>30 ns,tthl=>20 ns),
  -- Output delay class #19 (/act and clko nominal)
        (tlh=>17 ns,thl=>9 ns,tlz=>0 ns,thz=>0 ns,tzl=>0 ns,tzh=>0 ns,
        ttlh=>17 ns,tthl=>9 ns),
  -- Output delay class #20 (/act and clko worst)
        (tlh=>30 ns,thl=>13 ns,tlz=>0 ns,thz=>0 ns,tzl=>0 ns,tzh=>0 ns,
        ttlh=>30 ns,tthl=>13 ns));
--***************************************************************
  -- Definition of an array of the indexes.   Each array holds the specific
  -- 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=>(
```

Figure 5.8.3.2-1 continued. The DQD DS Package.

```
        andx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        bndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        cndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        dndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        endx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        fndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        gndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        d32ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>2),
        d33ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>2),
        sow_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        p_undx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        ldlndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        not_ovd_sndx=>(sync_spec=>(1=>(1)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        dta_clk_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(1)),
delay_spec=>(1=>(0)),elec_spec=>1),
        clkndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>3),
        ena_0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        not_eocndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        not_mclr_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(2)),
delay_spec=>(1=>(0)),elec_spec=>1),
        cnt_1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        cnt_2ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
        sdindx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>2),
        not_stb_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(3)),
delay_spec=>(1=>(0)),elec_spec=>1),
        actndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(17)),elec_spec=>16),
        not_actndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(17)),elec_spec=>16),
        cnt_0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(17)),elec_spec=>16),
```

Figure 5.8.3.2-1 continued. The DQD DS Package.

```
            dta_0_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(5)),elec_spec=>16),
            dta_1_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(8)),elec_spec=>16),
            srmndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(17)),elec_spec=>15),
            urc_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(17)),elec_spec=>15),
            not_lrc_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(14)),elec_spec=>14),
            clk1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(17)),elec_spec=>11),
            ldl1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(17)),elec_spec=>13),
            rdy_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(17)),elec_spec=>12),
            not_dta_0_rzndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(2)),elec_spec=>12),
            not_dta_1_rzndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(2)),elec_spec=>12),
            stb_csndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(11)),elec_spec=>12),
            ovr_0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(17)),elec_spec=>12),
            not_eoc1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(17)),elec_spec=>12),
            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=>5.0    v)),
          edsnfo=>(
            andx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>1),
            bndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>1),
            cndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>1),
            dndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>1),
            endx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>1),
            fndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>1),
            gndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>1),
            d32ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>2),
```

Figure 5.8.3.2-1 continued. The DQD DS Package.

```
          d33ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>2),
          sow_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
          p_undx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
          ldlndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
          not_ovd_sndx=>(sync_spec=>(1=>(1)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
          dta_clk_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(1)),
delay_spec=>(1=>(0)),elec_spec=>1),
          clkndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>3),
          ena_0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
          not_eocndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
          not_mclr_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(2)),
delay_spec=>(1=>(0)),elec_spec=>1),
          cnt_1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
          cnt_2ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>1),
          sdindx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(0)),elec_spec=>2),
          not_stb_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(3)),
delay_spec=>(1=>(0)),elec_spec=>1),
          actndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(16)),elec_spec=>16),
          not_actndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(19)),elec_spec=>16),
          cnt_0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(16)),elec_spec=>16),
          dta_0_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(4)),elec_spec=>16),
          dta_1_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(7)),elec_spec=>16),
          srmndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(16)),elec_spec=>15),
          urc_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(16)),elec_spec=>15),
          not_lrc_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(13)),elec_spec=>14),
          clk1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(19)),elec_spec=>11),
          ldl1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(16)),elec_spec=>13),
          rdy_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
delay_spec=>(1=>(16)),elec_spec=>12),
```

Figure 5.8.3.2-1 continued. The DQD DS Package.

```
--*******************************************************************
--   (c) Copyright 1994 by the
--          Naval Air Warfare Center, Aircraft Division, Indianapolis
--   Source
--          Author(s):              Charles K. Rogers
--          Organization:             NAWC-ADI
--                                  Code 306, MS-42
--                                  6000 E 21st St
--                                  Indianapolis,  IN  46219-2189
--                                  Phone:   317-353-3579
--                                  EMail:   ROGERSC1@po2.nawc-ad-indy.navy.mil
--          Reference:              MIL-M-28787/212
--          Project:                SHARP   TIREP
--   DESC Certification
--          Status:                 TBD
--*******************************************************************
--   Revision History
--          Version:                1.0
--                Date:             23 May 1994
--                Comments:         Original Release
--*******************************************************************
--   Module Description
--          File:                   eia_ds_p.vhd
--          Module  Name(s):        eia_567_dspackage
--          Constraints:            none
--          Limitations:            none
--          I/O  Format(s):         none
--          Purpose and Use:        This is the eia 567 package which contains the
--                design specific declarations and functions as originally defined
--                by Len Finegold in the eia_567_tv, pv and toolbox packages.
--          Notes:                  none
--*******************************************************************
--   Standard Libraries/Packages
--          std_logic_1164          standard multi-value logic package
--   Associated Packages (order of analysis implied)
--          eia_567                  standard eia-567 package
--   Component  Models
--          none
--   Platform:                      486/33MHz PC
--   Software/Version:              V-System for Windows, Version 3.0
--*******************************************************************
--   Design Specification Elements
--          none
--*******************************************************************
LIBRARY ieee;
LIBRARY eia;
USE ieee.STD_LOGIC_1164.ALL;
USE eia.EIA_567.ALL;
USE  work.dqd_ds.ALL;
```

Figure 5.8.4-1. The EIA_567_DS Package.

```
PACKAGE EEIA_567_DS IS
------------------------------------------------------------------------
-- 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;
------------------------------------------------------------------------
-- function which selects the correct info based on the user specified
-- operating point and x generation
------------------------------------------------------------------------
    FUNCTION GET_TIMING(op_point: POINT) RETURN OPNT;
    END EIA_567_DS;

PACKAGE BODY EIA_567_DS 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
```

Figure 5.8.4-1 continued. The EIA_567_DS Package.

```
        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 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;
------------------------------------------------------------------------
```

Figure 5.8.4-1 continued. The EIA_567_DS Package.

```
-- Function to find the user selected operating point
----------------------------------------------------------------------
   FUNCTION GET_TIMING(op_point: POINT) RETURN OPNT IS
      VARIABLE found:BOOLEAN:= FALSE;
      VARIABLE temppoint:OPNT;
      BEGIN
----------------------------------------------------------------------
-- first scan operating points stored for an exact match
----------------------------------------------------------------------
      FOR index IN EDS'RANGE LOOP
         IF (EDS(index).selpoint=op_point)   THEN
            temppoint:= EDS(index);
            found:= TRUE;
            EXIT;
            END IF;
         END LOOP;
----------------------------------------------------------------------
-- if not found, additional scans for a "close" match can be inserted
-- here
----------------------------------------------------------------------
      IF NOT found THEN
         FOR index IN EDS'RANGE LOOP
            IF (EDS(index).selpoint.selection=op_point.selection)   THEN
               temppoint:= EDS(index);
               EXIT;
               END IF;
            END LOOP;
         ASSERT FALSE REPORT "DELAYS based upon selection GENERIC."
            SEVERITY NOTE;
         END IF;
      RETURN temppoint;
      END GET_TIMING;
   END EIA_567_DS;
```

Figure 5.8.4-1 continued. The EIA_567_DS Package.

```
--***************************************************************
--   (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
```

Figure 5.8.5-1. The DQD EDS Model.

```
--                              EMail:   ROGERSC1@po2.nawc-ad-indy.navy.mil
--        Reference:            MIL-M-28787/212
--        Project:              SHARP   TIREP
--    DESC Certification
--        Status:               TBD
--***************************************************************
--    Revision History
--        Version:              1.0
--            Date:             20 May 1994
--            Comments:         Original Release
--***************************************************************
--    Module Description
--        File:                 dqd_eds.vhd
--        Module  Name(s):      dqd_eds entity/arch_dqd_eds_str architecture
--        Constraints:          none
--        Limitations:          none
--        I/O  Format(s):       std_logic
--        Purpose and Use:      This VHDL module describes the DQD standard
--            hardware module in accordance with M28787/212. This module
--            implements a standard serial output circuit.
--        Notes:                none
--***************************************************************
--    Standard Libraries/Packages
--        std_logic_1164        standard multi-value logic package
--    Associated Packages (order of analysis implied)
--        eia_567                standard eia package
--        dqd_ds                DQD design specification package
--        eia_567_ds            eia design specification package
--    Component  Models
--        dqd_core              behavorial VHDL module core
--        i_driver              generic input driver
--        o_driver              generic output driver
--    Platform:                 486/33MHz PC
--    Software/Version:         V-System for Windows, Version 3.0
--***************************************************************
--    Design Specification Elements
--        entire file
--***************************************************************
LIBRARY ieee;
LIBRARY eia;
USE ieee.STD_LOGIC_1164.ALL;
USE eia.EIA_567.ALL;
USE  work.dqd_ds.ALL;
USE work.EIA_567_DS.ALL;
ENTITY dqd_eds  IS
    GENERIC(wi_a,wi_b,wi_c,wi_d,wi_e,wi_f,wi_g,wi_d32,wi_d33,wi_sow_s,
        wi_p_u,wi_ldl,wi_not_ovd_s,wi_dta_clk_s,wi_clk,wi_ena_0,
        wi_not_eoc,wi_not_mclr_s,wi_cnt_1,wi_cnt_2,wi_sdi,
        wi_not_stb_s,wi_act,wi_not_act,wi_cnt_0,wi_dta_0s,wi_dta_1s,wi_srm,
```

Figure 5.8.5-1 continued. The DQD EDS Model.

```
        wi_urc_s,wi_not_lrcs,wi_clk1,wi_ldl1,wi_rdy_s,wi_not_dta0rz,
        wi_not_dta1rz,wi_stb_cs,wi_ovr_0,wi_not_eoc1: 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=>5.0 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(a,b,c,d,e,f,g,d32,d33,sow_s,p_u,ldl,not_ovd_s,dta_clk_s: IN STD_LOGIC;
          clk,ena_0,not_eoc,not_mclr_s,cnt_1,cnt_2,sdi,not_stb_s: IN STD_LOGIC;
          act,not_act,cnt_0,dta_0_s,dta_1_s,srm,urc_s,not_lrc_s,clk1: INOUT STD_LOGIC;
          ldl1,rdy_s,not_dta_0_rz,not_dta_1_rz,stb_cs,ovr_0,not_eoc1: INOUT STD_LOGIC;
          vcc,gnd:IN STD_LOGIC);
    END dqd_eds;
ARCHITECTURE arch_dqd_eds_str OF dqd_eds IS
    COMPONENT dqd_core
        PORT(a,b,c,d,e,f,g,d32,d33,sow_s,p_u,ldl,not_ovd_s,dta_clk_s: IN STD_LOGIC;
          clk,ena_0,not_eoc,not_mclr_s,cnt_1,cnt_2,sdi,not_stb_s: IN STD_LOGIC;
          act,not_act,cnt_0,dta_0_s,dta_1_s,srm,urc_s,not_lrc_s,clk1: INOUT STD_LOGIC;
          ldl1,rdy_s,not_dta_0_rz,not_dta_1_rz,stb_cs,ovr_0,not_eoc1: INOUT STD_LOGIC);
        END COMPONENT;
    COMPONENT i_driver
        GENERIC(win_d:TIME;
          pin_data: EV_SIGNAL_LIMIT);
        PORT(a:IN STD_LOGIC;
          y:OUT STD_LOGIC);
        END COMPONENT;
    COMPONENT o_driver
        GENERIC(delay_y:DELAY;
```

Figure 5.8.5-1 continued. The DQD EDS Model.

```
            wir_o:TIME:=0  ns;
            pin_data: EV_SIGNAL_LIMIT;
            m_gen:SEVERITY_LEVEL;
            x_gen:BOOLEAN);
        PORT(a:IN STD_LOGIC;
            y:INOUT STD_LOGIC);
        END COMPONENT;
    COMPONENT async_checker
        GENERIC(x_gen:BOOLEAN:= TRUE;
            m_gen:SEVERITY_LEVEL:= WARNING;
            asyncconstraint: ASYNC);
        PORT(data_in: IN STD_LOGIC;
            data_out: OUT STD_LOGIC:='U');
        END COMPONENT;
    COMPONENT sync_checker
        GENERIC(x_gen:BOOLEAN:= TRUE;
            m_gen:SEVERITY_LEVEL:= WARNING;
            syncconstraint: SYNC);
        PORT(clk,data_in: IN STD_LOGIC;
            data_out: OUT STD_LOGIC:='U');
        END COMPONENT;
    SIGNAL ia,ib,ic,id,ie,i_f,ig,id32,id33,isow_s,ip_u,ildl,inot_ovd_s:STD_LOGIC;
    SIGNAL  idta_clk_s,iclk,iena_0,inot_eoc,inot_mclr_s,icnt_1,icnt_2:STD_LOGIC;
    SIGNAL  isdi,inot_stb_s,iact,inot_act,icnt_0,idta_0_s,idta_1_s,isrm:STD_LOGIC;
    SIGNAL  iurc_s,inot_lrc_s,iclk1,ildl1,irdy_s,inot_dta_0_rz:STD_LOGIC;
    SIGNAL  inot_dta_1_rz,istb_cs,iovr_0,inot_eoc1,odta_clk_s:STD_LOGIC;
    SIGNAL  onot_mclr_s,onot_stb_s,onot_ovd_s:STD_LOGIC;
    FOR ALL:dqd_core  USE ENTITY work.dqd_core;
    FOR ALL:i_driver USE ENTITY eia.i_driver;
    FOR ALL:o_driver  USE ENTITY eia.o_driver;
    FOR ALL:async_checker  USE ENTITY eia.async_checker;
    FOR ALL:sync_checker  USE ENTITY eia.sync_checker;
    CONSTANT EDS:OPNT:=  GET_TIMING(user_operating_point);
---------------------------------------------------------------------
-- The following constant definition is represented in one of its most
-- expanded forms.
---------------------------------------------------------------------
    CONSTANT INTERFACE_CONNECTIONS:PIN_AND_SIGNAL_CORROLATION:=(
-- 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.
        andx=>(pin_id=>"34                                      ",
            signal_name=>"a                                     "),
        bndx=>(pin_id=>"8                                       ",
            signal_name=>"b                                     "),
        cndx=>(pin_id=>"13                                      ",
```

Figure 5.8.5-1 continued. The DQD EDS Model.

```
      signal_name=>"c                                           "),
dndx=>(pin_id=>"12                                              ",
      signal_name=>"d                                           "),
endx=>(pin_id=>"32                                              ",
      signal_name=>"e                                           "),
fndx=>(pin_id=>"36                                              ",
      signal_name=>"f                                           "),
gndx=>(pin_id=>"22                                              ",
      signal_name=>"g                                           "),
d32ndx=>(pin_id=>"19                                            ",
      signal_name=>"d32                                         "),
d33ndx=>(pin_id=>"39                                            ",
      signal_name=>"d33                                         "),
sow_sndx=>(pin_id=>"14                                          ",
      signal_name=>"sow    s                                    "),
p_undx=>(pin_id=>"4                                             ",
      signal_name=>"p.u.                                        "),
ldlndx=>(pin_id=>"7                                             ",
      signal_name=>"ldl                                         "),
not_ovd_sndx=>(pin_id=>"15                                      ",
      signal_name=>"not   ovd   s                             "),
dta_clk_sndx=>(pin_id=>"18                                      ",
      signal_name=>"dta   clk   s                             "),
clkndx=>(pin_id=>"24                                            ",
      signal_name=>"clk                                         "),
ena_0ndx=>(pin_id=>"25                                          ",
      signal_name=>"ena    0                                    "),
not_eocndx=>(pin_id=>"27                                        ",
      signal_name=>"not     eoc                                 "),
not_mclr_sndx=>(pin_id=>"29                                     ",
      signal_name=>"not   mclr   s                              "),
cnt_1ndx=>(pin_id=>"33                                          ",
      signal_name=>"cnt    2                                    "),
cnt_2ndx=>(pin_id=>"30                                          ",
      signal_name=>"cnt    1                                    "),
sdindx=>(pin_id=>"40                                            ",
      signal_name=>"sdi                                         "),
not_stb_sndx=>(pin_id=>"9                                       ",
      signal_name=>"not   stb   s                             "),
actndx=>(pin_id=>"2                                             ",
      signal_name=>"act                                         "),
not_actndx=>(pin_id=>"37                                        ",
      signal_name=>"not     act                                 "),
cnt_0ndx=>(pin_id=>"5                                           ",
      signal_name=>"cnt    0                                    "),
dta_0_sndx=>(pin_id=>"11                                        ",
      signal_name=>"dta   0   s                                 "),
dta_1_sndx=>(pin_id=>"31                                        ",
      signal_name=>"dta   1   s                                 "),
```

Figure 5.8.5-1 continued. The DQD EDS Model.

```
    srmndx=>(pin_id=>"16                                        ",
        signal_name=>"srm                                       "),
    urc_sndx=>(pin_id=>"20                                      ",
        signal_name=>"urc    s                                  "),
    not_lrc_sndx=>(pin_id=>"26                                  ",
        signal_name=>"not  lrc  s                               "),
    clk1ndx=>(pin_id=>"38                                       ",
        signal_name=>"clk1                                      "),
    ldl1ndx=>(pin_id=>"3                                        ",
        signal_name=>"ldl1                                      "),
    rdy_sndx=>(pin_id=>"6                                       ",
        signal_name=>"rdy    s                                  "),
    not_dta_0_rzndx=>(pin_id=>"23                               ",
        signal_name=>"not  dta  0  rz                           "),
    not_dta_1_rzndx=>(pin_id=>"17                               ",
        signal_name=>"not  dta  1  rz                           "),
    stb_csndx=>(pin_id=>"21                                     ",
        signal_name=>"stb    cs                                 "),
    ovr_0ndx=>(pin_id=>"28                                      ",
        signal_name=>"ovr    0                                  "),
    not_eoc1ndx=>(pin_id=>"35                                   ",
        signal_name=>"not    eoc1                               "),
    vccndx=>(pin_id=>"1                                         ",
        signal_name=>"vcc                                       "),
    gndndx=>(pin_id=>"10                                        ",
        signal_name=>"gnd                                       "));
BEGIN
ASSERT VALID_OPERATING_POINT(user_operating_point)
    REPORT "operating outside OF the recommended RANGE OF this design"
    SEVERITY WARNING;
s0:i_driver GENERIC MAP(win_d=>wi_a,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(andx).elec_spec))
    PORT MAP(a=>a,y=>ia);
s1:i_driver GENERIC MAP(win_d=>wi_b,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(bndx).elec_spec))
    PORT MAP(a=>b,y=>ib);
s2:i_driver GENERIC MAP(win_d=>wi_c,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(cndx).elec_spec))
    PORT MAP(a=>c,y=>ic);
s3:i_driver GENERIC MAP(win_d=>wi_d,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(dndx).elec_spec))
    PORT MAP(a=>d,y=>id);
s4:i_driver GENERIC MAP(win_d=>wi_e,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(endx).elec_spec))
    PORT MAP(a=>e,y=>ie);
s5:i_driver GENERIC MAP(win_d=>wi_f,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(fndx).elec_spec))
    PORT MAP(a=>f,y=>i_f);
s6:i_driver GENERIC MAP(win_d=>wi_g,
```

Figure 5.8.5-1 continued. The DQD EDS Model.

```
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(gndx).elec_spec))
        PORT MAP(a=>g,y=>ig);
    s7:i_driver GENERIC MAP(win_d=>wi_d32,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(d32ndx).elec_spec))
        PORT MAP(a=>d32,y=>id32);
    s8:i_driver GENERIC MAP(win_d=>wi_d33,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(d33ndx).elec_spec))
        PORT MAP(a=>d33,y=>id33);
    s9:i_driver GENERIC MAP(win_d=>wi_sow_s,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(sow_sndx).elec_spec))
        PORT MAP(a=>sow_s,y=>isow_s);
    s10:i_driver GENERIC MAP(win_d=>wi_p_u,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(p_undx).elec_spec))
        PORT MAP(a=>p_u,y=>ip_u);
    s11:i_driver GENERIC MAP(win_d=>wi_ldl,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(ldlndx).elec_spec))
        PORT MAP(a=>ldl,y=>ildl);
    s12:i_driver GENERIC MAP(win_d=>wi_not_ovd_s,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(not_ovd_sndx).elec_spec))
        PORT MAP(a=>not_ovd_s,y=>inot_ovd_s);
    s13:i_driver GENERIC MAP(win_d=>wi_dta_clk_s,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(dta_clk_sndx).elec_spec))
        PORT MAP(a=>dta_clk_s,y=>idta_clk_s);
    s14:i_driver GENERIC MAP(win_d=>wi_clk,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(clkndx).elec_spec))
        PORT MAP(a=>clk,y=>iclk);
    s15:i_driver GENERIC MAP(win_d=>wi_ena_0,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(ena_0ndx).elec_spec))
        PORT MAP(a=>ena_0,y=>iena_0);
    s16:i_driver GENERIC MAP(win_d=>wi_not_eoc,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(not_eocndx).elec_spec))
        PORT MAP(a=>not_eoc,y=>inot_eoc);
    s17:i_driver GENERIC MAP(win_d=>wi_not_mclr_s,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(not_mclr_sndx).elec_spec))
        PORT MAP(a=>not_mclr_s,y=>inot_mclr_s);
    s18:i_driver GENERIC MAP(win_d=>wi_cnt_1,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(cnt_1ndx).elec_spec))
        PORT MAP(a=>cnt_1,y=>icnt_1);
    s19:i_driver GENERIC MAP(win_d=>wi_cnt_2,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(cnt_2ndx).elec_spec))
        PORT MAP(a=>cnt_2,y=>icnt_2);
    s20:i_driver GENERIC MAP(win_d=>wi_sdi,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(sdindx).elec_spec))
        PORT MAP(a=>sdi,y=>isdi);
    s21:i_driver GENERIC MAP(win_d=>wi_not_stb_s,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(not_stb_sndx).elec_spec))
        PORT MAP(a=>not_stb_s,y=>inot_stb_s);
    s22:o_driver
        GENERIC MAP(delay_y=>    DELAYS(EDS.edsnfo(actndx).delay_spec(1)),
```

Figure 5.8.5-1 continued. The DQD EDS Model.

```
        pin_data=>   ELECTRICAL_PIN_SPEC(EDS.edsnfo(actndx).elec_spec),
        m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_act)
    PORT MAP(a=>iact,y=>act);
s23:o_driver
    GENERIC MAP(delay_y=>   DELAYS(EDS.edsnfo(not_actndx).delay_spec(1)),
        pin_data=>   ELECTRICAL_PIN_SPEC(EDS.edsnfo(not_actndx).elec_spec),
        m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_not_act)
    PORT MAP(a=>inot_act,y=>not_act);
s24:o_driver
    GENERIC MAP(delay_y=>   DELAYS(EDS.edsnfo(cnt_0ndx).delay_spec(1)),
        pin_data=>   ELECTRICAL_PIN_SPEC(EDS.edsnfo(cnt_0ndx).elec_spec),
        m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_cnt_0)
    PORT MAP(a=>icnt_0,y=>cnt_0);
s25:o_driver
    GENERIC MAP(delay_y=>   DELAYS(EDS.edsnfo(dta_0_sndx).delay_spec(1)),
        pin_data=>   ELECTRICAL_PIN_SPEC(EDS.edsnfo(dta_0_sndx).elec_spec),
        m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_dta_0s)
    PORT MAP(a=>idta_0_s,y=>dta_0_s);
s26:o_driver
    GENERIC MAP(delay_y=>   DELAYS(EDS.edsnfo(dta_1_sndx).delay_spec(1)),
        pin_data=>   ELECTRICAL_PIN_SPEC(EDS.edsnfo(dta_1_sndx).elec_spec),
        m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_dta_1s)
    PORT MAP(a=>idta_1_s,y=>dta_1_s);
s27:o_driver
    GENERIC MAP(delay_y=>   DELAYS(EDS.edsnfo(srmndx).delay_spec(1)),
        pin_data=>   ELECTRICAL_PIN_SPEC(EDS.edsnfo(srmndx).elec_spec),
        m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_srm)
    PORT MAP(a=>isrm,y=>srm);
s28:o_driver
    GENERIC MAP(delay_y=>   DELAYS(EDS.edsnfo(urc_sndx).delay_spec(1)),
        pin_data=>   ELECTRICAL_PIN_SPEC(EDS.edsnfo(urc_sndx).elec_spec),
        m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_urc_s)
    PORT MAP(a=>iurc_s,y=>urc_s);
s29:o_driver
    GENERIC MAP(delay_y=>   DELAYS(EDS.edsnfo(not_lrc_sndx).delay_spec(1)),
        pin_data=>   ELECTRICAL_PIN_SPEC(EDS.edsnfo(not_lrc_sndx).elec_spec),
        m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_not_lrcs)
    PORT MAP(a=>inot_lrc_s,y=>not_lrc_s);
s30:o_driver
    GENERIC MAP(delay_y=>   DELAYS(EDS.edsnfo(clk1ndx).delay_spec(1)),
        pin_data=>   ELECTRICAL_PIN_SPEC(EDS.edsnfo(clk1ndx).elec_spec),
        m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_clk1)
    PORT MAP(a=>iclk1,y=>clk1);
s31:o_driver
    GENERIC MAP(delay_y=>   DELAYS(EDS.edsnfo(ldl1ndx).delay_spec(1)),
        pin_data=>   ELECTRICAL_PIN_SPEC(EDS.edsnfo(ldl1ndx).elec_spec),
        m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_ldl1)
    PORT MAP(a=>ildl1,y=>ldl1);
s32:o_driver
    GENERIC MAP(delay_y=>   DELAYS(EDS.edsnfo(rdy_sndx).delay_spec(1)),
```

Figure 5.8.5-1 continued. The DQD EDS Model.

```
            pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(rdy_sndx).elec_spec),
            m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_rdy_s)
        PORT MAP(a=>irdy_s,y=>rdy_s);
    s33:o_driver
        GENERIC MAP(delay_y=>  DELAYS(EDS.edsnfo(not_dta_0_rzndx).delay_spec(1)),
            pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(not_dta_0_rzndx).elec_spec),
            m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_not_dta0rz)
        PORT MAP(a=>inot_dta_0_rz,y=>not_dta_0_rz);
    s34:o_driver
        GENERIC MAP(delay_y=>  DELAYS(EDS.edsnfo(not_dta_1_rzndx).delay_spec(1)),
            pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(not_dta_1_rzndx).elec_spec),
            m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_not_dta1rz)
        PORT MAP(a=>inot_dta_1_rz,y=>not_dta_1_rz);
    s35:o_driver
        GENERIC MAP(delay_y=>  DELAYS(EDS.edsnfo(stb_csndx).delay_spec(1)),
            pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(stb_csndx).elec_spec),
            m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_stb_cs)
        PORT MAP(a=>istb_cs,y=>stb_cs);
    s36:o_driver
        GENERIC MAP(delay_y=>  DELAYS(EDS.edsnfo(ovr_0ndx).delay_spec(1)),
            pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(ovr_0ndx).elec_spec),
            m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_ovr_0)
        PORT MAP(a=>iovr_0,y=>ovr_0);
    s37:o_driver
        GENERIC MAP(delay_y=>  DELAYS(EDS.edsnfo(not_eoc1ndx).delay_spec(1)),
            pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(not_eoc1ndx).elec_spec),
            m_gen=>m_generation,x_gen=>x_generation,wir_o=>wi_not_eoc1)
        PORT MAP(a=>inot_eoc1,y=>not_eoc1);
    s38:async_checker
        GENERIC MAP(x_generation,m_generation,
            ASYNCS(EDS.edsnfo(dta_clk_sndx).async_spec(1)))
        PORT MAP(data_in=>idta_clk_s,data_out=>odta_clk_s);
    s39:async_checker
        GENERIC MAP(x_generation,m_generation,
            ASYNCS(EDS.edsnfo(not_mclr_sndx).async_spec(1)))
        PORT MAP(data_in=>inot_mclr_s,data_out=>onot_mclr_s);
    s40:async_checker
        GENERIC MAP(x_generation,m_generation,
            ASYNCS(EDS.edsnfo(not_stb_sndx).async_spec(1)))
        PORT MAP(data_in=>inot_stb_s,data_out=>onot_stb_s);
    s41:sync_checker
        GENERIC MAP(x_generation,m_generation,
            SYNCS(EDS.edsnfo(not_ovd_sndx).sync_spec(1)))
        PORT MAP(inot_stb_s,inot_ovd_s,onot_ovd_s);
    s42:dqd_core  PORT MAP(ia,ib,ic,id,ie,i_f,ig,id32,id33,isow_s,ip_u,ildl,
        onot_ovd_s,odta_clk_s,iclk,iena_0,inot_eoc,onot_mclr_s,icnt_1,icnt_2,
        isdi,onot_stb_s,iact,inot_act,icnt_0,idta_0_s,idta_1_s,isrm,iurc_s,
        inot_lrc_s,iclk1,ildl1,irdy_s,inot_dta_0_rz,inot_dta_1_rz,istb_cs,
        iovr_0,inot_eoc1);
    END  arch_dqd_eds_str;
```

Figure 5.8.5-1 continued. The DQD EDS Model.

5.9.1.4 The variable used to select the constraint could then be carried in the generic map for the model, up to the testbench. This would allow the user to select the delay constraint implemented by the model through the generic.

5.9.1.5 The approach described here does not provide for dynamic selection of the constraint by the model. In the case of the example, either the clock-to-data valid or the reset-to-data valid constraint would be used during a given simulation. Both constraints would not be used at the same time.

5.9.2 Utilization of Multiple Output Drivers

An alternative approach to handling multiple constraints on a pin can provide a somewhat more realistic utilization of the constraints. It should be noted however, that this approach may not handle all cases of multiple constraints.

5.9.2.1 Under this approach, the engineer would employ an output driver following an input driver for the pin which exhibits the longest delays. In the case cited above, if the reset-to-data valid delay is longer than the clock-to-data valid, then the reset input would be comprised of an input driver followed immediately by an output driver.

5.9.2.2 In order to realize the appropriate constraints (in this case delays), a set of delay constraints would be established as the difference between the long delay path and the short delay path. These constraints would then be referenced through the generic map of the added output driver.

5.9.2.3 This approach provides a more realistic implementation of multiple delay constraints on a given output. The drawback to this approach lies in the fact that it may not be capable of handling all multiple constraint cases. For example, if clock-to-data valid, clock-to-carry out, reset-to-data valid and reset-to-carry out constraints are all specified, this approach will not work unless modifications are made to the core component model which allow the model to separate the data out and carry out functions.

This page intentionally left blank