Chapter 6

6.0 Modeling Electronic Assemblies

6.1 Introduction

6.1.1 Background

This chapter was established to provide some guidance and instruction for modeling electronic assemblies in VHDL. On the SHARP TIREP project, SEM format A modules were modeled. These are small circuit card assemblies, approximately 1x2 inches. Even though these CCA's are very simple, the concepts and approaches developed to model them can readily be applied to larger and more complex assemblies and systems. In the previous chapters (4 and 5) approaches to modeling digital components have been presented. In this chapter, mechanisms for modeling support circuitry and parts will be examined.

6.1.2 Objective

One of the key objectives of the TIREP project was to develop an electronic specification for the modules which could be provided to a manufacturing facility for fabrication. In order to accomplish this, it is necessary that the model contain information about components and parts used in the assembly. This includes mechanical parts, discrete capacitors, resistors, diodes, transistors, etc. and even the printed wiring board.

6.2 General Purpose Parts and Components

For all of the components described in this seciton, only those elements necessary to identify the component in the subject application will be covered. It is the concensus of the TIREP team that the development of DID compliant models for such elementry is beyond the intent of DI-EGDS-80811.

6.2.1 Mechanical Parts

Mechanical components can be captured quite simply in a model which does not include any ports. Figure 6.2.1-1 contains an example of a generic part which might be used to describe mechanical components. The generic map for this part contains only a part number, part description and a quantity. Since mechanical parts do not generally carry reference designations, it is not necessary to separately instantiate each occurrence of a part when that part is used more than once on the assembly. Mechanical parts do not provide any behavioral aspects for the CCA model.

```
--*****************************************************************
--   (c) Copyright 1994 by the
--        Naval Air Warfare Center, Aircraft Division, Indianapolis
--   Source
--        Author(s):          Charles K. Rogers
--        Organization:        NAWC-ADI
--                            Code 306, MS-42
--                            6000 E 21st St
--                            Indianapolis,  IN  46219-2189
--                            Phone:   317-353-3579
--                            EMail:   ROGERSC1@po2.nawc-ad-indy.navy.mil
--        Reference:          MIL-M-28787/175
--        Project:            SHARP   TIREP
--   DESC Certification
--        Status:             TBD
--*****************************************************************
--   Revision History
--        Version:            1.0
```

Figure 6.2.1-1. A Generic Part Model.

```
--            Date:               20 May 1994
--            Comments:           Original Release
--****************************************************************
--    Module Description
--        File:                   mec_prt.vhd
--        Module  Name(s):        partentity/arch_part_beharchitecture
--        Constraints:            none
--        Limitations:            none
--        I/O  Format(s):         none
--        Purpose and Use:        This is a generic part component which is called
--            for components which have no electrical function (e.g. mechanical
--            part). This allows the design engineer to carry a complete parts
--            list with a structural assembly model.
--        Notes:                  none
--****************************************************************
--    Standard Libraries/Packages
--        none
--    Associated Packages (order of analysis implied)
--        eia_567                  standard eia library
--    Component  Models
--        none
--    Platform:                   486/33MHz PC
--    Software/Version:           V-System for Windows, Version 3.0
--****************************************************************
--    Design Specification Elements
--        none
--****************************************************************
LIBRARY eia;
USE eia.EIA_567.ALL;
ENTITY part IS
    GENERIC   (part_num,part_desc:EIA_STRING;
-- A quantity of 0 implies the quantity is As Required (AR)
        qty:NATURAL);
    END  part;
ARCHITECTURE arch_part_beh  OF part IS
    BEGIN
    END  arch_part_beh;
```

Figure 6.2.1-1 continued. A Generic Part Model.

6.2.2 Electronic Components without Behavior

Some components (such as filter capacitors) may feature functionless behavioral architectures. A model for such a capacitor is shown in Figure 6.2.2-1. Generics are used to map such things as the part number, value and rating. An INTERFACE_CONNECTIONS list is included to clarify the pin number/pin name relationship. As noted, the architecture for this device is empty.

6.2.3 Electronic Components with Elementary Behavior

Some components may feature elementary behavior which can be modeled in VHDL. Consider the example of Figure 6.2.3-1. This figure contains a model for a resistor network used in a pull-up or pull-down mode.

```
--*****************************************************************
--   (c) Copyright 1994 by the
--        Naval Air Warfare Center, Aircraft Division, Indianapolis
--   Source
--        Author(s):            Charles K. Rogers
--        Organization:          NAWC-ADI
--                              Code 306, MS-42
--                              6000 E 21st St
--                              Indianapolis,  IN  46219-2189
--                              Phone:   317-353-3579
--                              EMail:   ROGERSC1@po2.nawc-ad-indy.navy.mil
--        Reference:            MIL-M-28787/175
--        Project:              SHARP   TIREP
--   DESC Certification
--        Status:               TBD
--*****************************************************************
--   Revision History
--        Version:              1.0
--             Date:            20 May 1994
--             Comments:        Original Release
--*****************************************************************
--   Module Description
--        File:                 capactr.vhd
--        Module  Name(s):      capacitorentity/arch_capacitor_beharchitecture
--        Constraints:          none
--        Limitations:          none
--        I/O  Format(s):       std_logic
--        Purpose and Use:      This VHDL module describes a capacitor component
--             for use in a structural VHDL model of a circuit card assembly.
--        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 capacitor IS
   GENERIC(part_num: EIA_STRING;
```

Figure 6.2.2-1. A Filter Capacitor Model.

```
            value:CAPACITANCE;
            rating: VOLTAGE);
        PORT(p1,p2: IN STD_LOGIC);
        TYPE PIN_INDEX IS (p1ndx,p2ndx);
        TYPE PIN_AND_SIGNAL_CORROLATIONS IS ARRAY(PIN_INDEX) OF PIN_RECORD;
        CONSTANT INTERFACE_CONNECTIONS:PIN_AND_SIGNAL_CORROLATION:=(
            p1ndx=>(pin_id=>"1                                        ",
                signal_name=>"+  (if  used)                           "),
            p2ndx=>(pin_id=>"2                                        ",
                signal_name=>"-  (if  used)                           "));
        END capacitor;
    ARCHITECTURE arch_capacitor_beh OF capacitor IS
        BEGIN
        END  arch_capacitor_beh;
```

Figure 6.2.2-1 continued. A Filter Capacitor Model.

6.2.3.1 In this model, it is assumed that the common input will be driven to either a '1' or '0' state. The outputs will then be resistive 'H' or 'L' respectively. In the event that the input is not forced either high or low, the outputs will be driven to a 'W' state.

6.2.3.2 As with the capacitor model presented above, this model also contains the declaration of an INTERFACE_CONNECTIONS constant which links pin numbers to pin names.

6.2.3.3 Discrete solid-state components (such as diodes, transistors, etc.) may be modeled in a similar manner where appropriate.

6.2.4 Connector Modeling

There are a couple of fashions in which connectors can be modeled. One approach would be to model the connector as an input only device on the circuit card assembly, similar to the filter capacitor described above. An alternative approach would be to model the connector as a set of inputs and outputs which are defined in accordance with the circuit in which the connector is used. Under this approach, a generic delay could be associated with the transfer of signals through the connector. Since vectors are not permitted as ports, each pin on the connector must be individually declared in the port statement (see EIA-567, paragraph 3.3.1, item 2).

6.2.4.1 The connector model of Figure 6.2.4.1-1 is by far the simplest, however, it is also the least accurate in the application. Additionally, this connector will not introduce any resolution problems when trying to model bidirectional signals through the connector.

6.2.4.2 The connector model of Figure 6.2.4.2-1 is the more accurate model of the connector. In this model, the signals into and out of the CCA are passed through the connector. When appropriate, this model will also attach a timing delay to the signals as they are passed through the model.

```
--*****************************************************************
--  (c) Copyright 1994 by the
--      Naval Air Warfare Center, Aircraft Division, Indianapolis
--  Source
--      Author(s):          Charles K. Rogers
--      Organization:       NAWC-ADI
```

Figure 6.2.3-1. A Resistor Network Model.

```
--                          Code 306, MS-42
--                          6000 E 21st St
--                          Indianapolis,  IN  46219-2189
--                          Phone:   317-353-3579
--                          EMail:   ROGERSC1@po2.nawc-ad-indy.navy.mil
--      Reference:          MIL-M-28787/175
--      Project:            SHARP   TIREP
--   DESC Certification
--      Status:             TBD
--****************************************************************
--   Revision History
--      Version:            1.0
--          Date:           20 May 1994
--          Comments:       Original Release
--****************************************************************
--   Module Description
--      File:               res_net.vhd
--      Module  Name(s):    resistor_network entity/arch_resistor_network_beh
--          architecture
--      Constraints:        none
--      Limitations:        none
--      I/O  Format(s):     std_logic
--      Purpose and Use:    This VHDL module describes a resistor network
--          component for use in a structural VHDL model of a circuit card
--          assembly.
--      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 resistor_network  IS
   GENERIC(part_num: EIA_STRING;
      value: RESISTANCE;
      rating: POWER);
   PORT(com: IN STD_LOGIC;
      p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13: OUT STD_LOGIC);
```

Figure 6.2.3-1 continued. A Resistor Network Model.

```
    TYPE PIN_INDEX IS (comndx,p1ndx,p2ndx,p3ndx,p4ndx,p5ndx,p6ndx,p7ndx,
      p8ndx,p9ndx,p10ndx,p11ndx,p12ndx,p13ndx);
    TYPE PIN_AND_SIGNAL_CORROLATION IS ARRAY(PIN_INDEX) OF PIN_RECORD;
    CONSTANT INTERFACE_CONNECTIONS:PIN_AND_SIGNAL_CORROLATION:=(
      comndx=>(pin_id=>"14                    ",
        signal_name=>"common                  "),
      p1ndx=>(pin_id=>"1                      ",
        signal_name=>"p1                      "),
      p2ndx=>(pin_id=>"2                      ",
        signal_name=>"p2                      "),
      p3ndx=>(pin_id=>"3                      ",
        signal_name=>"p3                      "),
      p4ndx=>(pin_id=>"4                      ",
        signal_name=>"p4                      "),
      p5ndx=>(pin_id=>"5                      ",
        signal_name=>"p5                      "),
      p6ndx=>(pin_id=>"6                      ",
        signal_name=>"p6                      "),
      p7ndx=>(pin_id=>"7                      ",
        signal_name=>"p7                      "),
      p8ndx=>(pin_id=>"8                      ",
        signal_name=>"p8                      "),
      p9ndx=>(pin_id=>"9                      ",
        signal_name=>"p9                      "),
      p10ndx=>(pin_id=>"10                    ",
        signal_name=>"p10                     "),
      p11ndx=>(pin_id=>"11                    ",
        signal_name=>"p11                     "),
      p12ndx=>(pin_id=>"12                    ",
        signal_name=>"p12                     "),
      p13ndx=>(pin_id=>"13                    ",
        signal_name=>"p13                     "));
    END resistor_network;
  ARCHITECTURE arch_resistor_network_beh OF resistor_network IS
    BEGIN
    PROCESS(com)
      BEGIN
      IF com='1' THEN
          p1<='H';p2<='H';p3<='H';p4<='H';p5<='H';p6<='H';p7<='H';
          p8<='H';p9<='H';p10<='H';p11<='H';p12<='H';p13<='H';
        ELSIF com='0' THEN
          p1<='L';p2<='L';p3<='L';p4<='L';p5<='L';p6<='L';p7<='L';
          p8<='L';p9<='L';p10<='L';p11<='L';p12<='L';p13<='L';
        ELSE
          p1<='W';p2<='W';p3<='W';p4<='W';p5<='W';p6<='W';p7<='W';
          p8<='W';p9<='W';p10<='W';p11<='W';p12<='W';p13<='W';
        END IF;
      END PROCESS;
    END arch_resistor_network_beh;
```

Figure 6.2.3-1 continued. A Resistor Network Model.

```
--****************************************************************
--   (c) Copyright 1994 by the
--        Naval Air Warfare Center, Aircraft Division, Indianapolis
--   Source
--        Author(s):            Charles K. Rogers
--        Organization:           NAWC-ADI
--                              Code 306, MS-42
--                              6000 E 21st St
--                              Indianapolis,  IN  46219-2189
--                              Phone:   317-353-3579
--                              EMail:   ROGERSC1@po2.nawc-ad-indy.navy.mil
--        Reference:            MIL-M-28787/175
--        Project:              SHARP   TIREP
--   DESC Certification
--        Status:               TBD
--****************************************************************
--   Revision History
--        Version:              1.0
--             Date:            23 May 1994
--             Comments:        Original Release
--****************************************************************
--   Module Description
--        File:                 connctr.vhd
--        Module  Name(s):      connectorentity/arch_connector_opt1_beh
--             architecture
--        Constraints:          none
--        Limitations:          none
--        I/O  Format(s):       std_logic
--        Purpose and Use:      This VHDL module describes a generic connector.
--             In this implementation, the connector is solely an input device
--             and only the pins on the CCA side of the connector are referenced.
--        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
--        part_num                     generic
--        port                         declaration
--****************************************************************
LIBRARY ieee;
LIBRARY eia;
USE ieee.STD_LOGIC_1164ALL;
```

Figure 6.2.4.1-1. An Input Only Connector Model.

```
USE eia.EIA_567.ALL;
ENTITY connector IS
--*****************************************************************
-- The "part_num" should reflect the number of the connector actually used
-- in the design.
--*****************************************************************
    GENERIC(part_num: EIA_STRING:="m28754/8-01                    ");
--*****************************************************************
-- The port reflects the number of pins available in the conector.          Since
-- this device is input only, just one side of the connector is called out.
--*****************************************************************
    PORT(p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,p17,p18,p19,
        p20,p21,p22,p23,p24,p25,p26,p27,p28,p29,p30,p31,p32,p33,p34,p35,
        p36,p37,p38,p39,p40:IN STD_LOGIC);
    END connector;
ARCHITECTURE arch_connector_opt1_beh OF connector IS
    BEGIN
    END  arch_connector_opt1_beh;
```

Figure 6.2.4.1-1 continued. An Input Only Connector.

```
--*****************************************************************
--  (c) Copyright 1994 by the
--      Naval Air Warfare Center, Aircraft Division, Indianapolis
--  Source
--      Author(s):          Charles K. Rogers
--      Organization:        NAWC-ADI
--                          Code 306, MS-42
--                          6000 E 21st St
--                          Indianapolis, IN 46219-2189
--                          Phone:   317-353-3579
--                          EMail:   ROGERSC1@po2.nawc-ad-indy.navy.mil
--      Reference:          MIL-M-28787/175
--      Project:            SHARP   TIREP
--  DESC Certification
--      Status:             TBD
--*****************************************************************
--  Revision History
--      Version:            1.0
--          Date:           20 May 1994
--          Comments:       Original Release
--*****************************************************************
--  Module Description
--      File:               connctr.vhd
--      Module  Name(s):    connectorentity/arch_connector_opt2_beh
--          architecture
```

Figure 6.2.4.2-1. An Input/Output Connector Model.

```
--        Constraints:          none
--        Limitations:          none
--        I/O  Format(s):       std_logic
--        Purpose and Use:      This VHDL module describes the connector used
--             for the DQD standard electronic module.         The connector has two
--             sides (a and b).   Side a is the circuit card side, while side b
--             is the system side.A generic time is included to reflect delay
--             through the connector.
--        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
--        entire file
--*****************************************************************
LIBRARY ieee;
LIBRARY eia;
USE ieee.STD_LOGIC_1164.ALL;
USE eia.EIA_567.ALL;
ENTITY connector  IS
--*****************************************************************
-- The "part_num" generic identifies the part number for the connector.
-- the "c_del" generic is used for the propagation delay through the
-- connector.
--*****************************************************************
    GENERIC(part_num: EIA_STRING:="m28754/8-01                    ";
        c_del: TIME:=0  ns);
--*****************************************************************
-- Each pin on the connector must be declared as either an input or an
-- output.
--*****************************************************************
    PORT(a2,a3,a4,a5,a6,a11,a16,a17,a20,a21,a23,a26,a28,a31,a35,a37,a38,
        b1,b7,b8,b9,b10,b12,b13,b14,b15,b18,b19,b22,b24,b25,b27,b29,b30,
        b32,b33,b34,b36,b39,b40:IN STD_LOGIC;
        a1,a7,a8,a9,a10,a12,a13,a14,a15,a18,a19,a22,a24,a25,a27,a29,a30,
        a32,a33,a34,a36,a39,a40,b2,b3,b4,b5,b6,b11,b16,b17,b20,b21,b23,
        b26,b28,b31,b35,b37,b38:OUT STD_LOGIC);
    END  connector;
ARCHITECTURE arch_connector_opt2_beh  OF  connector  IS
    BEGIN
--*****************************************************************
-- Each signal assignment must be independently made for each pin in
```

Figure 6.2.4.2-1 continued. An Input/Output Connector.

```
-- the connector.
--************************************************************
      a1<=b1    AFTER c_del;
      a7<=b7    AFTER c_del;
      a8<=b8    AFTER c_del;
      a9<=b9    AFTER c_del;
      a10<=b10    AFTER c_del;
      a12<=b12    AFTER c_del;
      a13<=b13    AFTER c_del;
      a14<=b14    AFTER c_del;
      a15<=b15    AFTER c_del;
      a18<=b18    AFTER c_del;
      a19<=b19    AFTER c_del;
      a22<=b22    AFTER c_del;
      a24<=b24    AFTER c_del;
      a25<=b25    AFTER c_del;
      a27<=b27    AFTER c_del;
      a29<=b29    AFTER c_del;
      a30<=b30    AFTER c_del;
      a32<=b32    AFTER c_del;
      a33<=b33    AFTER c_del;
      a34<=b34    AFTER c_del;
      a36<=b36    AFTER c_del;
      a39<=b39    AFTER c_del;
      a40<=b40    AFTER c_del;
      b2<=a2    AFTER c_del;
      b3<=a3    AFTER c_del;
      b4<=a4    AFTER c_del;
      b5<=a5    AFTER c_del;
      b6<=a6    AFTER c_del;
      b11<=a11    AFTER c_del;
      b16<=a16    AFTER c_del;
      b17<=a17    AFTER c_del;
      b20<=a20    AFTER c_del;
      b21<=a21    AFTER c_del;
      b23<=a23    AFTER c_del;
      b26<=a26    AFTER c_del;
      b28<=a28    AFTER c_del;
      b31<=a31    AFTER c_del;
      b35<=a35    AFTER c_del;
      b37<=a37    AFTER c_del;
      b38<=a38    AFTER c_del;
      END   arch_connector_opt2_beh;
```

Figure 6.2.4.2-1 continued. An Input/Output Connector.

6.2.4.3 For the input only connector, unused pins would be modeled as inputs with the pins left "open" (or unconnected). For the input/output connector, unused pins might be assigned based upon a user defined convention (i.e. unused pins on the system side of the connector are inputs and on the CCA side are outputs). When instantiated, unused pins on the connector would be left "open" (or unconnected) in the structural model of the design.

6.2.5 Modeling Printed Circuit Boards

The final component we will examine is the printed circuit board (pcb). This device poses some special problems to the circuit card assembly model in that it is necessary to somehow describe physical aspects of this part.

6.2.5.1 In order to accomplish the task of modeling the pcb, a graphics package was developed as shown in Figure 6.2.5.1-1. In this package elementary graphics elements are defined for use in the model of a mechanical part. Following are the data type declarations defined in this package.

Declaration	Data Type
length	physical
cor_pnt	record
line	record
arc	record
hole	record
origin	record
area	record

6.2.5.1.1 The "length" declaration defines a standard unit of measure. Appropriate declarations are included for both english and metric units of length, however the metric declaration is commented mou making the english units the one employed for this example. Due to limitations imposed on the declarations of physical types, an attempt to define both metric and english units of length in the same package has not been made.

6.2.5.1.2 A "cor_pnt" is a coordinate point. This record is used to declare a point in the X-Y plane. This record contains two elements of type "length". A "cor_pnt" is used to define other graphic elements. The "cor_pnt" may also be used to explictly locate components or parts in a model.

6.2.5.1.3 A "line" is defined by two points ("cor_pnt"). The first point will be known as the beginning point (p1) and the second point will be known as the ending point (p2). A "line" does not extend beyond the beginning or ending point.

6.2.5.1.4 An "arc" is defined as a center point (c), a beginning point (p1) and and ending point (p2). The distance from "c" to "p1" must be equal to the distance from "c" to "p2". This is the radius of the arc. The arc is generated in a clockwise direction from "p1" to "p2" at the radius established by the center point (c).

6.2.5.1.5 A "hole" is defined by a center point (c) and a diameter (d). This data type may be used to represent any circular object.

```
--****************************************************************
--   (c) Copyright 1994 by the
--        Naval Air Warfare Center, Aircraft Division, Indianapolis
--   Source
--        Author(s):           Charles K. Rogers
--        Organization:          NAWC-ADI
--                             Code 306, MS-42
--                             6000 E 21st St
--                             Indianapolis,  IN  46219-2189
--                             Phone:   317-353-3579
--                             EMail:   ROGERSC1@po2.nawc-ad-indy.navy.mil
--        Reference:           MIL-M-28787/175
--        Project:             SHARP   TIREP
```

Figure 6.2.5.1-1. A Graphics Package.

```
--    DESC Certification
--          Status:                 TBD
--***************************************************************
--    Revision History
--          Version:                1.0
--                Date:             20 May 1994
--                Comments:         Original Release
--***************************************************************
--    Module Description
--          File:                   graphicp.vhd
--          Module  Name(s):        graphicspackage
--          Constraints:            none
--          Limitations:            none
--          I/O  Format(s):         none
--          Purpose and Use:        This package contains the declarations necessary
--                to describe a graphical entity (such as a printed wiring board
--                form factor). The dimensions provided in the entity description
--                shall be nominal.
--          Notes:                  none
--***************************************************************
--    Standard Libraries/Packages
--          none
--    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
--***************************************************************
PACKAGE graphics IS
---------------------------------------------------------------------
-- when defining a graphic entity, the physical unit of length is
-- required
---------------------------------------------------------------------
TYPE length IS RANGE -1e8 TO 1e8
      UNITS
      mic;                      -- microinch
      mil=1000  mic;            -- milliinch
      inch=1000  mil;    -- inch
      ft=12   inch;             -- foot
      yd=3   ft;                -- yard
      END UNITS;
---------------------------------------------------------------------
-- alternate length declaration for metric
---------------------------------------------------------------------
-- type length is range -1e8 to 1e8
```

Figure 6.2.5.1-1 continued. A Graphics Package.

```
--          units
--          um;                      -- micrometer
--          mm=1000  um;        -- millimeter
--          cm=10  mm;          -- centimeter
--          m=100  cm;          -- meter
--          end units;
---------------------------------------------------------------------
-- a point definition
---------------------------------------------------------------------
TYPE cor_pnt IS RECORD
    x,y:length;
    END RECORD;
---------------------------------------------------------------------
-- a line segment is defined by two points
---------------------------------------------------------------------
TYPE LINE IS RECORD
    p1,p2:cor_pnt;
    END RECORD;
---------------------------------------------------------------------
-- an arc is defined by three points, a center point, a starting point
-- and an ending point.   point 1 (p1) is the beginning of the arc which
-- is drawn to point 2 (p2) in a clockwise direction.
---------------------------------------------------------------------
TYPE arc IS RECORD
    c,p1,p2:cor_pnt;
    END RECORD;
---------------------------------------------------------------------
-- a hole is defined by center point and a diameter.   The function of
-- defined holes should be described in comments of the model
---------------------------------------------------------------------
TYPE hole IS RECORD
    c:cor_pnt;
    d:length;
    END RECORD;
---------------------------------------------------------------------
-- the origin is defined by a point.   if an origin is not
-- declared, the point (0.0, 0.0) will be assumed.
---------------------------------------------------------------------
TYPE origin IS RECORD
    x,y:length;
    END RECORD;
---------------------------------------------------------------------
-- an area is a rectangle.   it is defined by two points which represent
-- the opposing corners of the rectangle.
---------------------------------------------------------------------
TYPE area IS RECORD
    p1,p2:cor_pnt;
    END RECORD;
END graphics;
```

Figure 6.2.5.1-1 continued. A Graphics Package.

6.2.5.1.6 The "origin" is defined as a point in the same manner that "cor_pnt" was declared. The orgin is a unique point in that dimensioning for an object is ultimately referenced back to this point. For this reason, the "origin" receives it’s own declaration.

6.2.5.1.7 Finally, "area" is declared. An area is defined by the opposing corners of a rectangle. Areas are useful for sepcifying the location of certain parts or components. "Areas" may also be used to identify locations where components may not be placed. Since an "area" is defined by two points, the X and Y components of the points must be different.

6.2.5.2 With this package defined, it is possible to generate a description for the form factor of a printed wiring board. Figure 6.2.5.2-1 shows the pcb component for the DQD module. In this example, the desired pcb layout and form factor information are captured in generics for this component. If these generics are plotted out, Figure 6.2.5.2-2 is obtained.

```
--*****************************************************************
--   (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_pcb.vhd
--        Module  Name(s):      dqd_pcb entity/arch_dqd_pcb_beh architecture
--        Constraints:          none
--        Limitations:          none
--        I/O  Format(s):       none
--        Purpose and Use:      This model provides the basic requirements
--             for the M28787/212 printed wiring board. The attached generics
--             provide the pcb form factor along with routing restrictions and
--             requirements.      This component model is approximate. Inadequate
--             detail exists to explictly model this board.
--        Notes:                none
--*****************************************************************
--   Standard Libraries/Packages
--        none
```

Figure 6.2.5.2-1. A Printed Circuit Board Model.

```
--    Associated Packages (order of analysis implied)
--     eia_567                    standard EIA-567 package
--     graphics                   general purpose graphics 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;
USE  work.graphics.ALL;
ENTITY dqd_pcb  IS
--***************************************************************
-- Call out graphics components as required to describe information
-- about the printed circuit board.
--***************************************************************
   GENERIC(part_num: EIA_STRING:="mm28787175pcb                 ";
         bd_matl: EIA_STRING:="mil-p-13949/4 sfn0620 b28              ";
         max_bd_thck:length:=0.031   inch;
         min_cond_width:length:=0.008    inch;
         min_cond_space:length:=0.008    inch;
         min_copper_thck:length:=0.0018    inch;
-----------------------------------------------------------------------
-- board outline
-----------------------------------------------------------------------
         o1:origin:=(0.100  inch,0.100  inch);
         l1:LINE:=((-0.010 inch,0.140 inch),(-0.100 inch,0.230 inch));
         l2:LINE:=((-0.100 inch,0.230 inch),(-0.100 inch,1.200 inch));
         l3:LINE:=((-0.100 inch,1.200 inch),(2.200 inch,1.200 inch));
         l4:LINE:=((2.200 inch,1.200 inch),(2.200 inch,0.230 inch));
         l5:LINE:=((2.200 inch,0.230 inch),(2.110 inch,0.140 inch));
         l6:LINE:=((2.110 inch,0.140 inch),(2.050 inch,0.140 inch));
         l7:LINE:=((2.050 inch,0.140 inch),(2.050 inch,0.050 inch));
         l8:LINE:=((2.050 inch,0.050 inch),(0.050 inch,0.050 inch));
         l9:LINE:=((0.050 inch,0.050 inch),(0.050 inch,0.140 inch));
         l10:LINE:=((0.050 inch,0.140 inch),(-0.010 inch,0.140 inch));
-----------------------------------------------------------------------
-- connector pin 1 reference hole
-----------------------------------------------------------------------
         h1:hole:=((0.100  inch,0.100  inch),0.062  inch);
-----------------------------------------------------------------------
-- mounting holes
-----------------------------------------------------------------------
         h2:hole:=((-0.025  inch,1.100  inch),0.094  inch);
         h3:hole:=((1.050  inch,1.100  inch),0.094  inch);
         h4:hole:=((2.125  inch,1.100  inch),0.094  inch);
```

Figure 6.2.5.2-1 continued. A Printed Circuit Board.

```
-----------------------------------------------------------------------
-- the area outlined by the following shall be free of components and
-- routings
-----------------------------------------------------------------------
        l11:LINE:=((-0.010 inch,0.140 inch),(-0.100 inch,0.230 inch));
        l12:LINE:=((-0.100 inch,0.230 inch),(-0.100 inch,1.200 inch));
        l13:LINE:=((-0.100 inch,1.200 inch),(2.200 inch,1.200 inch));
        l14:LINE:=((2.200 inch,1.200 inch),(2.200 inch,0.230 inch));
        l15:LINE:=((2.200 inch,0.230 inch),(2.110 inch,0.140 inch));
        l16:LINE:=((2.110 inch,0.140 inch),(2.100 inch,0.140 inch));
        l17:LINE:=((2.100 inch,0.140 inch),(2.100 inch,1.000 inch));
        l18:LINE:=((2.100 inch,1.000 inch),(2.000 inch,1.100 inch));
        l19:LINE:=((2.000 inch,1.100 inch),(1.200 inch,1.100 inch));
        a1:arc:=((1.050 inch,1.100 inch),(1.200 inch,1.100 inch),(0.900 inch,1.100 inch));
        l20:LINE:=((0.900 inch,1.100 inch),(0.100 inch,1.100 inch));
        l21:LINE:=((0.100 inch,1.100 inch),(0.000 inch,1.000 inch));
        l22:LINE:=((0.000 inch,1.000 inch),(0.000 inch,0.140 inch));
        l23:LINE:=((0.000 inch,0.140 inch),(-0.010 inch,0.140 inch)));
    END dqd_pcb;
ARCHITECTURE arch_dqd_pcb_beh OF dqd_pcb IS
    BEGIN
    END  arch_dqd_pcb_beh;
```

Figure 6.2.5.2-1 continued. A Printed Circuit Board.

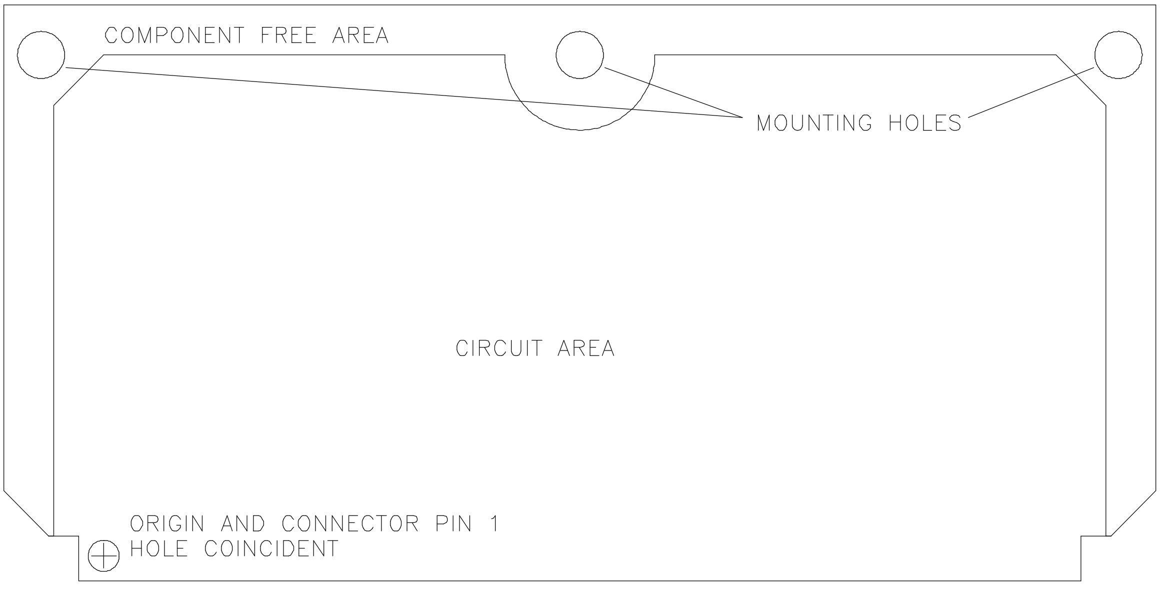

Figure 6.2.5.2-2. A Printed Circuit Board Outline.

6.2.5.3 Under this approach, a manufacturing site could take this model to develop the pcb layout. The actual component placements and routings (apart from those defined in the model) would be at the discretion of the layout tools and engineers.

6.2.5.4 The pcb modeling approach presented herein is under investigation. A known limitation for this model is that it does not carry any tolerancing information with the design. As this approach is refined, this document will be updated to reflect the findings.

6.3 EDS Components

DID compliant VHDL models will generally be required for all microelectronic devices, whether simple gates or complex microprocessors. This model will be the EDS model as described in chapters 4 or 5.

6.3.1 Inherited Timing

In a top-down design approach, it is desired that an assembly level model "inherit" the component timing from the component models. The assembly level model would simply implement the timing associated with routing delays for the module.

6.3.2 Zero Timing

In the case of the TIREP project, this introduces a problem due to the fact that the design is developed from a bottom up or reverse engineering position. In other words, the point of reference on the TIREP project is not the component, but the module specification. When a design engineer selects a component for a module, special care must be taken to insure that the timing characteristics of the component do not fall outside the timing requirements of the module. The module timing, in this case, includes the component level timing. If the component carried it's timing into the assembly model, then the component timing would be added to the module timing, creating delays which no longer comply with the module requirements.

6.3.2.1 In order to deal with this problem, it is recommended that the OPERATING_SELECTION type declared in the EIA_567 library or in the EIA_567_EV package, be expanded to include the "tzero" operating point as shown below. This added operating selection point is added to the end of the enumerated list. Since this element is not required under either the EIA-567 or DID-EGDS-80811, it is not desired to have this as the default (or first) element in the list.

```
TYPE OPERATING_SELECTION IS (TMIN, TNOM, TMAX, tzero);
```

6.3.2.2 With this done, it is necessary only to add the "tzero" operating point to the electronic data sheet. When this component is used in the assembly, the "tzero" operating point can be imposed. The "tzero" operating point would be one in which all outputs are assigned delay elements of class 0. Following is an example of the "tzero" operating point for the DQD module example previously discussed. It is not necessary to zero out the synchronous and asynchronous constraint checking, however, leaving it in could potentially create some confusing messages since input, multiple constraint checkers would be used.

```
-- The nominal or typical operating point
        OPERATING_SELECTION'pos(tzero)=>(
            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),
    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=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>1),
    dta_clk_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        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=>(0)),
        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=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>1),
    actndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>16),
    not_actndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>16),
    cnt_0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>16),
    dta_0_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>16),
    dta_1_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>16),
    srmndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>15),
    urc_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>15),
    not_lrc_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>14),
    clk1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>11),
    ldl1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>13),
    rdy_sndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>12),
    not_dta_0_rzndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
        delay_spec=>(1=>(0)),elec_spec=>12),
```

```
not_dta_1_rzndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
     delay_spec=>(1=>(0)),elec_spec=>12),
stb_csndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
     delay_spec=>(1=>(0)),elec_spec=>12),
ovr_0ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
     delay_spec=>(1=>(0)),elec_spec=>12),
not_eoc1ndx=>(sync_spec=>(1=>(0)),async_spec=>(1=>(0)),
     delay_spec=>(1=>(0)),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))),
```

6.3.3 Figure 6.3.3-1 contains the code for an EDS component for use on the DQD module. This component is based upon an Altera EPM5128. In this model, it is appropriate to combine the EDS model with the component model. There is no value added by keeping these models separated. This model is based upon the "dqd_core" component which was introduced in chapter 5.

```
--****************************************************************
--   (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_cmp.vhd
--        Module  Name(s):    dqd_comp entity/arch_dqd_comp_str architecture
--        Constraints:        none
--        Limitations:        none
--        I/O  Format(s):     std_logic
--        Purpose and Use:    This VHDL module describes a component for the
--             DQD standard hardware module in accordance with M28787/212.This
--             module implements a standard serial output circuit.
--        Notes:              none
```

Figure 6.3.3-1. An EDS Component Model.

```
--*****************************************************************
--   Standard Libraries/Packages
--        std_logic_1164          standard multi-value logic package
--   Associated Packages (order of analysis implied)
--        eia_567                 standard eia package
--        dqdcds                  DQD component design specification package
--        eia_567cds              eia component design specification package
--   Component  Models
--        dqd_core                     behavorial VHDL module core
--        i_driver                     generic input driver
--        o_driver                     generic output driver
--        asynchronous_checker         generic asynchronous checker
--        synchronous_checker          generic synchronous checker
--   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.dqdcds.ALL;
USE work.eia_567cds.ALL;
ENTITY dqd_comp 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,
        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:cmp_point:=(selection=>tzero,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;
--*******************************************************************
```

Figure 6.3.3-1 continued. An EDS Component Model.

```
-- 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;
        part_num: EIA_STRING:="epm5128                              ");
    PORT(in1,in2,in3,in4,in5,in6,in7,in8:IN STD_LOGIC;
        io1,io2,io3,io4,io5,io6,io7,io8,io9,io10,io11,io12,io13:INOUT STD_LOGIC;
        io14,io15,io16,io17,io18,io19,io20,io21,io22,io23:INOUT STD_LOGIC;
        io24,io25,io26,io27,io28,io29,io30,io31,io32,io33:INOUT STD_LOGIC;
        io34,io35,io36,io37,io38,io39,io40,io41,io42,io43:INOUT STD_LOGIC;
        io44,io45,io46,io47,io48,io49,io50,io51,io52:INOUT STD_LOGIC;
        vcc1,vcc2,vcc3,vcc4,gnd1,gnd2,gnd3,gnd4:IN STD_LOGIC);
    END dqd_comp;
ARCHITECTURE arch_dqd_comp_str OF dqd_comp 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:  OUT STD_LOGIC;
            ldl1,rdy_s,not_dta_0_rz,not_dta_1_rz,stb_cs,ovr_0,not_eoc1:  OUT 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;
            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;
```

Figure 6.3.3-1 continued. An EDS Component Model.

```
            m_gen:SEVERITY_LEVEL:= WARNING;
            syncconstraint: SYNC);
        PORT(clk,data_in: IN STD_LOGIC;
            data_out: OUT STD_LOGIC:='U');
        END COMPONENT;
    SIGNAL a,b,c,d,e,F,g,d32,d33,sow_s,p_u,ldl,not_ovd_s: STD_LOGIC;
    SIGNAL  dta_clk_s,clk,ena_0,not_eoc,not_mclr_s,cnt_1,cnt_2:STD_LOGIC;
    SIGNAL  sdi,not_stb_s,act,not_act,cnt_0,dta_0_s,dta_1_s,srm:STD_LOGIC;
    SIGNAL  urc_s,not_lrc_s,clk1,ldl1,rdy_s,not_dta_0_rz:STD_LOGIC;
    SIGNAL  not_dta_1_rz,stb_cs,ovr_0,not_eoc1,idta_clk_s:STD_LOGIC;
    SIGNAL  inot_mclr_s,inot_stb_s,inot_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.
        in1ndx=>(pin_id=>"1                                   ",
            signal_name=>"in1    (p_u)                        "),
        in2ndx=>(pin_id=>"2                                   ",
            signal_name=>"in2    (not_ovd_s)                  "),
        vcc1ndx=>(pin_id=>"3                                  ",
            signal_name=>"vcc    (vcc)                        "),
        io1ndx=>(pin_id=>"4                                   ",
            signal_name=>"io1    (n/c)                        "),
        io2ndx=>(pin_id=>"5                                   ",
            signal_name=>"io2    (n/c)                        "),
        io3ndx=>(pin_id=>"6                                   ",
            signal_name=>"io3    (not_lrc_s)                  "),
        io4ndx=>(pin_id=>"7                                   ",
            signal_name=>"io4    (not_eoc1)                   "),
        io5ndx=>(pin_id=>"8                                   ",
            signal_name=>"io5    (dta_1_s)                    "),
        io6ndx=>(pin_id=>"9                                   ",
            signal_name=>"io6    (cnt_0)                      "),
        io7ndx=>(pin_id=>"10                                  ",
            signal_name=>"io7    (n/c)                        "),
        io8ndx=>(pin_id=>"11                                  ",
```

Figure 6.3.3-1 continued. An EDS Component Model.

```
      signal_name=>"io8    (n/c)                              "),
io9ndx=>(pin_id=>"12                                          ",
      signal_name=>"io9    (n/c)                              "),
io10ndx=>(pin_id=>"13                                         ",
      signal_name=>"io10    (n/c)                             "),
io11ndx=>(pin_id=>"14                                         ",
      signal_name=>"io11    (n/c)                             "),
io12ndx=>(pin_id=>"15                                         ",
      signal_name=>"io12    (n/c)                             "),
gnd1ndx=>(pin_id=>"16                                         ",
      signal_name=>"gnd        (gnd)                          "),
io13ndx=>(pin_id=>"17                                         ",
      signal_name=>"io13    (n/c)                             "),
io14ndx=>(pin_id=>"18                                         ",
      signal_name=>"io14    (n/c)                             "),
io15ndx=>(pin_id=>"19                                         ",
      signal_name=>"io15    (rdy_s)                           "),
vcc2ndx=>(pin_id=>"20                                         ",
      signal_name=>"vcc    (vcc)                              "),
io16ndx=>(pin_id=>"21                                         ",
      signal_name=>"io16    (n/c)                             "),
io17ndx=>(pin_id=>"22                                         ",
      signal_name=>"io17    (n/c)                             "),
io18ndx=>(pin_id=>"23                                         ",
      signal_name=>"io18    (n/c)                             "),
io19ndx=>(pin_id=>"24                                         ",
      signal_name=>"io19    (n/c)                             "),
io20ndx=>(pin_id=>"25                                         ",
      signal_name=>"io20    (n/c)                             "),
io21ndx=>(pin_id=>"26                                         ",
      signal_name=>"io21    (n/c)                             "),
io22ndx=>(pin_id=>"27                                         ",
      signal_name=>"io22     (not_dta_1_rz)                   "),
io23ndx=>(pin_id=>"28                                         ",
      signal_name=>"io23     (not_dta_0_rz)                   "),
io24ndx=>(pin_id=>"29                                         ",
      signal_name=>"io24     (not_act)                        "),
io25ndx=>(pin_id=>"30                                         ",
      signal_name=>"io25     (dta_0_s)                        "),
io26ndx=>(pin_id=>"31                                         ",
      signal_name=>"io26     (act)                            "),
in3ndx=>(pin_id=>"32                                          ",
      signal_name=>"in3    (not_mclr_s)                       "),
gnd2ndx=>(pin_id=>"33                                         ",
      signal_name=>"gnd        (gnd)                          "),
in4ndx=>(pin_id=>"34                                          ",
      signal_name=>"in4    (not_eoc)                          "),
in5ndx=>(pin_id=>"35                                          ",
      signal_name=>"in5    (ldl)                              "),
```

Figure 6.3.3-1 continued. An EDS Component Model.

```
in6ndx=>(pin_id=>"36                                    ",
      signal_name=>"in6    (dta_clk_s)                  "),
vcc3ndx=>(pin_id=>"37                                   ",
      signal_name=>"vcc    (vcc)                        "),
io27ndx=>(pin_id=>"38                                   ",
      signal_name=>"io27    (n/c)                       "),
io28ndx=>(pin_id=>"39                                   ",
      signal_name=>"io28    (n/c)                       "),
io29ndx=>(pin_id=>"40                                   ",
      signal_name=>"io29    (sow_s)                     "),
io30ndx=>(pin_id=>"41                                   ",
      signal_name=>"io30    (sdi)                       "),
io31ndx=>(pin_id=>"42                                   ",
      signal_name=>"io31    (not_stb_s)                 "),
io32ndx=>(pin_id=>"43                                   ",
      signal_name=>"io32    (g)                         "),
io33ndx=>(pin_id=>"44                                   ",
      signal_name=>"io33    (F)                         "),
io34ndx=>(pin_id=>"45                                   ",
      signal_name=>"io34    (ena_0)                     "),
io35ndx=>(pin_id=>"46                                   ",
      signal_name=>"io35    (stb_cs)                    "),
io36ndx=>(pin_id=>"47                                   ",
      signal_name=>"io36    (n/c)                       "),
io37ndx=>(pin_id=>"48                                   ",
      signal_name=>"io37    (n/c)                       "),
io38ndx=>(pin_id=>"49                                   ",
      signal_name=>"io38    (n/c)                       "),
gnd3ndx=>(pin_id=>"50                                   ",
      signal_name=>"gnd        (gnd)                    "),
io39ndx=>(pin_id=>"51                                   ",
      signal_name=>"io39    (ldl1)                      "),
io40ndx=>(pin_id=>"52                                   ",
      signal_name=>"io40    (n/c)                       "),
io41ndx=>(pin_id=>"53                                   ",
      signal_name=>"io41    (urc_s)                     "),
vcc4ndx=>(pin_id=>"54                                   ",
      signal_name=>"vcc    (vcc)                        "),
io42ndx=>(pin_id=>"55                                   ",
      signal_name=>"io42    (srm)                       "),
io43ndx=>(pin_id=>"56                                   ",
      signal_name=>"io43    (ovr_0)                     "),
io44ndx=>(pin_id=>"57                                   ",
      signal_name=>"io44    (clk1)                      "),
io45ndx=>(pin_id=>"58                                   ",
      signal_name=>"io45    (e)                         "),
io46ndx=>(pin_id=>"59                                   ",
      signal_name=>"io46    (d33)                       "),
io47ndx=>(pin_id=>"60                                   ",
```

Figure 6.3.3-1 continued. An EDS Component Model.

```
            signal_name=>"io47    (d32)                         "),
        io48ndx=>(pin_id=>"61                                   ",
            signal_name=>"io48    (d)                           "),
        io49ndx=>(pin_id=>"62                                   ",
            signal_name=>"io49    (cnt_2)                       "),
        io50ndx=>(pin_id=>"63                                   ",
            signal_name=>"io50    (c)                           "),
        io51ndx=>(pin_id=>"64                                   ",
            signal_name=>"io51    (b)                           "),
        io52ndx=>(pin_id=>"65                                   ",
            signal_name=>"io52    (a)                           "),
        in7ndx=>(pin_id=>"66                                    ",
            signal_name=>"in7    (cnt_1)                        "),
        gnd4ndx=>(pin_id=>"67                                   ",
            signal_name=>"gnd        (gnd)                      "),
        in8ndx=>(pin_id=>"68                                    ",
            signal_name=>"in8    (clk)                          "));
    BEGIN
    ASSERT VALID_OPERATING_POINT(user_operating_point)
        REPORT "operating outside OF the recommended RANGE OF this design"
        SEVERITY WARNING;
-- component will not work without power
    PROCESS(vcc1,vcc2,vcc3,vcc4,gnd1,gnd2,gnd3,gnd4)
        BEGIN
        IF vcc1/='1' OR gnd1/='0' OR vcc2/='1' OR gnd2/='0' OR
            vcc3/='1' OR gnd3/='0' OR vcc4/='1' OR gnd4/='0' THEN
            io26<='U';io24<='U';io5<='U';io6<='U';io25<='U';io39<='U';
            io41<='U';io42<='U';io14<='U';io22<='U';io23<='U';io35<='U';
            io43<='U';io3<='U';io44<='U';io4<='U';
            END IF;
        END PROCESS;
    s0:i_driver GENERIC MAP(win_d=>wi_a,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io52ndx).elec_spec))
        PORT MAP(a=>io52,y=>a);
    s1:i_driver GENERIC MAP(win_d=>wi_b,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io51ndx).elec_spec))
        PORT MAP(a=>io51,y=>b);
    s2:i_driver GENERIC MAP(win_d=>wi_c,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io50ndx).elec_spec))
        PORT MAP(a=>io50,y=>c);
    s3:i_driver GENERIC MAP(win_d=>wi_d,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io48ndx).elec_spec))
        PORT MAP(a=>io48,y=>d);
    s4:i_driver GENERIC MAP(win_d=>wi_e,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io45ndx).elec_spec))
        PORT MAP(a=>io45,y=>e);
    s5:i_driver GENERIC MAP(win_d=>wi_f,
            pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io33ndx).elec_spec))
        PORT MAP(a=>io33,y=>     F);
```

Figure 6.3.3-1 continued. An EDS Component Model.

```
s6:i_driver GENERIC MAP(win_d=>wi_g,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io32ndx).elec_spec))
    PORT MAP(a=>io32,y=>g);
s7:i_driver GENERIC MAP(win_d=>wi_d32,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io47ndx).elec_spec))
    PORT MAP(a=>io47,y=>d32);
s8:i_driver GENERIC MAP(win_d=>wi_d33,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io46ndx).elec_spec))
    PORT MAP(a=>io46,y=>d33);
s9:i_driver GENERIC MAP(win_d=>wi_cnt_1,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(in7ndx).elec_spec))
    PORT MAP(a=>in7,y=>cnt_1);
s10:i_driver GENERIC MAP(win_d=>wi_cnt_2,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io49ndx).elec_spec))
    PORT MAP(a=>io49,y=>cnt_2);
s11:i_driver GENERIC MAP(win_d=>wi_clk,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(in8ndx).elec_spec))
    PORT MAP(a=>in8,y=>clk);
s12:i_driver GENERIC MAP(win_d=>wi_ldl,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(in5ndx).elec_spec))
    PORT MAP(a=>in5,y=>ldl);
s13:i_driver GENERIC MAP(win_d=>wi_p_u,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(in1ndx).elec_spec))
    PORT MAP(a=>in1,y=>p_u);
s14:i_driver GENERIC MAP(win_d=>wi_dta_clk_s,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(in6ndx).elec_spec))
    PORT MAP(a=>in6,y=>dta_clk_s);
s15:i_driver GENERIC MAP(win_d=>wi_not_ovd_s,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(in2ndx).elec_spec))
    PORT MAP(a=>in2,y=>not_ovd_s);
s16:i_driver GENERIC MAP(win_d=>wi_not_mclr_s,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(in3ndx).elec_spec))
    PORT MAP(a=>in3,y=>not_mclr_s);
s17:i_driver GENERIC MAP(win_d=>wi_not_eoc,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(in4ndx).elec_spec))
    PORT MAP(a=>in4,y=>not_eoc);
s18:i_driver GENERIC MAP(win_d=>wi_ena_0,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io34ndx).elec_spec))
    PORT MAP(a=>io34,y=>ena_0);
s19:i_driver GENERIC MAP(win_d=>wi_not_stb_s,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io31ndx).elec_spec))
    PORT MAP(a=>io31,y=>not_stb_s);
s20:i_driver GENERIC MAP(win_d=>wi_sdi,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io30ndx).elec_spec))
    PORT MAP(a=>io30,y=>sdi);
s21:i_driver GENERIC MAP(win_d=>wi_sow_s,
        pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io29ndx).elec_spec))
    PORT MAP(a=>io29,y=>sow_s);
s22:sync_checker GENERIC MAP(x_gen=>x_generation,m_gen=>m_generation,
```

Figure 6.3.3-1 continued. An EDS Component Model.

```
          syncconstraint=>    SYNCS(EDS.edsnfo(in2ndx).sync_spec(1)))
      PORT MAP(clk=>not_stb_s,data_in=>not_ovd_s,data_out=>inot_ovd_s);
s23:async_checker GENERIC MAP(x_gen=>x_generation,m_gen=>m_generation,
          asyncconstraint=>    ASYNCS(EDS.edsnfo(in6ndx).async_spec(1)))
      PORT MAP(data_in=>dta_clk_s,data_out=>idta_clk_s);
s24:async_checker GENERIC MAP(x_gen=>x_generation,m_gen=>m_generation,
          asyncconstraint=>    ASYNCS(EDS.edsnfo(in3ndx).async_spec(1)))
      PORT MAP(data_in=>not_mclr_s,data_out=>inot_mclr_s);
s25:async_checker GENERIC MAP(x_gen=>x_generation,m_gen=>m_generation,
          asyncconstraint=>    ASYNCS(EDS.edsnfo(io31ndx).async_spec(1)))
      PORT MAP(data_in=>not_stb_s,data_out=>inot_stb_s);
s26:o_driver GENERIC MAP(delay_y=>    DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
          wir_o=>wi_not_lrcs,m_gen=>m_generation,x_gen=>x_generation,
          pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io3ndx).elec_spec))
      PORT MAP(a=>not_lrc_s,y=>io3);
s27:o_driver GENERIC MAP(delay_y=>    DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
          wir_o=>wi_not_eoc1,m_gen=>m_generation,x_gen=>x_generation,
          pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io4ndx).elec_spec))
      PORT MAP(a=>not_eoc1,y=>io4);
s28:o_driver GENERIC MAP(delay_y=>    DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
          wir_o=>wi_dta_1s,m_gen=>m_generation,x_gen=>x_generation,
          pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io5ndx).elec_spec))
      PORT MAP(a=>dta_1_s,y=>io5);
s29:o_driver GENERIC MAP(delay_y=>    DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
          wir_o=>wi_cnt_0,m_gen=>m_generation,x_gen=>x_generation,
          pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io6ndx).elec_spec))
      PORT MAP(a=>cnt_0,y=>io6);
s30:o_driver GENERIC MAP(delay_y=>    DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
          wir_o=>wi_rdy_s,m_gen=>m_generation,x_gen=>x_generation,
          pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io14ndx).elec_spec))
      PORT MAP(a=>rdy_s,y=>io14);
s31:o_driver GENERIC MAP(delay_y=>    DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
          wir_o=>wi_not_dta1rz,m_gen=>m_generation,x_gen=>x_generation,
          pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io22ndx).elec_spec))
      PORT MAP(a=>not_dta_1_rz,y=>io22);
s32:o_driver GENERIC MAP(delay_y=>    DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
          wir_o=>wi_not_dta0rz,m_gen=>m_generation,x_gen=>x_generation,
          pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io23ndx).elec_spec))
      PORT MAP(a=>not_dta_0_rz,y=>io23);
s33:o_driver GENERIC MAP(delay_y=>    DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
          wir_o=>wi_not_act,m_gen=>m_generation,x_gen=>x_generation,
          pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io24ndx).elec_spec))
      PORT MAP(a=>not_act,y=>io24);
s34:o_driver GENERIC MAP(delay_y=>    DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
          wir_o=>wi_dta_0s,m_gen=>m_generation,x_gen=>x_generation,
          pin_data=>    ELECTRICAL_PIN_SPEC(EDS.edsnfo(io25ndx).elec_spec))
      PORT MAP(a=>dta_0_s,y=>io25);
s35:o_driver GENERIC MAP(delay_y=>    DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
          wir_o=>wi_act,m_gen=>m_generation,x_gen=>x_generation,
```

Figure 6.3.3-1 continued. An EDS Component Model.

```
            pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io26ndx).elec_spec))
        PORT MAP(a=>act,y=>io26);
    s36:o_driver GENERIC MAP(delay_y=>  DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
            wir_o=>wi_stb_cs,m_gen=>m_generation,x_gen=>x_generation,
            pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io35ndx).elec_spec))
        PORT MAP(a=>stb_cs,y=>io35);
    s37:o_driver GENERIC MAP(delay_y=>  DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
            wir_o=>wi_ldl1,m_gen=>m_generation,x_gen=>x_generation,
            pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io39ndx).elec_spec))
        PORT MAP(a=>ldl1,y=>io39);
    s38:o_driver GENERIC MAP(delay_y=>  DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
            wir_o=>wi_urc_s,m_gen=>m_generation,x_gen=>x_generation,
            pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io41ndx).elec_spec))
        PORT MAP(a=>urc_s,y=>io41);
    s39:o_driver GENERIC MAP(delay_y=>  DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
            wir_o=>wi_srm,m_gen=>m_generation,x_gen=>x_generation,
            pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io42ndx).elec_spec))
        PORT MAP(a=>srm,y=>io42);
    s40:o_driver GENERIC MAP(delay_y=>  DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
            wir_o=>wi_ovr_0,m_gen=>m_generation,x_gen=>x_generation,
            pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io43ndx).elec_spec))
        PORT MAP(a=>ovr_0,y=>io43);
    s41:o_driver GENERIC MAP(delay_y=>  DELAYS(EDS.edsnfo(io3ndx).delay_spec(1)),
            wir_o=>wi_clk1,m_gen=>m_generation,x_gen=>x_generation,
            pin_data=>  ELECTRICAL_PIN_SPEC(EDS.edsnfo(io44ndx).elec_spec))
        PORT MAP(a=>clk1,y=>io44);
    s42:dqd_core  PORT MAP(a,b,c,d,e, F,g,d32,d33,sow_s,p_u,ldl,inot_ovd_s,
            idta_clk_s,clk,ena_0,not_eoc,inot_mclr_s,cnt_1,cnt_2,sdi,inot_stb_s,
            act,not_act,cnt_0,dta_0_s,dta_1_s,srm,urc_s,not_lrc_s,clk1,
            ldl1,rdy_s,not_dta_0_rz,not_dta_1_rz,stb_cs,ovr_0,not_eoc1);
END  arch_dqd_comp_str;
```

Figure 6.3.3-1 continued. An EDS Component Model.

6.4 The Circuit Card Assembly

Modeling of an assembly is accomplished via structural VHDL. Any type of component may be included in an assembly model. In general, it is not necessary to develop a complete Electronic Data Sheet model for primitive components. Simple models which describe the component and how it interfaces to the system are considered to be adequate representations of these devices.

6.4.1 The CCA Model

The last step is to generate the circuit card assembly (or CCA) model for the module. The CCA model should be kept separate from the EDS model for the assembly. In the CCA model, only those components which represent actual hardware elements are called out. A printed wiring board layout engineer should be able to look at this model and extract the CCA schematic diagram. A sample CCA model is depicted in Figure 6.4.1-1.

6.4.2 Interface Connections

The INTERFACE_CONNECTIONS constant is included in this model. This is the appropriate place for this information as it pertains directly to the architecture of the CCA.

```
--*****************************************************************
--   (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_cca.vhd
--        Module  Name(s):      dqd_ccaentity/arch_dqd_cca_strarchitecture
--        Constraints:          none
--        Limitations:          none
--        I/O  Format(s):       std_logic
--        Purpose and Use:      This VHDL module contains the structural
--             description of the DQD circuit card assembly.
--        Notes:                If a fixed voltage_vector is employed
--             throughout the design, the operating point declaration
--             "POINT" can be declared for use by all elements of the
--             design hence eliminating the need for carrying separate
--             "POINT" declarations for each component/assembly.
--*****************************************************************
--   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
--        dqdcds.cmp_point      operating point declaration for component
--   Component  Models
--        dqd_core              core component model
--        capacitor             capacitor component
--        resistor_network      resistor network component
--        connector             connector component
--   Platform:                  486/33MHz PC
--   Software/Version:          V-System for Windows, Version 3.0
--*****************************************************************
```

Figure 6.4.1-1. A DQD CCA Model.

```
--    Design Specification Elements
--          entire file
--**************************************************************
LIBRARY ieee;
LIBRARY eia;
USE ieee.STD_LOGIC_1164.ALL;
USE eia.EIA_567.ALL;
USE   work.dqdcds.cmp_point;
USE  work.dqd_ds.ALL;
ENTITY dqd_cca IS
      GENERIC(x_generation: BOOLEAN:= TRUE;
            m_generation: SEVERITY_LEVEL:= WARNING;
            user_operating_point:  POINT:=(selection=>      TNOM,temp=>27      degrees_c,
                  SUPPLY=>(0=>5.0        v)));
      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:='U';
            ldl1,rdy_s,not_dta_0_rz,not_dta_1_rz,stb_cs,ovr_0,not_eoc1:  INOUT STD_LOGIC:='U';
            vcc,gnd:IN STD_LOGIC);
      END  dqd_cca;
ARCHITECTURE arch_dqd_cca_str  OF dqd_cca  IS
      COMPONENT  dqd_comp
            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,
                  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;
                  user_operating_point:cmp_point:=(selection=>tzero,temp=>27              degrees_c,
                        SUPPLY=>(0=>5.0        v));
                  x_generation: BOOLEAN:= FALSE;
                  m_generation: SEVERITY_LEVEL:= FAILURE;
                  part_num: EIA_STRING);
            PORT(in1,in2,in3,in4,in5,in6,in7,in8:IN STD_LOGIC;
                  io1,io2,io3,io4,io5,io6,io7,io8,io9,io10,io11,io12,io13:INOUT STD_LOGIC;
                  io14,io15,io16,io17,io18,io19,io20,io21,io22,io23:INOUT STD_LOGIC;
                  io24,io25,io26,io27,io28,io29,io30,io31,io32,io33:INOUT STD_LOGIC;
                  io34,io35,io36,io37,io38,io39,io40,io41,io42,io43:INOUT STD_LOGIC;
                  io44,io45,io46,io47,io48,io49,io50,io51,io52:INOUT STD_LOGIC;
                  vcc1,vcc2,vcc3,vcc4,gnd1,gnd2,gnd3,gnd4:IN STD_LOGIC);
                  END COMPONENT;
      COMPONENT  capacitor
            GENERIC(part_num: EIA_STRING;
                  value:CAPACITANCE;
                  rating: VOLTAGE);
            PORT(p1,p2: IN STD_LOGIC);
            END COMPONENT;
      COMPONENT  resistor_network
            GENERIC(part_num: EIA_STRING;
```

Figure 6.4.1-1 continued. A DQD CCA Model.

```
            value: RESISTANCE;
            rating: POWER);
        PORT(com: IN STD_LOGIC;
            p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13: OUT STD_LOGIC);
        END COMPONENT;
    COMPONENT connector
        GENERIC(part_num: EIA_STRING;
            c_del: TIME:=0 ns);
        PORT(a2,a3,a4,a5,a6,a11,a16,a17,a20,a21,a23,a26,a28,a31,a35,a37,a38,
            b1,b7,b8,b9,b10,b12,b13,b14,b15,b18,b19,b22,b24,b25,b27,b29,b30,
            b32,b33,b34,b36,b39,b40: IN STD_LOGIC;
            a1,a7,a8,a9,a10,a12,a13,a14,a15,a18,a19,a22,a24,a25,a27,a29,a30,
            a32,a33,a34,a36,a39,a40,b2,b3,b4,b5,b6,b11,b16,b17,b20,b21,b23,
            b26,b28,b31,b35,b37,b38: OUT STD_LOGIC);
        END COMPONENT;
    FOR ALL: dqd_comp USE ENTITY work.dqd_comp;
    FOR ALL: capacitor USE ENTITY work.capacitor;
    FOR ALL: resistor_network USE ENTITY work.resistor_network;
    FOR ALL: connector USE ENTITY work.connector;
    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 elements of 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                                     ",
```

Figure 6.4.1-1 continued. A DQD CCA Model.

```
        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                                  "),
```

Figure 6.4.1-1 continued. A DQD CCA Model.

```
    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                                     "));
SIGNAL ivcc,iact,ildl1,ip_u,icnt_0,irdy_s,ildl,ib,inot_stb_s,ignd,
    idta_0_s,id,ic,isow_s,inot_ovd_s,isrm,inot_dta_1_rz,idta_clk_s,id32,
    iurc_s,istb_cs,ig,inot_dta_0_rz,iclk,iena_0,inot_lrc_s,inot_eoc,iovr_0,
    inot_mclr_s,icnt_2,idta_1_s,ie,icnt_1,ia,inot_eoc1,iif,inot_act,
    iclk1,id33,isdi:STD_LOGIC;
BEGIN
c1:capacitor GENERIC MAP("m39014/02-1230                   ",0.1 uf, 100 v)
    PORT MAP (ivcc,ignd);
r1:resistor_network GENERIC MAP("m8340101k2402jb              ",
        24 kilohms, 0.1 W)
    PORT MAP (ivcc,ia,ib,ic,id,ie,iif,ig,icnt_1,icnt_2,inot_ovd_s,inot_mclr_s,
        idta_clk_s,inot_eoc);
r2:resistor_network GENERIC MAP("m8340101k2402jb              ",
        24 kilohms, 0.1 W)
    PORT MAP (ivcc,iena_0,iclk,ildl,inot_stb_s,isow_s,isdi,id32,id33,ip_u,
        OPEN,OPEN,OPEN,OPEN);
p1:connector GENERIC MAP("m28754/8-01                      ",0 ns)
    PORT MAP(iact,ildl1,p_u,icnt_0,irdy_s,idta_0_s,isrm,inot_dta_1_rz,iurc_s,
        istb_cs,inot_dta_0_rz,inot_lrc_s,iovr_0,idta_1_s,inot_eoc1,inot_act,iclk1,
        vcc,ldl,b,not_stb_s,gnd,d,c,sow_s,not_ovd_s,dta_clk_s,d32,g,clk,ena_0,
        not_eoc,not_mclr_s,cnt_2,e,cnt_1,a, F,d33,sdi,ivcc,ildl,ib,inot_stb_s,ignd,
        id,ic,isow_s,inot_ovd_s,idta_clk_s,id32,ig,iclk,iena_0,inot_eoc,
        inot_mclr_s,icnt_2,ie,icnt_1,ia,iif,id33,isdi,act,ldl1,ip_u,cnt_0,rdy_s,
        dta_0_s,srm,not_dta_1_rz,urc_s,stb_cs,not_dta_0_rz,not_lrc_s,ovr_0,
        dta_1_s,not_eoc1,not_act,clk1);
u1:dqd_comp GENERIC MAP    (part_num=>"epm5128jmb          ")
    PORT MAP(ip_u,inot_ovd_s,inot_mclr_s,inot_eoc,ildl,idta_clk_s,icnt_1,iclk,
        OPEN,OPEN,inot_lrc_s,inot_eoc1,idta_1_s,icnt_0, OPEN,OPEN,OPEN,OPEN,OPEN
        OPEN,OPEN,OPEN,irdy_s,OPEN,OPEN,OPEN,OPEN,OPEN,OPEN,inot_dta_1_rz,
        inot_dta_0_rz,inot_act,idta_0_s,iact, OPEN,OPEN,isow_s,isdi,inot_stb_s,ig,
        iif,iena_0,istb_cs,OPEN,OPEN,OPEN,ildl1,OPEN,iurc_s,isrm,iovr_0,iclk1,ie,
        id33,id32,id,icnt_2,ic,ib,ia,ivcc,ivcc,ivcc,ivcc,ignd,ignd,ignd,ignd);
END  arch_dqd_cca_str;
```

Figure 6.4.1-1 continued. A DQD CCA Model.

This page intentionally left blank