ispLSI 8000V Family VHDL Code Examples

Introduction

Lattice Semiconductor has introduced a high density CPLD family that offers significant performance capabilities over FPGA solutions. The architecture of the ispLSI8000V family has been specifically designed to support the HDL synthesis flow. This application note includes coding examples designed to allow the user to take advantage of the hardware capabilities of this powerful ispLSI8000V architecture.

8000V Architecture Features

The ispLSI8000V family has a number of performance enhancing architectural features. These include an internal tristate bus, I/O registers, and product term sharing array (PTSA), and product term (PT) control functions. The ispLSI8000V registers have added clock enable control as well as a preset capability. In the macrocell, there are global reset/preset controls to the register as well as a PT reset/preset control. In addition there are global clock pins, shared PT clocks for the Generic Logic Blocks (GLBs) or individual macrocell PT clocks. The latter provides the potential to have 1080 different clocks for the macrocell registers in the ispLSI8000V.

Example VHDL Code Overview

There are five sample files presented in this application note. Each one is designed to highlight certain architectural features, illustrate how to code for that feature, and finally show the results in report files with a minimal amount of extraneous information. The capabilities of all these files can be combined to implement the users desired CPLD design.

The tri_bus.vhd file focuses on the 108 bit wide internal tristate bus. The register.vhd file demonstrates to the user the enhanced register control capabilities available in ispLSI8000V devices. The macros.vhd file shows how to instantiate Lattice hard macros that have been optimized for minimum levels of logic and macrocell utilization. The io_reg.vhd file highlights the savings which can be achieved through the use if I/O cell registers as well and the Product Term Sharing Array (PTSA). Finally, the controls.vhd file illustrates the PT control capabilities of the ispLSI8000V macrocell and the control mux architecture in the ispLSI8000V I/O cells.

Tristate Bus

The ispLSI8000V is the first CPLD to offer an internal tristate bus. The tristate bus is intended to facilitate the design of peripheral functions for a processor by interfacing directly to the processor's bidirectional data bus. The architecture of the internal tristate bus is demonstrated in Figure 1. Each GLB has twenty macrocells. Eighteen of the macrocells drive directly onto the tristate bus with the remaining two macrocells serving as output enable controls.

The tri_bus.vhd file has four sixteen bit registers (rega, regb, regc, and regd) that are individually selected by two address lines to either be written to or read from. The register outputs are all connected to a common internal tristate bus that is then connected to the bidirectional data bus pins. An active RD signal and the appropriate address value enable the output enables of the individual registers. The output enable of the data bus pins is simply enabled by RD pin being active.

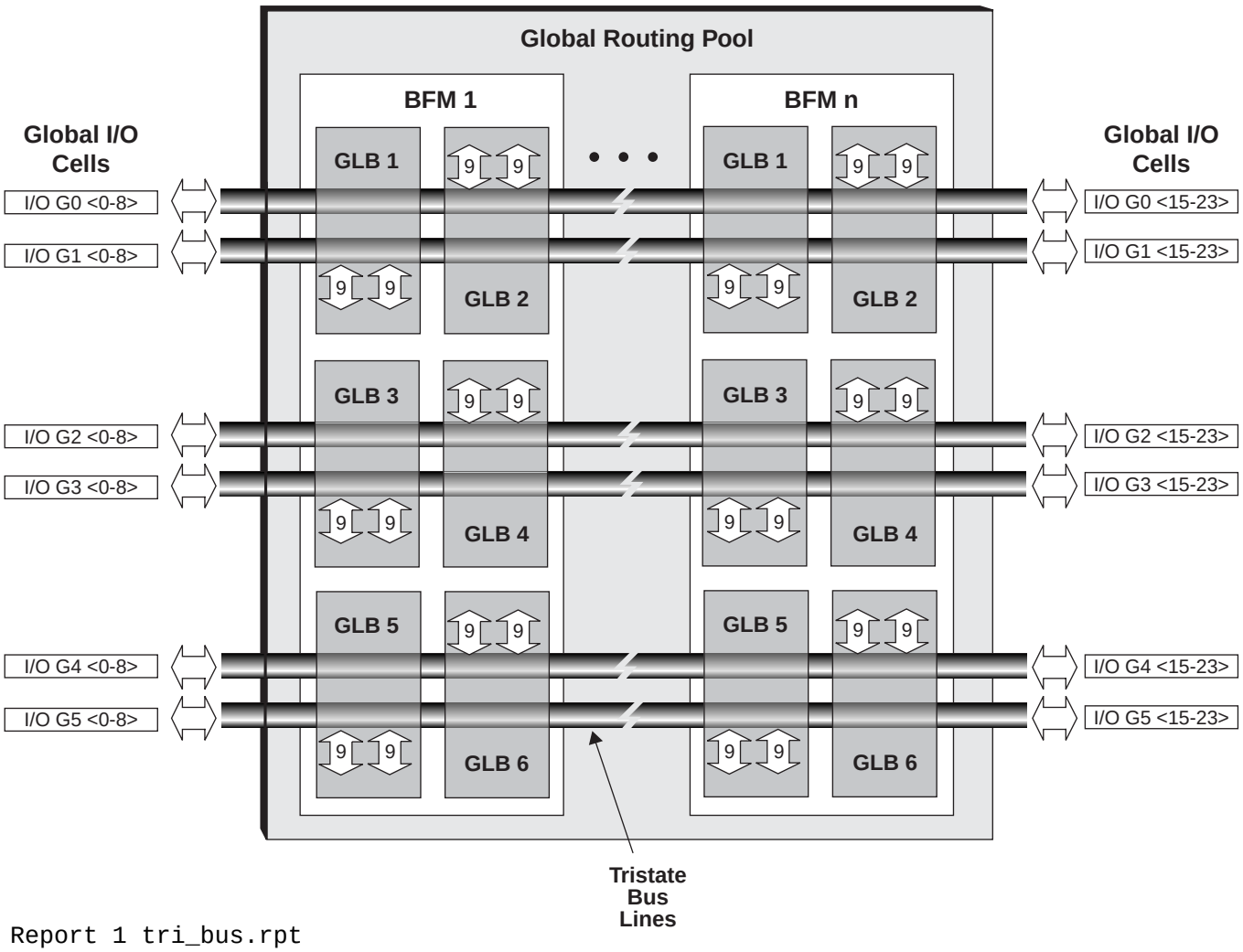
The resulting Report File sections highlight the tristate bus implementation. The excerpts show the internal tristate bus implementation of bit 15 (TRI_BUS_15), the tristate bus output enable control for rega (N_15), bit 15 of rega (REGAZ0Z_15), the output enable control for the data bus (_BUF_1326), and the bidirectional data bus pin (D(15)).

Conclusion:

The Lattice 8000V architecture offers many rich features and controls to support the HDL synthesis process.

Enhanced register control capabilities, optimization of hard macros, the power and felxibility available through the use of I/O cells and the PTSA, and the control capabilities PT's and I/Os have all been demonstrated. Users are able to choose which of these features is necessary to complete their specific design.

Table 1. Internal Tristate Bus




```
Report 1 tri_bus.rpt
Post-Route Design Implementation
_____________________

..
Number of Internal Tristate Nets:16
.
.Internal Tristate Net 7: TRI_BUS_15

   4 Tristate Driver(s)
       (glb02.O7, REGDZ0Z_15, N_14)
       (glb04.O7, REGCZ0Z_15, N_17)
       (glb01.O7, REGAZ0Z_15, N_15)
       (glb03.O7, REGBZ0Z_15, N_16)
   1 Fanout(s)
       (D(15).IR, TRI_BUS_15)
.
.
.
Internal Tristate Output N_15.

        3 Input(s)
            A(0)X, RDX, A(1)X
```

```
        2 Fanout(s)
            glb01.OE0, glb01.OE1
        1 Product Term(s)
        1 GLB Level(s)
        N_15 = !A(0)X & !A(1)X & !RDX
.......

    Internal Tristate Output REGAZ0Z_15 with Global Reset used.

        1 Input(s)
            D_15_Z0
        1 Fanout(s)
            TRI_BUS_15.A5
        1 Product Term(s)
        1 GLB Level(s)

        REGAZ0Z_15.D = D_15_Z0
        REGAZ0Z_15.C = WRX
        REGAZ0Z_15.CE = !A(0)X & !A(1)X
        REGAZ0Z_15.R = GND
......

    Internal Tristate Enable

        N_15 glb01.OE0

        N_15 glb01.OE1
........
    Output Control _BUF_1326

        1 Input(s)
            RDX
        16 Fanout(s)
            D(0).CBUS0, D(14).CBUS0, D(13).CBUS0, D(4).CBUS0,
            D(11).CBUS0, D(7).CBUS0, D(3).CBUS0, D(10).CBUS0, D(6).CBUS0,
            D(12).CBUS0, D(5).CBUS0, D(1).CBUS0, D(8).CBUS0, D(9).CBUS0,
            D(15).CBUS0, D(2).CBUS0
        1 Product Term(s)
        -1 GLB Level(s)

        _BUF_1326 = RDX
........

Bidirectional D(15) IOG0<17>

    Input (TRI_BUS_15.Z0, TRI_BUS_15)
    Output D_15_Z0
        4 Fanout(s)
            glb02.I12, glb04.I12, glb01.I12, glb03.I12
    Enable (glb04.CNTL, _BUF_1326)

    D(15) = TRI_BUS_15
    !D(15).E = !_BUF_1326
```

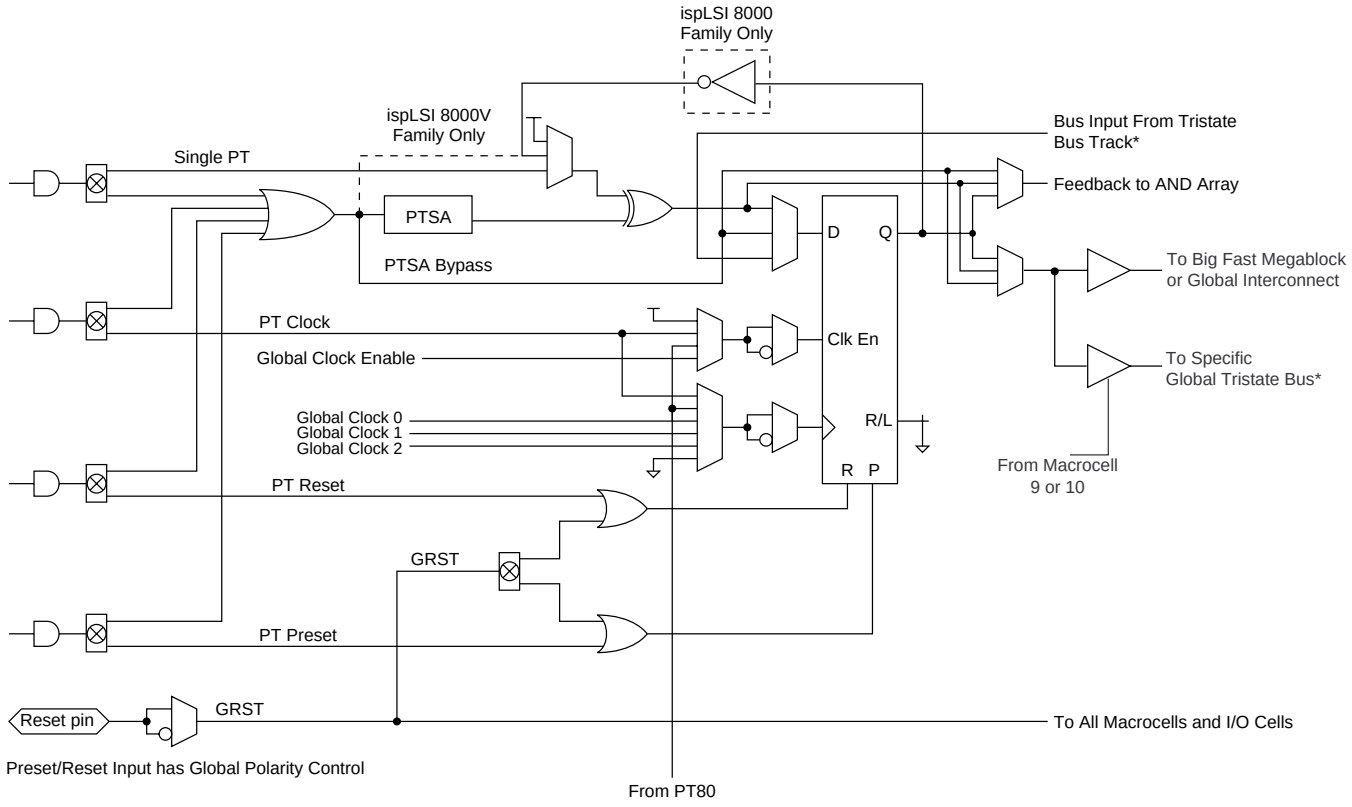
Register Features

The ispLIS8000V macrocell registers offer a rich set of controls, which include preset and reset controls. This permits pre-loading counters to non-zero values and initializing state machines to predefined states. Additionally, there is now a clock enable for gated clocks, as well as clock and clock enable inversion at the macrocell level. Figure 2 shows the macrocell of the ispLSI8000V.

The register.vhd file has four four bit registers (rega, regb, regc, and regd) to demonstrate the macrocell register controls. rega shows a register with preset capability. regb shows a clocked register with clock enable control. regc is another clock enable co ntrolled register with a write signal for the clock. regd is a register using the inverted edge of the clock.

The resulting Report File section demonstrates the register implementations. The rega (REGAZ0Z_0) portion shows the preset (REGAZ0Z_0.PR = !PRESETX). The regb (REGBZ0Z_0) section indicates the use of the clock enable control (REGBZ0Z_0.CE = A(0)X & !A(1)X & !WRX). The regd (REGDZ0Z_0) segment shows the clock inversion at the macrocell (!REGDZ0Z_0.C = CLKX). Finally, the regc section (REGCZ0Z_0) highlights another clock enable controlled register using a write signal for the clock (REGCZ0Z_0.C = WRX).

Figure 2 ispLSI 8000V Macrocell



*Not available for Macrocells 9 and 10.

⊗ : Function Selector (E^2 Cell Controlled)

```
Report 2 register.rpt
    Local Feedback REGAZ0Z_0 with Global Reset used.

        1 Input(s)
            D_0_Z0
        1 Fanout(s)
            glb00.I9
        1 Product Term(s)
        1 GLB Level(s)

        REGAZ0Z_0.D = D_0_Z0
        REGAZ0Z_0.C = WRX
        REGAZ0Z_0.CE = !A(0)X & !A(1)X
        REGAZ0Z_0.R = GND
        REGAZ0Z_0.PR = !PRESETX

    Local Feedback REGBZ0Z_0 with Global Reset used.

        1 Input(s)
            D_0_Z0
        1 Fanout(s)
            glb00.I33
        1 Product Term(s)
        1 GLB Level(s)

        REGBZ0Z_0.D = D_0_Z0
        REGBZ0Z_0.C = CLKX
        REGBZ0Z_0.CE = A(0)X & !A(1)X & !WRX
        REGBZ0Z_0.R = GND

    Local Feedback REGDZ0Z_0 with Global Reset used.

        1 Input(s)
            D_0_Z0
        1 Fanout(s)
            glb00.I4
        1 Product Term(s)
        1 GLB Level(s)

        REGDZ0Z_0.D = D_0_Z0
        !REGDZ0Z_0.C = CLKX
        REGDZ0Z_0.CE = A(0)X & A(1)X & !WRX
        REGDZ0Z_0.R = GND

    Local Feedback REGCZ0Z_0 with Global Reset used.

        1 Input(s)
            D_0_Z0
        1 Fanout(s)
            glb00.I40
        1 Product Term(s)
        1 GLB Level(s)

        REGCZ0Z_0.D = D_0_Z0
        REGCZ0Z_0.C = WRX
        REGCZ0Z_0.CE = !A(0)X & A(1)X
        REGCZ0Z_0.R = GND
.
.
.
```

Hard Macro Instantiation

Lattice offers hard macros with all versions of its developement tools. The hard macros offer the capability of instantiating optimized logic elements into the front end tools. These logic elements have been optimized for performance. The number of logic levels has been reduced, thereby increasing speed. Area has been optimized by minimizing the number of macrocells utilized to implement the logic function.

The macros.vhd file shows how to instantiate Lattice hard macros. They indicate the syntax of the macro components and the port mapping. Two binary up counters with asynchronous reset are instantiated in the file to form a twenty bit counter: CBUA16 and CBUA4. The macros are documented in the “Macro Library Reference Manual” and “5K/8K Macro Library Supplement” manuals. Two lines that need to appear in the library section to enable access to the hard macro libraries are:

```
Library lattice;
use lattice.components.all;
```

The resulting Report File sections demonstrate the macro instantiations. The post route summary section indicates that the 20 bit counter has fit into one GLB. The GLB statistics section indicates that all bits of the counter are implemented in one level of logic for maximum speed performance.

```
Report 3 macros.rpt

Post-Route Design Implementation
________________

Number of Macrocells:        20
Number of GLBs:                 1
.
.
.
GLB and GLB Output Statistics

    GLB Name, Location      GLB Statistics          GLB Output Statistics
    GLB Output Name         Ins, Outs, PTs,TurboPTs Ins, FOs, PTs, Levels, Turbo

        glb00,                  22, 20, 60,  0
            U1_QI0_part0                                 2,  1,  2,  1, OFF
            U1_QI10_part0                               12,  1,  2,  1, OFF
            U1_QI11_part0                               13,  1,  2,  1, OFF
            U1_QI12_part0                               14,  1,  2,  1, OFF
            U1_QI13_part0                               15,  1,  2,  1, OFF
            U1_QI14_part0                               16,  1,  2,  1, OFF
            U1_QI15_part0                               17,  1,  2,  1, OFF
            U1_QI1_part0                                 3,  1,  2,  1, OFF
            U1_QI2_part0                                 4,  1,  2,  1, OFF
            U1_QI3_part0                                 5,  1,  2,  1, OFF
            U1_QI4_part0                                 6,  1,  2,  1, OFF
            U1_QI5_part0                                 7,  1,  2,  1, OFF
            U1_QI6_part0                                 8,  1,  2,  1, OFF
            U1_QI7_part0                                 9,  1,  2,  1, OFF
            U1_QI8_part0                                10,  1,  2,  1, OFF
            U1_QI9_part0                                11,  1,  2,  1, OFF
            U2_QI0_part0                                18,  1,  2,  1, OFF
            U2_QI1_part0                                19,  1,  2,  1, OFF
            U2_QI2_part0                                20,  1,  2,  1, OFF
            U2_QI3_part0                                21,  1,  2,  1, OFF
.
```

I/O Registers and PTSA

The ispLSI8000V devices offers I/O cells with registers that are available either on the input path or the output path. Figure 3 shows the structure of the I/O cell. By implementing an input register in the I/O cell, users can obtain better set up and hold timing for their design. Similarly, by putting an output register in the I/O cell, the designer can obtain a much faster clock to out timing.

ispLSI8000V devices also provide a wider PTSA capability then its competitors. Each macrocell has four PTs dedicated to it. With the PTSA, up to seven groupings of these four PTs can be wire ORed together into one macrocell. This sharing is circular so that macrocells on the upper end of the PTSA borrow from macrocells on the lower end and vice versa. Refer to Figure 4 for the structure of the PTSA.

The io_reg.vhd file has a 28 bit bus that is latched (rege). This register is then decoded into 28 different conditions that are then ORed into one equation. The I/O cell registers are obtained by locking the register clock IOCLK to a global I/O clock (GIOCLK1) on the device. The pin can either be locked in the Assign Pin Location window of the ispEXPERT Compiler GUI or by reading in a property file, along with the design EDIF netlist file, into the compiler. The property file would contain the following line to lock the clock pin:

```
PROPERTY io_reg PIN io_clk LOCK AJ18
ENDPROPERTY
```

Design properties are documented in Chapter 2 of the ispEXPERT Compiler User's Manual. Property file syntax is documented in Appendix B.

The resulting Report File sections show the input register implementation and PTSA usage. The excerpts show the IOCLK pin locked to the global I/O clock pin and the resulting macrocell utilization of one instead of 29 if the data bus was registered in the macrocells. The report file also demonstrates bit 0 of the bus (REGEZ0Z_0) implemented in the I/O cell register and the GLB statistics showing the 28 PTs feeding into macrocell (Y_C).

Figure 3 I/O Cell

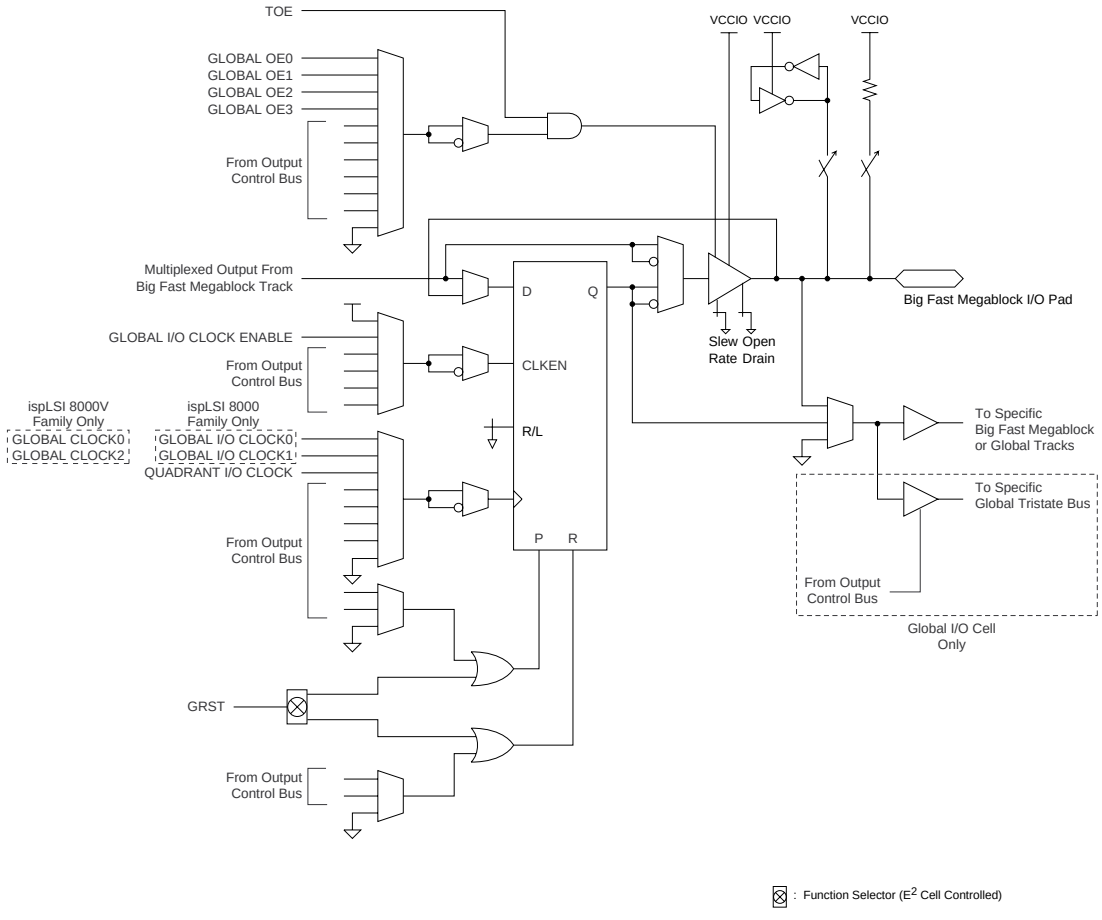
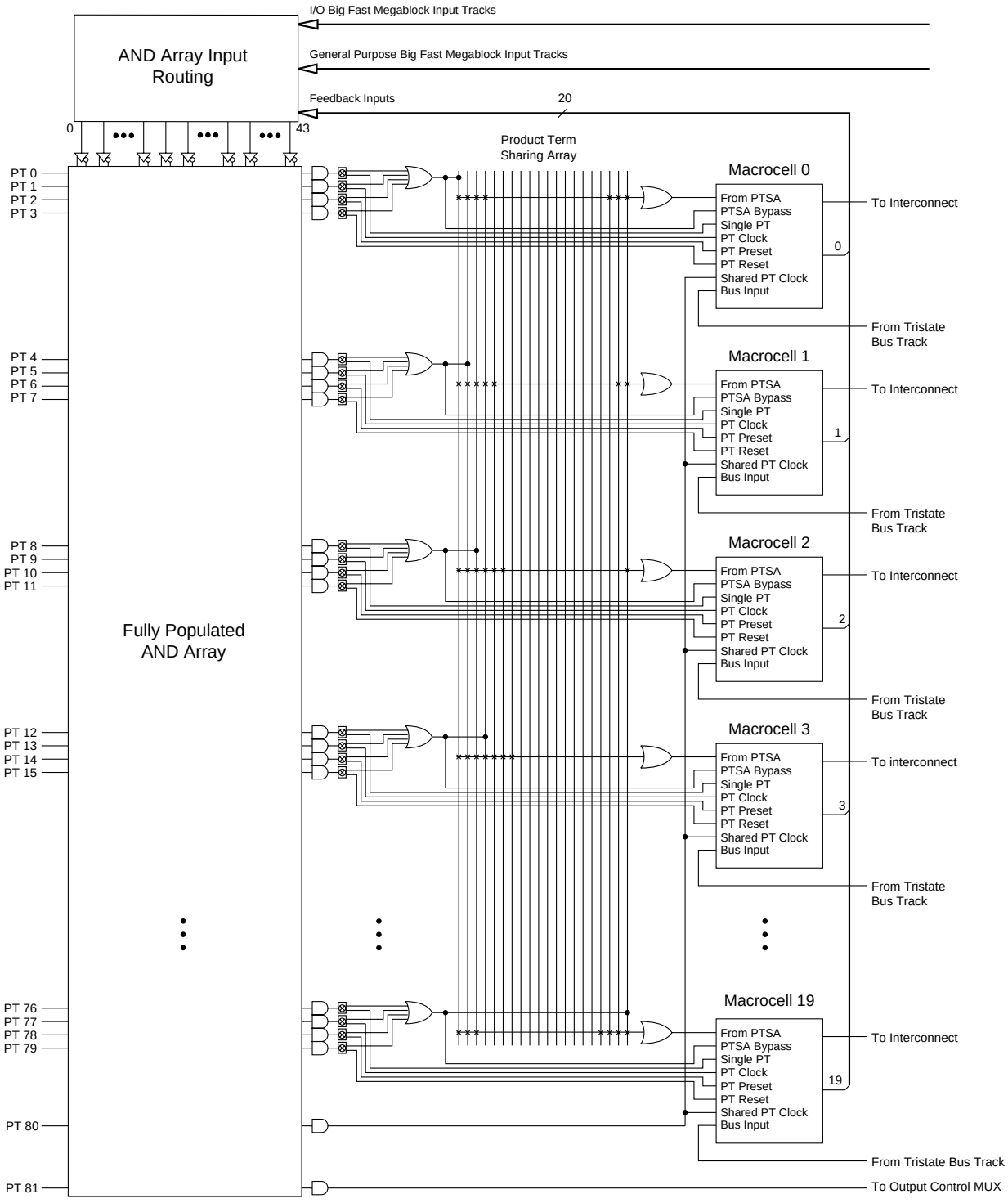


Figure 4 ispLSI 8000V GLB



Note: Macrocells 9 and 10 do not support Tristate Bus Feedback.

▨ Function Selector (E^2 Cell Controlled)

```
Report 4 io_reg.rpt
.
        IO_CLK                  LOCK AJ18, PULLUP
.
Post-Route Design Implementation
————————————————

Number of Macrocells:           1
Number of GLBs:                 1
Number of IOCs:                 29
Number of GLB Levels:           1
.
Registered Input DATA(0) IOS0<0> with Global Reset used.


    Output REGEZ0Z_0
        1 Fanout(s)
            glb00.I40
    Global Clock 1 (IO_CLK.O, IO_CLKX)

    DATA(0).G = IO_CLKX
    DATA(0).R = CLEAR
.
.
.
GLB and GLB Output Statistics

    GLB Name, Location      GLB Statistics          GLB Output Statistics
    GLB Output Name         Ins, Outs, PTs,TurboPTs Ins, FOs, PTs, Levels, Turbo

        glb00,                  28,  1, 28,  0
            Y_C                                         28,  1, 28,  1, OFF
```

Product Term Controls

The ispLSI8000V macrocell and I/O cell registers provide a wide selection of controls. These include global pins, shared PT and dedicated PT controls for reset, preset, clock, clock enable, and output enable. These controls are shown in Figure 2 for the macrocell and Figure 3 for the I/O cell. The I/O cell control multiplexors include inputs that are labeled “From Output Control Bus.” These signals come from PT81 that is generated in each GLB (refer to Figure 4). In the 8840V, there are 42 GLBs that can each generate a control bus signal for PT controls to the I/O cells.

The controls.vhd file has four registers (rega, regb, regc, and regd) to demonstrate the PT controls. rega shows a four bit register using a global clock pin. regb shows another four bit register with a shared PT clock (shared_pt_clk). regc is a single bit register that uses a dedicated PT clock (dedicated_pt_clk). regd is another single bit register that is implemented in an I/O cell and uses PT controls generated by PT81 in the GLBs. The regd is assigned to the I/O cell by reading in a property file along with the design EDIF netlist file into the compiler. The property file should contain the following line to assign the I/O cell:

PROPERTY CONTROLS INST regd_int REGTYPE IOC ENDPROPERTY

The resulting Report File section demonstrates the register control implementations. The regb (REGB_C_0 and REGB_C_2) portion shows the shared PT clock (IN_1X & IN_CLKX). The rega (REGA_C_2) section indicates the use of the global clock (CLKX). Below this section is the generation of one of the PT81 controls for the I/O cell (CONTROL_PT_OE_IZ0). The regc (REGC_C) segment shows the dedicated PT clock (IN_2X & IN_CLKX). Again, below this section is another PT81 control function for the I/O cell register (CONTROL_PT_CLK). Finally, the regd section (REGD) highlights the PT81 I/O cell controls for the three state output REGD.

```
Report 5 controls.rpt
.
    Output REGB_C_0 with Global Reset used.

        1 Input(s)
            D(0)X
        1 Fanout(s)
            REGB(0).IR
        1 Product Term(s)
        1 GLB Level(s)

        REGB_C_0.D = D(0)X
        REGB_C_0.C = IN_1X & IN_CLKX
        REGB_C_0.R = !CLEARX
.
    Output REGB_C_2 with Global Reset used.

        1 Input(s)
            D(2)X
        1 Fanout(s)
            REGB(2).IR
        1 Product Term(s)
        1 GLB Level(s)

        REGB_C_2.D = D(2)X
        REGB_C_2.C = IN_1X & IN_CLKX
        REGB_C_2.R = !CLEARX

    Output REGA_C_2 with Global Reset used.

        1 Input(s)
            D(2)X
        1 Fanout(s)
            REGA(2).IR
        1 Product Term(s)
        1 GLB Level(s)

        REGA_C_2.D = D(2)X
        REGA_C_2.C = CLKX
        REGA_C_2.R = !CLEARX

    Output Control CONTROL_PT_OE_IZ0

        2 Input(s)
            IN_1X, IN_4X
        1 Fanout(s)
            REGD.CBUS7
        1 Product Term(s)
        -1 GLB Level(s)

        CONTROL_PT_OE_IZ0 = IN_1X & IN_4X
.
    Output REGC_C with Global Reset used.
```

```
        1 Input(s)
            D(0)X
        1 Fanout(s)
            REGC.IR
        1 Product Term(s)
        1 GLB Level(s)

        REGC_C.D = D(0)X
        REGC_C.C = IN_2X & IN_CLKX
        REGC_C.R = !CLEARX
.
.
.
    Output Control CONTROL_PT_CLK

        2 Input(s)
            IN_3X, IN_CLKX
        1 Fanout(s)
            REGD.CBUS5
        1 Product Term(s)
        -1 GLB Level(s)

        CONTROL_PT_CLK = IN_3X & IN_CLKX
.
.
.
Three-State Output REGD IOS1<11>

    Input (D(0).BO, D(0)X)
    Clock (glb03.CNTL, CONTROL_PT_CLK)
    Clken (glb04.CNTL, _BUF_670)
    Reset (glb00.CNTL, _BUF_672)
    Preset (glb01.CNTL, _BUF_673)
    Enable (glb02.CNTL, CONTROL_PT_OE_IZ0)

    REGD = D(0)X
    !REGD.E = !CONTROL_PT_OE_IZ0
    REGD.C = CONTROL_PT_CLK
    REGD.CE = _BUF_670
    REGD.R = _BUF_672
    REGD.PR = _BUF_673
.
.
.
```

```
Listing 1 tri_bus.vhd
LIBRARY IEEE,STD;
USE IEEE.std_logic_1164.all ;
USE IEEE.std_logic_unsigned.all ;

— Sample VHDL code to demonstrate 8840 Internal tristate bus


ENTITY training IS
      PORT (
            d           :   INOUT   std_logic_vector(15 downto 0);
            wr          :   IN            std_logic;
            rd          :   IN            std_logic;
            a           :   IN            std_logic_vector(1 downto 0);
            clear :   IN            std_logic

      );

END training ;


ARCHITECTURE behav OF training IS

      signal rega, regb, regc, regd : std_logic_vector(15 downto 0);
      signal tri_bus                              : std_logic_vector(15 downto 0);

begin

reg_a: PROCESS (wr, clear, a)
BEGIN
      IF (clear = '0' ) THEN
            rega <= "0000000000000000";
      ELSIF (wr'event and wr = '1') THEN
            IF (a = "00") THEN
                  rega <= d;
            END IF ;
      END IF ;
END PROCESS reg_a;

reg_b: PROCESS (wr, clear, a)
BEGIN
      IF (clear = '0') THEN
            regb <= "0000000000000000";
      ELSIF (wr'event and wr = '1') THEN
            IF (a = "01") THEN
                  regb <= d;
            END IF ;
      END IF ;
END PROCESS reg_b;

reg_c: PROCESS (wr, clear, a)
BEGIN
      IF (clear = '0') THEN
```

```
            regc <= “0000000000000000”;
      ELSIF (wr’event and wr = ‘1’) THEN
            IF (a = “10”) THEN
                  regc <= d;
            END IF ;
      END IF ;
END PROCESS reg_c;

reg_d: PROCESS (wr, clear, a)
BEGIN
      IF (clear = ‘0’) THEN
            regd <= “0000000000000000”;
      ELSIF (wr’event and wr = ‘1’) THEN
            IF (a = “11”) THEN
                  regd <= d;
            END IF ;
      END IF ;
END PROCESS reg_d;

————————————————————————————————————-
————————————— Tristate Bus Example ——————————————-
————————————————————————————————————-

tri_bus <= rega when (rd = ‘0’ and a = “00”) else “ZZZZZZZZZZZZZZZZ”;
tri_bus <= regb when (rd = ‘0’ and a = “01”) else “ZZZZZZZZZZZZZZZZ”;
tri_bus <= regc when (rd = ‘0’ and a = “10”) else “ZZZZZZZZZZZZZZZZ”;
tri_bus <= regd when (rd = ‘0’ and a = “11”) else “ZZZZZZZZZZZZZZZZ”;

d <= tri_bus when rd = ‘0’ else “ZZZZZZZZZZZZZZZZ”;

END behav ;

Listing 2 register.vhd
LIBRARY IEEE,STD;
USE IEEE.std_logic_1164.all ;
USE IEEE.std_logic_unsigned.all ;

— Sample VHDL code to demonstrate 8840 Register Controls


ENTITY training IS
      PORT (
            d           :   INOUT   std_logic_vector(3 downto 0);
            clk         :   IN            std_logic;
            wr          :   IN            std_logic;
            rd          :   IN            std_logic;
            a           :   IN            std_logic_vector(1 downto 0);
            clear :   IN            std_logic;
            preset      :   IN            std_logic

      );

END training ;
```

```
ARCHITECTURE behav OF training IS

      signal rega, regb, regc, regd : std_logic_vector(3 downto 0);

begin

-------------------------------------------
------------- Register Preset Example -----------------
-------------------------------------------

reg_a: PROCESS (wr, preset, clear, a)
BEGIN
      IF ((clear) = '0' and (preset) = '1') THEN
            rega <= "0000";
      ELSIF ((clear) = '0' and (preset) = '0') THEN
            rega <= "0000";
      ELSIF ((clear) = '1' and (preset) = '0') THEN
            rega <= "1111";
      ELSIF (wr'event and wr = '1' and a = "00") THEN
            rega <= d;
      END IF ;
END PROCESS reg_a;

-------------------------------------------
---------------- Clock Enable Example ----------------
-------------------------------------------

reg_b: PROCESS (clk, wr, clear, a)
BEGIN
      IF (clear = '0') THEN
            regb <= "0000";
      ELSIF (clk'event and clk = '1') THEN
            IF (wr = '0' and a = "01") THEN
                  regb <= d;
            END IF ;
      END IF ;
END PROCESS reg_b;

reg_c: PROCESS (wr, clear, a)
BEGIN
      IF (clear = '0') THEN
            regc <= "0000";
      ELSIF (wr'event and wr = '1' and a = "10") THEN
            regc <= d;
      END IF ;
END PROCESS reg_c;

-------------------------------------------
---------------- Clock Inversion Example ---------------
-------------------------------------------

reg_d: PROCESS (clk, wr, clear, a)
BEGIN
```

```
      IF (clear = '0') THEN
            regd <= "0000";
      ELSIF (clk'event and clk = '0' and wr = '0' and a = "11") THEN
            regd <= d;
      END IF ;
END PROCESS reg_d;

d <= (rega) when (rd = '0' and a = "00") else
     (regb) when (rd = '0' and a = "01") else
     (regc) when (rd = '0' and a = "10") else
     (regd) when (rd = '0' and a = "11") else "ZZZZ" ;

END behav ;

Listing 3 macros.vhd
LIBRARY IEEE,STD;
USE IEEE.std_logic_1164.all ;
USE IEEE.std_logic_unsigned.all ;
Library lattice;                          -- Note:  Library required
use lattice.components.all;               -- Note:  Use lattice macro library

-- Sample VHDL code to demonstrate 8840 hard macro instantialtion

ENTITY macros IS
   PORT (
         clock          :   IN      std_logic;
         clear          :   IN      std_logic;
         set            :   IN      std_logic;
         enable         :   IN      std_logic;
         q              :   OUT     std_logic_vector(19 downto 0)
        ) ;

END macros ;


ARCHITECTURE behav OF macros IS

   COMPONENT CBUA16
      port (
            CAO      :   OUT     std_logic;
            Q0, Q1, Q2, Q3       :   OUT     std_logic;
            Q4, Q5, Q6, Q7       :   OUT     std_logic;
            Q8, Q9, Q10, Q11     :   OUT     std_logic;
            Q12, Q13, Q14, Q15   :   OUT     std_logic;
            CAI      :   IN      std_logic;
            CD       :   IN      std_logic;
            CLK      :   IN      std_logic;
            D0, D1, D2, D3       :   IN      std_logic;
            D4, D5, D6, D7       :   IN      std_logic;
            D8, D9, D10, D11     :   IN      std_logic;
            D12, D13, D14, D15   :   IN      std_logic;
            EN       :   IN      std_logic;
            LD       :   IN      std_logic;
            SD       :   IN      std_logic
```

```
          ) ;
   END component ;

   COMPONENT CBUA4
      port (
            CAO      :   OUT     std_logic;
            Q0, Q1, Q2, Q3       :   OUT     std_logic;
            CAI      :   IN      std_logic;
            CD       :   IN      std_logic;
            CLK      :   IN      std_logic;
            D0, D1, D2, D3       :   IN      std_logic;
            EN       :   IN      std_logic;
            LD       :   IN      std_logic;
            SD       :   IN      std_logic
          ) ;
   END component ;

      signal carry      : std_logic;

BEGIN

U1: CBUA16 PORT MAP ( CAO=>carry,
                      Q0=>q(0), Q1=>q(1), Q2=>q(2), Q3=>q(3),
                      Q4=>q(4), Q5=>q(5), Q6=>q(6), Q7=>q(7),
                      Q8=>q(8), Q9=>q(9), Q10=>q(10), Q11=>q(11),
                      Q12=>q(12), Q13=>q(13), Q14=>q(14), Q15=>q(15),
                                CAI=>'1', CD=>clear, CLK=>clock,
                      D0=>'0', D1=>'0', D2=>'0', D3=>'0',
                      D4=>'0', D5=>'0', D6=>'0', D7=>'0',
                      D8=>'0', D9=>'0', D10=>'0', D11=>'0',
                      D12=>'0', D13=>'0', D14=>'0', D15=>'0',
                                EN=>enable, LD=>'0', SD=>set);

U2: CBUA4 PORT MAP (
                     Q0=>q(16), Q1=>q(17), Q2=>q(18), Q3=>q(19),
                               CAI=>carry, CD=>clear, CLK=>clock,
                     D0=>'0', D1=>'0', D2=>'0', D3=>'0',
                               EN=>enable, SD=>set, LD=>'0');

END behav ;

Listing 4 io_reg.vhd
LIBRARY IEEE,STD;
USE IEEE.std_logic_1164.all ;
use ieee.numeric_std.all;

— Sample VHDL code to demonstrate 8840 I/O Registers and Product Term Sharing Array


ENTITY training IS
      PORT (
            data  :   IN            std_logic_vector(27 downto 0);
            io_clk      :   IN            std_logic;
            clear :   IN            std_logic;
```

```
            y           :   OUT           std_logic

      );

END training ;


ARCHITECTURE behav OF training IS

      signal rege                   : std_logic_vector(27 downto 0);
      signal pt1, pt2, pt3, pt4           : std_logic;
      signal pt5, pt6, pt7, pt8           : std_logic;
      signal pt9, pt10, pt11, pt12  : std_logic;
      signal pt13, pt14, pt15, pt16 : std_logic;
      signal pt17, pt18, pt19, pt20 : std_logic;
      signal pt21, pt22, pt23, pt24 : std_logic;
      signal pt25, pt26, pt27, pt28 : std_logic;

begin


-------------------------------------------
------------- I/O Cell Register Example -------------
-------------------------------------------

reg_e: PROCESS ( io_clk, clear)
BEGIN
      IF (clear = '0') THEN
            rege <= x"0000000";
      ELSIF (io_clk'event and io_clk = '1') THEN
            rege <= data;
      END IF ;

      IF (rege = x"0000001") THEN   --     PT1
            pt1 <= '1';
      ELSE
            pt1 <= '0';
      END IF ;

      IF (rege = x"0000002") THEN   --     PT2
            pt2 <= '1';
      ELSE
            pt2 <= '0';
      END IF ;

      IF (rege = x"0000004") THEN   --     PT3
            pt3 <= '1';
      ELSE
            pt3 <= '0';
      END IF ;

      IF (rege = x"0000008") THEN   --     PT4
            pt4 <= '1';
      ELSE
```

```
      pt4 <= '0';
END IF ;

IF (rege = x"0000010") THEN   --     PT5
      pt5 <= '1';
ELSE
      pt5 <= '0';
END IF ;

IF (rege = x"0000020") THEN   --     PT6
      pt6 <= '1';
ELSE
      pt6 <= '0';
END IF ;

IF (rege = x"0000040") THEN   --     PT7
      pt7 <= '1';
ELSE
      pt7 <= '0';
END IF ;

IF (rege = x"0000080") THEN   --     PT8
      pt8 <= '1';
ELSE
      pt8 <= '0';
END IF ;

IF (rege = x"0000100") THEN   --     PT9
      pt9 <= '1';
ELSE
      pt9 <= '0';
END IF ;

IF (rege = x"0000200") THEN   --     PT10
      pt10 <= '1';
ELSE
      pt10 <= '0';
END IF ;

IF (rege = x"0000400") THEN   --     PT11
      pt11 <= '1';
ELSE
      pt11 <= '0';
END IF ;

IF (rege = x"0000800") THEN   --     PT12
      pt12 <= '1';
ELSE
      pt12 <= '0';
END IF ;

IF (rege = x"0001000") THEN   --     PT13
      pt13 <= '1';
ELSE
```

```
      pt13 <= '0';
END IF ;

IF (rege = x"0002000") THEN   --     PT14
      pt14 <= '1';
ELSE
      pt14 <= '0';
END IF ;

IF (rege = x"0004000") THEN   --     PT15
      pt15 <= '1';
ELSE
      pt15 <= '0';
END IF ;

IF (rege = x"0008000") THEN   --     PT16
      pt16 <= '1';
ELSE
      pt16 <= '0';
END IF ;

IF (rege = x"0010000") THEN   --     PT17
      pt17 <= '1';
ELSE
      pt17 <= '0';
END IF ;

IF (rege = x"0020000") THEN   --     PT18
      pt18 <= '1';
ELSE
      pt18 <= '0';
END IF ;

IF (rege = x"0040000") THEN   --     PT19
      pt19 <= '1';
ELSE
      pt19 <= '0';
END IF ;

IF (rege = x"0080000") THEN   --     PT20
      pt20 <= '1';
ELSE
      pt20 <= '0';
END IF ;

IF (rege = x"0100000") THEN   --     PT21
      pt21 <= '1';
ELSE
      pt21 <= '0';
END IF ;

IF (rege = x"0200000") THEN   --     PT22
      pt22 <= '1';
ELSE
```

```
            pt22 <= '0';
      END IF ;

      IF (rege = x"0400000") THEN   —     PT23
            pt23 <= '1';
      ELSE
            pt23 <= '0';
      END IF ;

      IF (rege = x"0800000") THEN   —     PT24
            pt24 <= '1';
      ELSE
            pt24 <= '0';
      END IF ;

      IF (rege = x"1000000") THEN   —     PT25
            pt25 <= '1';
      ELSE
            pt25 <= '0';
      END IF ;

      IF (rege = x"2000000") THEN   —     PT26
            pt26 <= '1';
      ELSE
            pt26 <= '0';
      END IF ;

      IF (rege = x"4000000") THEN   —     PT27
            pt27 <= '1';
      ELSE
            pt27 <= '0';
      END IF ;

      IF (rege = x"8000000") THEN   —     PT28
            pt28 <= '1';
      ELSE
            pt28 <= '0';
      END IF ;
END PROCESS reg_e;

————————————————————————————————————————-
———————————— Shared Product Terms Example ————————————-
————————————————————————————————————————-

y <=  pt1 OR pt2 OR pt3 OR pt4 OR pt5 OR pt6 OR pt7 OR
            pt8 OR pt9 OR pt10 OR pt11 OR pt12 OR pt13 OR pt14 OR
            pt15 OR pt16 OR pt17 OR pt18 OR pt19 OR pt20 OR pt21 OR
            pt22 OR pt23 OR pt24 OR pt25 OR pt26 OR pt27 OR pt28;

END behav ;

Listing 5 controls.vhd
LIBRARY IEEE,STD;
USE IEEE.std_logic_1164.all ;
```

```
USE IEEE.std_logic_unsigned.all ;

— Sample VHDL code to demonstrate 8840 I/O cell control bus


ENTITY training IS
      PORT (
            d                       :   IN            std_logic_vector(3 downto 0);
            clk                     :   IN            std_logic;
            in_clk                        :   IN            std_logic;
            in_1, in_2              :   IN            std_logic;
            in_3, in_4              :   IN            std_logic;
            clear                   :   IN            std_logic;
            rega, regb              :   OUT           std_logic_vector(3 downto 0);
            regc, regd              :   OUT           std_logic
      );

END training ;


ARCHITECTURE behav OF training IS

      signal shared_pt_clk                : std_logic;
      signal dedicated_pt_clk             : std_logic;

      signal regd_int                           : std_logic;
      signal control_pt_clk               : std_logic;
      signal control_pt_clken             : std_logic;
      signal control_pt_reset             : std_logic;
      signal control_pt_preset            : std_logic;
      signal control_pt_oe                : std_logic;

begin

————————————————————————————————————————-
———————————— Global Clock ————————————————
————————————————————————————————————————-

reg_a: PROCESS (clk, clear)
BEGIN
      IF (clear = ‘0’) THEN
            rega <= “0000”;
      ELSIF (clk’event and clk = ‘1’) THEN
            rega <= d;
      END IF ;
END PROCESS reg_a;

————————————————————————————————————————-
———————————— Shared PT Clock Example ————————————-
————————————————————————————————————————-

reg_b: PROCESS (shared_pt_clk, clear)
BEGIN
      IF (clear = ‘0’) THEN
```

```
            regb <= “0000”;
      ELSIF (shared_pt_clk’event and shared_pt_clk = ‘1’) THEN
            regb <= d;
      END IF ;
END PROCESS reg_b;

shared_pt_clk <= in_1 and in_clk;

-------------------------------------------
--------------- Dedicated PT Clock Example ---------------
-------------------------------------------

reg_c: PROCESS (dedicated_pt_clk, clear)
BEGIN
      IF (clear = ‘0’) THEN
            regc <= ‘0’;
      ELSIF (dedicated_pt_clk’event and dedicated_pt_clk = ‘1’) THEN
            regc <= d(0);
      END IF ;
END PROCESS reg_c;

dedicated_pt_clk <= in_2 and in_clk;

-------------------------------------------
-------------- I/O Cell Control Bus Example --------------
-------------------------------------------

reg_d: PROCESS (control_pt_clk, control_pt_clken, control_pt_reset,
control_pt_preset)
BEGIN
      IF (control_pt_reset = ‘0’ and (control_pt_preset) = ‘1’) THEN
            regd_int <= ‘0’;
      ELSIF (control_pt_reset = ‘0’ and (control_pt_preset) = ‘0’) THEN
            regd_int <= ‘0’;
      ELSIF (control_pt_reset = ‘1’ and (control_pt_preset) = ‘0’) THEN
            regd_int <= ‘1’;
      ELSIF (control_pt_clk’event and control_pt_clk = ‘1’) THEN
            IF (control_pt_clken = ‘1’) THEN
                  regd_int <= d(0);
            END IF ;
      END IF ;
END PROCESS reg_d;

control_pt_clk <= in_3 and in_clk;
control_pt_clken <= in_1 and in_2;
control_pt_reset <= in_2 and in_3;
control_pt_preset <= in_3 and in_4;
control_pt_oe <= in_4 and in_1;

regd <= regd_int when control_pt_oe = ‘0’ else ‘Z’ ;


END behav ;
```