2

Coding if and case Statements for Late Arriving Signals

Designers usually know which signals in a design are late arriving signals. This knowledge can be used to structure HDL so that the late arriving signals appear closer to the output. This chapter gives examples of restructuring if and case statements for late arriving signals.

Sequential if Statements: Late Arriving Data Signal

Sequential if statements allow you to structure your HDL based on critical signals. You may want to use sequential if statements, as shown in Example 1-1 on page 1-2 and Example 1-2 on page 1-3, if the input d in those examples is a late arriving signal. Figure 1-1 on page 1-4 shows that d is an input to the last SELECT_OP in the chain. Therefore, d is as close to the output as possible.

However, if b is a late arriving signal, Example 1-1 on page 1-2 and Example 1-2 on page 1-3 do not show the optimal coding style. When b is the late arriving signal, the optimal coding style involves restructuring the HDL as shown in Example 2-1 and Example 2-2. This restructuring moves b (`b_is_late` in Example 2-1 and Example 2-2) closer to the output `z`.

Example 2-1 Improved Verilog for Priority Encoded if

```
module mult_if_improved(a, b_is_late, c, d, sel, z);
input a, b_is_late, c, d;
input [3:0] sel;
output z;
reg z, z1;

always @(a or b_is_late or c or d or sel)
begin
     z1 = 0;
     if (sel[0])
          z1 = a;
     if (sel[2])
          z1 = c;
     if (sel[3])
          z1 = d;
     if (sel[1] & ~(sel[2]|sel[3]))
          z = b_is_late;
     else
          z = z1;
end
endmodule
```

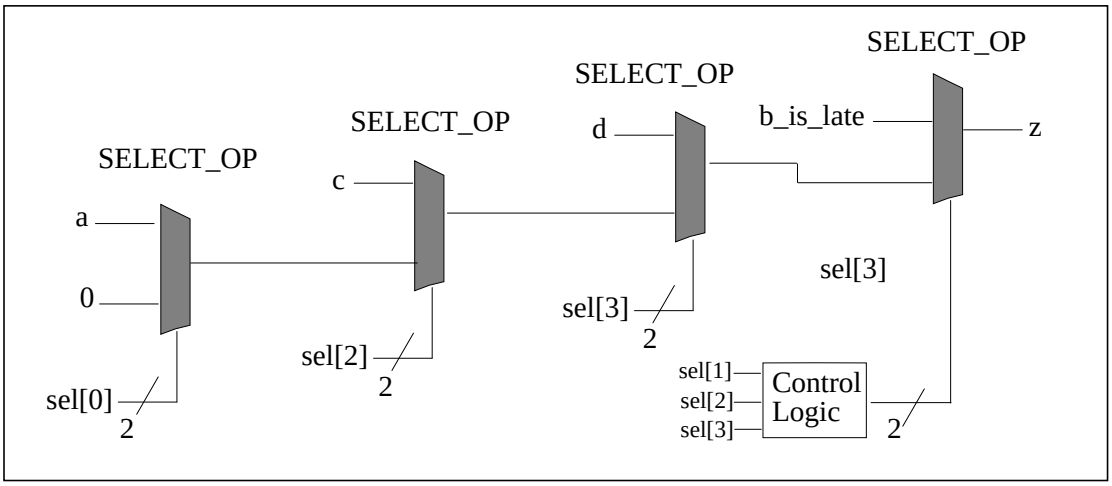
Example 2-2 is the equivalent VHDL example.

Example 2-2 Improved VHDL for Priority Encoded if

```
library IEEE;
use IEEE.std_logic_1164.all;

entity mult_if_improved is
port (a, b_is_late, c, d: in std_logic;
        sel: in std_logic_vector(3 downto 0);
        z: out std_logic);
end mult_if_improved;

architecture one of mult_if_improved is
signal z1: std_logic;
begin
        process (a,b_is_late,c,d,sel,z1)
        begin
             z1 <= '0';
             if (sel(0) = '1') then
                  z1 <= a;
             end if;
             if (sel(2) = '1') then
                  z1 <= c;
             end if;
             if (sel(3) = '1') then
                  z1 <= d;
             if ((sel(1) = '1') and ((sel(2) or sel(3)) = '0')) then
                  z <= b_is_late;
             else
                  z = z1;
             end if;
        end process;
end one;
```

Figure 2-1 shows the structure implied by the HDL in Example 2-1 and Example 2-2.

Figure 2-1 Improved Structure: Priority Encoded b if Late Arriving Signal



Single if Statement: Late Arriving Control Signal

If a single if statement, as shown in Example 1-3 and Example 1-4 on page 1-6, has a late arriving signal as a condition in one of the branches of the if statement, consider the following Verilog and VHDL examples, where `CTRL_is_late_arriving` is a late arriving input signal.

Example 2-3 Verilog Example of Single if With Late Arriving Control Signal

```
module single_if_late(A, C, CTRL_is_late_arriving, Z);
input [6:1] A;
input [5:1] C;
input CTRL_is_late_arriving;
output Z;
reg Z;

always @(C or A or CTRL_is_late_arriving)
begin
    if (C[1] == 1'b1)
        Z = A[1];
    else if (C[2] == 1'b0)
        Z = A[2];
    else if (C[3] == 1'b1)
        Z = A[3];
    else if (C[4] == 1'b1 && CTRL_is_late_arriving == 1'b0)
                // late arriving signal in if condition
        Z = A[4];

    else if (C[5] == 1'b0)
        Z = A[5];
    else
        Z = A[6];
end
endmodule
```

Example 2-4 shows the equivalent VHDL.

Example 2-4 VHDL Example of Single if With Late Arriving Control Signal

```
library IEEE;
use IEEE.std_logic_1164.all;

entity single_if_late is
port (A: in std_logic_vector(6 downto 1);
        C: in std_logic_vector(5 downto 1);
        CTRL_is_late_arriving: in std_logic;
        Z: out std_logic);
end single_if_late;

architecture one of single_if_late is
begin
        process(A, C, CTRL_is_late_arriving)
        begin
             if (C(1) = '1') then
                  Z <= A(1);
             elsif (C(2) = '0') then
                  Z <= A(2);
             elsif (C(3) = '1') then
                  Z <= A(3);
             elsif (C(4) = '1' and CTRL_is_late_arriving = '0') then
                  Z <= A(4);
             elsif (C(5) = '0') then
                  Z <= A(5);
             else
                  Z <= A(6);
             end if;
        end process;
end one;
```

Example 2-3 and Example 2-4 result in a structure similar to that shown in Figure 1-2 on page 1-7. The control logic between Example 2-3 and Example 2-4 is slightly different due to the logical AND (&&) in one of the if conditions in Example 2-3.

Because CTRL_is_late_arriving is a late arriving signal, the objective is to push this signal as far out as possible. That is, the signal should be as close to the output port, Z, as possible in the structure implied by the HDL.

Example 2-5 and Example 2-6 show the modified Verilog and VHDL that achieve this objective. The signal `CTRL_is_late_arriving` is pulled out of the `if...else if` statement and is thereby pushed closer to the output.

Example 2-5 Improved Verilog for Single if With Late Arriving Control Signal

```
module single_if_improved(A, C, CTRL_is_late_arriving, Z);
input [6:1] A;
input [5:1] C;
input CTRL_is_late_arriving;
output Z;
reg Z;
reg Z1;
wire Z2, prev_cond;

always @(A or C)
begin
        if (C[1] == 1'b1)
             Z1 = A[1];
        else if (C[2] == 1'b0)
             Z1 = A[2];
        else if (C[3] == 1'b1)
             Z1 = A[3];
        else if (C[5] == 1'b0)// removed the branch with the
             Z1 = A[5];      //late-arriving control signal
        else
             Z1 = A[6];
end

assign Z2 = A[4];
assign prev_cond = (C[1] == 1'b1) || (C[2] == 1'b0) || (C[3] == 1'b1);

always @(C or prev_cond or CTRL_is_late_arriving or Z1 or Z2)
begin
        if (C[4] == 1'b1 && CTRL_is_late_arriving == 1'b0)
             if (prev_cond)
                  Z = Z1;
             else
                  Z = Z2;
             else
                  Z = Z1;
        end
endmodule
```

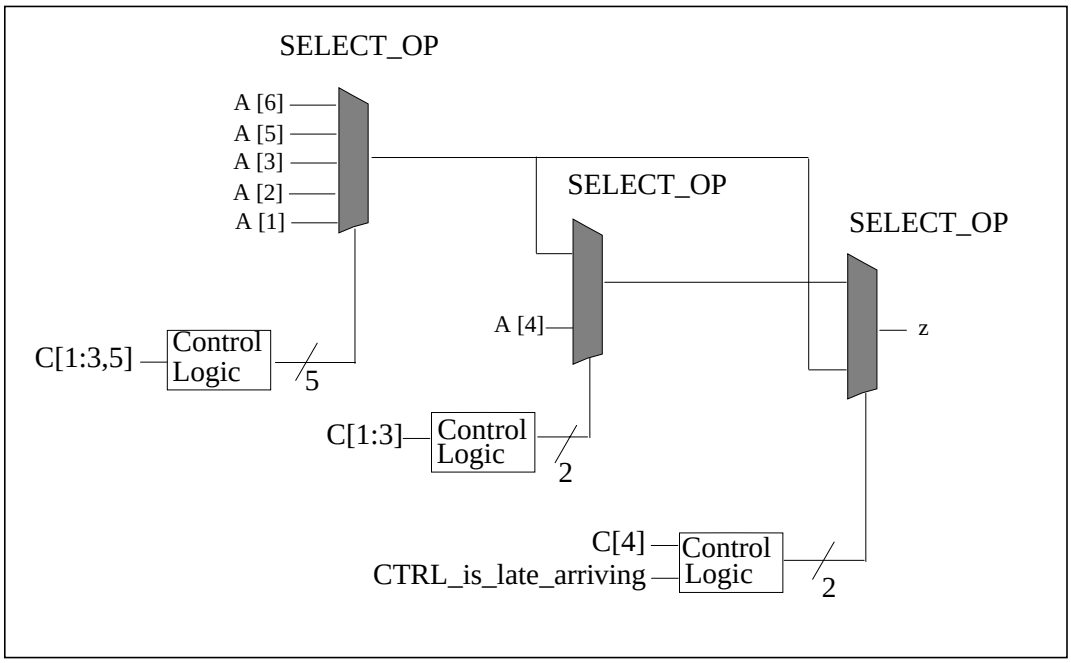
Example 2-6 shows the equivalent improved VHDL.

Example 2-6 Improved VHDL for Single if With Late Arriving Control Signal

```
library IEEE;
use IEEE.std_logic_1164.all;

entity single_if_improved is
port (A: in std_logic_vector(6 downto 1);
        C: in std_logic_vector(5 downto 1);
        CTRL_is_late_arriving: in std_logic;
        Z: out std_logic);
end single_if_improved;

architecture one of single_if_improved is
signal Z1, Z2: std_logic;
signal prev_cond: boolean;
begin
        Z2 <= A(4);
        prev_cond <= ((C(1) = '1') or (C(2) = '0') or (C(3) = '1'));
        process(A, C)
        begin
             if (C(1) = '1') then
                  Z1 <= A(1);
             elsif (C(2) = '0') then
                  Z1 <= A(2);
             elsif (C(3) = '1') then
                  Z1 <= A(3);
             elsif (C(5) = '0') then    -- removed the branch with the late
                  Z1 <= A(5);           -- arriving control signal
             else
                  Z1 <= A(6);
             end if;
        end process;

        process(C, prev_cond, CTRL_is_late_arriving, Z1, Z2)
        begin
             if ((C(4) = '1') and (CTRL_is_late_arriving = '0')) then
                  if (prev_cond) then
                       Z <= Z1;
                  else
                       Z <= Z2;
                  end if;
             else
                  Z <= Z1;
             end if;
        end process;
end one;
```

Figure 2-2 shows the structure implied by the improved HDL.

Figure 2-2 Improved Structure for Single if With Late Arriving Control Signal



Now that you have seen some basic if and case statements, you can apply the information to some more-complex examples (with late arriving signals) that use if and case statements together.

if Statement With Nested case Statement: Late Arriving Data Signal

Example 2-7 is a Verilog example of a case statement nested in an if statement. The data signal `DATA_is_late_arriving` in one branch of the case statement is a late arriving signal.

Example 2-7 Original Verilog for case Statement in if Statement

```
module case_in_if_01(A, DATA_is_late_arriving, C, sel, Z);
input [8:1] A;
input DATA_is_late_arriving;
input [2:0] sel;
input [5:1] C;
output Z;
reg Z;

always @ (sel or C or A or DATA_is_late_arriving)
begin
    if (C[1])
        Z = A[5];
    else if (C[2] == 1'b0)
        Z = A[4];
    else if (C[3])
        Z = A[1];
    else if (C[4])
        case (sel)
            3'b010: Z = A[8];
            3'b011: Z = DATA_is_late_arriving;
            3'b101: Z = A[7];
            3'b110: Z = A[6];
            default: Z = A[2];
        endcase
    else if (C[5] == 1'b0)
        Z = A[2];
    else
        Z = A[3];
end
endmodule
```

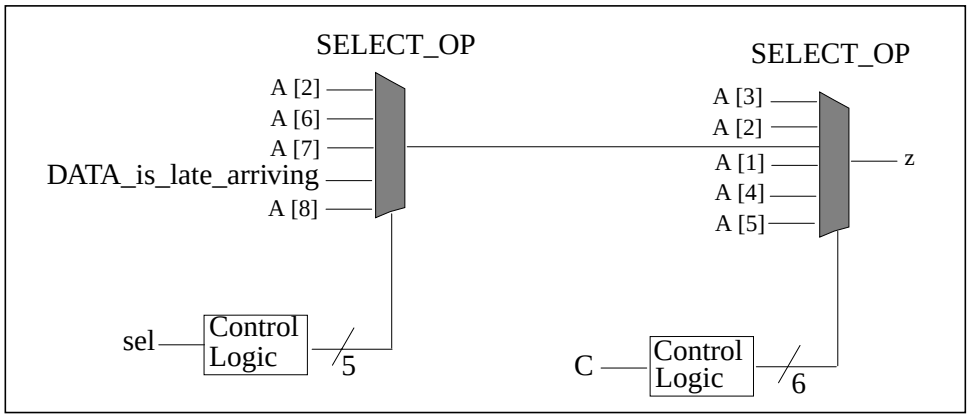
Example 2-8 is the VHDL equivalent of Example 2-7.

Example 2-8 Original VHDL for case Statement in if Statement

```
library IEEE;
use IEEE.std_logic_1164.all;

entity case_in_if_01 is
port(A: in std_logic_vector(8 downto 1);
     DATA_is_late_arriving: in std_logic;
     sel: in std_logic_vector(2 downto 0);
     C: in std_logic_vector(5 downto 1);
     Z: out std_logic);
end case_in_if_01;

architecture orig of case_in_if_01 is
begin
     process(sel, C, A, DATA_is_late_arriving)
     begin
          if (C(1) = '1') then
               Z <= A(5);
          elsif (C(2) = '0') then
               Z <= A(4);
          elsif (C(3) = '1') then
               Z <= A(1);
          elsif (C(4) = '1') then
               case sel is
                    when "010" =>Z <= A(8);
                    when "011" =>Z <= DATA_is_late_arriving;
                    when "101" =>Z <= A(7);
                    when "110" =>Z <= A(6);
                    when others =>Z <= A(2);
               end case;
          elsif (C(5) = '0') then
               Z <= A(2);
          else
               Z <= A(3);
          end if;
     end process;
 end orig;
```

Figure 2-3 shows the structure implied by the HDL in Example 2-7 and Example 2-8.

Figure 2-3 Structure Implied by Original HDL in Example 2-7, Example 2-8



You know that `DATA_is_late_arriving` is a late arriving signal in Example 2-7 and Example 2-8. In Figure 2-3, this signal is an input to the first `SELECT_OP` in the path. To improve the startpoint for synthesis, modify the HDL to move `DATA_is_late_arriving` closer to the output `Z`.

You can do this by moving the assignment of `DATA_is_late_arriving` out of the nested case statement into a separate if statement. This makes `DATA_is_late_arriving` an input to another `SELECT_OP` that is closer to the output port `Z`.

Example 2-9 shows the improved version of the Verilog shown in Example 2-7.

Example 2-9 Improved Verilog for case Statement in if Statement

```
module case_in_if_01_improved(A,DATA_is_late_arriving,C,sel,Z);
input [8:1] A;
input DATA_is_late_arriving;
input [2:0] sel;
input [5:1] C;
output Z;
reg Z;
reg Z1, FIRST_IF;

always @(sel or C or A or DATA_is_late_arriving)
begin
        if (C[1])
             Z1 = A[5];
        else if (C[2] == 1'b0)
             Z1= A[4];
        else if (C[3])
             Z1 = A[1];
        else if (C[4])
             case (sel)
                  3'b010: Z1 = A[8];
             //3'b011: Z1 = DATA_is_late_arriving;
                  3'b101: Z1 = A[7];
                  3'b110: Z1 = A[6];
                  default: Z1 = A[2];
             endcase
        else if (C[5] == 1'b0)
             Z1 = A[2];
        else
             Z1 = A[3];

        FIRST_IF = (C[1] == 1'b1) || (C[2] == 1'b0) || (C[3] == 1'b1);

        if (!FIRST_IF && C[4] && (sel == 3'b011))
             Z = DATA_is_late_arriving;
        else
             Z = Z1;
end
endmodule
```

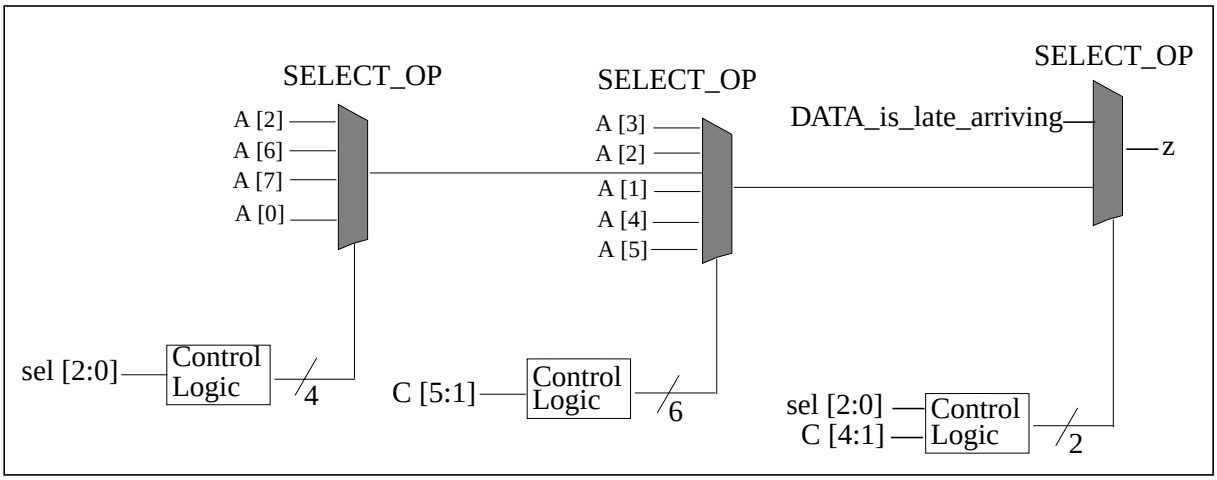
Example 2-10 Improved VHDL for case Statement in if Statement

```
library IEEE;
use IEEE.std_logic_1164.all;

entity case_in_if_01 is
port(A: in std_logic_vector(8 downto 1);
        DATA_is_late_arriving: in std_logic;
        sel: in std_logic_vector(2 downto 0);
        C: in std_logic_vector(5 downto 1);
        Z: out std_logic);
end case_in_if_01;

architecture improved of case_in_if_01 is
begin
        process(sel, C, A, DATA_is_late_arriving)
        variable Z1: std_logic;
        variable FIRST_IF: boolean;
        begin
             if (C(1) = '1') then
                  Z1 := A(5);
             elsif (C(2) = '0') then
                  Z1 := A(4);
             elsif (C(3) = '1') then
                  Z1 := A(1);
             elsif (C(4) = '1') then
                  case sel is
                       when "010" => Z1 := A(8);
                  -- when "011" => Z1 := DATA_is_late_arriving;
                       when "101" => Z1 := A(7);
                       when "110" => Z1 := A(6);
                       when others => Z1 := A(2);
                  end case;
             elsif (C(5) = '0') then
                  Z1 := A(2);
             else
                  Z1 := A(3);
             end if;

             FIRST_IF := (C(1) = '1') OR (C(2) = '0') OR (C(3) = '1');

             if (NOT FIRST_IF AND (C(4) = '1') AND (sel = "011")) then
                  Z <= DATA_is_late_arriving;
             else
                  Z <= Z1;
             end if;
        end process;
end improved;
```

Example 2-10 is an improved version of the VHDL shown in Example 2-8.

The structure implied by the modified HDL in Example 2-9 and Example 2-10 is given in Figure 2-4.

Figure 2-4 Structure Implied by Improved HDL in Example 2-9, Example 2-10



case Statement With Nested if Statement: Late Arriving Control Signal

Example 2-11 and Example 2-12 show Verilog and VHDL for an if statement nested in a case statement. These examples assume `sel[1]` is a late arriving signal.

Example 2-11 Original Verilog for if Statement in case Statement

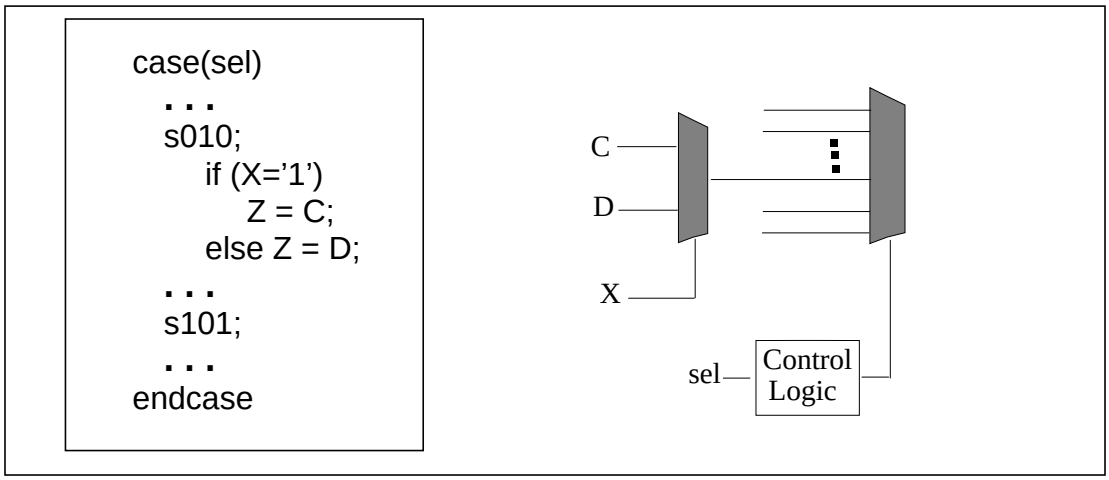
```
module if_in_case(sel, X, A, B, C, D, Z);
input [2:0] sel;  // sel[1] is late arriving
input X, A, B, C, D;
output Z;
reg Z;

always @(sel or X or A or B or C or D)
begin
    case (sel)
        3'b000: Z = A;
        3'b001: Z = B;
        3'b010: if (X == 1'b1)
                    Z = C;
                else
                    Z = D;
        3'b100: Z = A ^ B;
        3'b101: Z = !(A && B);
        3'b111: Z = !A;
        default: Z = !B;
    endcase
end
endmodule
```

Example 2-12 Original VHDL for if Statement in case Statement

```
library IEEE;
use IEEE.std_logic_1164.all;

entity if_in_case is
port(sel: in std_logic_vector(2 downto 0);
          -- sel(1) is late arriving
    X, A, B, C, D: in std_logic;
    Z: out std_logic);
end;

architecture orig of if_in_case is
begin
    process(sel, X, A, B, C, D)
    begin
        case sel is
            when ”000” => Z <= A;
            when ”001” => Z <= B;
            when ”010” =>if (X = ’1’) then
                             Z <= C;
                         else
                             Z <= D;
                         end if;
            when ”100” => Z <= A XOR B;
            when ”101” => Z <= A NAND B;
            when ”111” => Z <= NOT A;
            when others => Z <= NOT B;
        end case;
    end process;
end orig;
```

Figure 2-5 shows pseudo-HDL for an if statement nested in a case statement and the structure implied by this HDL.

Figure 2-5 Structure Implied by if Statement Nested in case Statement



In Figure 2-5, due to the control logic, there is a large delay between `sel` and the output Z. If `sel[1]` is a late arriving input, you want to minimize this delay.

In Example 2-11 and Example 2-12, you know that `sel[1]` is a late arriving input. Therefore, you should restructure the HDL in Example 2-11 and Example 2-12 to get the best startpoint for synthesis.

Example 2-13 and Example 2-14 show the modified Verilog and VHDL. The HDL has been modified to shift the dependency on `sel[1]` closer to the output; that is, to move `sel[1]` closer to the output, the nested if statement has been removed and placed outside the case statement.

Example 2-13 Improved Verilog for if Statement in case Statement

```
module if_in_case_improved(sel, X, A, B, C, D, Z);
input [2:0] sel;  // sel[1] is late arriving
input X, A, B, C, D;
output Z;
reg Z;
reg Z1, Z2;
reg [1:0] i_sel;

always @ (sel or X or A or B or C or D)
begin
    i_sel = {sel[2],sel[0]};
    case (i_sel)  // for sel[1]=0
        2'b00: Z1 = A;
        2'b01: Z1 = B;
        2'b10: Z1 = A ^ B;
        2'b11: Z1 = !(A && B);
        default: Z1 = !B;
    endcase

    case (i_sel)  // for sel[1]=1
        2'b00:if (X == 1'b1)
                    Z2 = C;
                else
                    Z2 = D;
        2'b11: Z2 = !A;
        default: Z2 = !B;
    endcase
    if (sel[1])
        Z = Z2;
    else
        Z = Z1;
end
endmodule
```

Example 2-14 Improved VHDL for if Statement in case Statement

```
library IEEE;
use IEEE.std_logic_1164.all;

entity if_in_case is
port(sel: in std_logic_vector(2 downto 0);
            -- sel(1) is late arriving
        X, A, B, C, D: in std_logic;
        Z: out std_logic);
end if_in_case;

architecture improved of if_in_case is
begin
    process(sel, X, A, B, C, D)
    variable Z1, Z2: std_logic;
    variable i_sel: std_logic_vector(1 downto 0);
    begin
        i_sel := sel(2) & sel(0);
        case i_sel is   -- for sel(1)= '0'
            when "00" => Z1 := A;
            when "01" => Z1 := B;
            when "10" => Z1 := A XOR B;
            when "11" => Z1 := A NAND B;
            when others => Z1 := NOT B;
        end case;

        case i_sel is   -- for sel(1)= '1'
            when "00" => if (X = '1') then
                             Z2 := C;
                         else
                             Z2 := D;
                         end if;
            when "11" => Z2 := NOT A;
            when others => Z2 := NOT B;
        end case;

        if (sel(1) = '1') then
            Z <= Z2;
        else
            Z <= Z1;
        end if;
```

```
    end process;
end improved;
```

Table 2-1 shows the area and timing results for the original and improved versions of the designs in Example 2-11 through Example 2-14. The timing results are for the path from `sel[1]` to `Z`.

Table 2-1 Timing and Area Results for if Statement in case Statement

	Data Arrival Time	**Area**
Original Design	2.81	47.7
Improved Design	2.36	44.4

Notice the improvements in timing and area.