Chapter 6

Figure 6.1 A 2-to-1 multiplexer

(a) Graphical symbol  
(b) Truth table  
(c) Sum-of-products circuit  
(d) Circuit with transmission gates

Figure 6.2 A 4-to-1 multiplexer

(a) Graphical symbol  
(b) Truth table  
(c) Circuit

Figure 6.3 Using 2-to-1 multiplexers to build a 4-to-1 multiplexer

Figure 6.4 A 16-to-1 multiplexer

Figure 6.5 A practical application of multiplexers

(a) A 2x2 crossbar switch  
(b) Implementation using multiplexers
Implementing programmable switches in an FPGA

Figure 6.6

(a) Part of the FPGA in Figure 3.39

(b) Implementation using pass transistors

(c) Implementation using multiplexers

Figure 6.7

Synthesis of a logic function using multiplexers

(a) Implementation using a 4-to-1 multiplexer

(b) Modified truth table

(c) Circuit

Figure 6.8

Three-input majority function

(a) Modified truth table

(b) Circuit

Figure 6.9

Three-input XOR function

(a) Truth table

(b) Circuit

Figure 6.10

Three-input XOR function

(a) Truth table

(b) Circuit

Figure 6.11

Three-input majority function using a 2-to-1 MUX

(a) Modified truth table

(b) Circuit
Boole’s (Shannon’s) Expansion

\[ f(w_1, w_2, ..., w_n) = w_1 \cdot f(0, w_2, ..., w_n) + w_1 \cdot f(1, w_2, ..., w_n) \]

\[ = w_1 \cdot f_1 + w_1 \cdot f_2 \]

\[ = \bar{w}_1 \cdot \bar{w}_2 \cdot f(0,0,w_3, ..., w_n) + \]

\[ \bar{w}_1 \cdot w_2 \cdot f(0,1,w_3, ..., w_n) + \]

\[ w_1 \cdot \bar{w}_2 \cdot f(1,0,w_3, ..., w_n) + \]

\[ w_1 \cdot w_2 \cdot f(1,1,w_3, ..., w_n) \]

\[ f = w_1 w_2 + w_1 w_3 + w_2 w_3 \]

\[ f = w_1 w_3 + w_2 w_3 \]

\[ f = w_1 w_3' + w_1 w_2 + w_1 w_3 \]

\[ f = w_1 w_2 + w_1 w_3 + w_2 w_3 \]

Figure 6.13: Example circuits
Figure 6.13 Example circuit

\[ f = w_2 \cdot w_3 + w_1 \cdot w_2 \cdot w_3' + w_2 \cdot w_3' \cdot w_4 + w_1 \cdot w_2' \cdot w_4' \]

Figure 6.14 Example circuits

(a) Using three 3-LUTs

(b) Using two 3-LUTs

Figure 6.15 An n-to-2^n decoder

Figure 6.16 A 2-to-4 decoder

Figure 6.17 A 3-to-8 decoder using two 2-to-4 decoder
Demultiplexers

- A demultiplexer is a circuit which places the value of a single data input onto multiple data outputs is a demultiplexer.

![Demultiplexer Diagram](image_url)
Figure 6.23  A 4-to-2 binary encoder

<table>
<thead>
<tr>
<th>w3 w2 w1 w0</th>
<th>y1 y0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 1</td>
<td>0 0</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>0 1</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>1 0</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>1 1</td>
</tr>
</tbody>
</table>

(a) Truth table

(b) Circuit

Figure 6.24  Truth table for a 4-to-2 priority encoder

<table>
<thead>
<tr>
<th>w2 w1 w0</th>
<th>y0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>d</td>
</tr>
<tr>
<td>0 0 1</td>
<td>d</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1 1</td>
</tr>
</tbody>
</table>

Figure 6.25  A BCD-to-7-segment display code converter

<table>
<thead>
<tr>
<th>w3 w2 w1 w0</th>
<th>a b c d e f g</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0 1 1 1 1 0 0</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>0 1 0 1 1 0 0</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>0 1 0 1 1 0 1</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>0 1 0 0 1 1 1</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>0 1 0 0 1 1 0</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>0 1 0 0 1 1 0</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>0 1 0 0 1 1 1</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>0 1 0 0 1 1 0</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>0 1 0 0 1 1 1</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>0 1 0 0 1 0 1</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>0 1 0 0 1 0 0</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>0 1 0 0 1 0 0</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>0 1 0 0 1 0 1</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>0 1 0 0 1 0 0</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>0 1 0 0 1 0 1</td>
</tr>
</tbody>
</table>

(c) Truth table

Figure 6.26  A four-bit comparator circuit

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY mux2to1 IS
PORT ( w0, w1, s : IN STD_LOGIC ;
       f : OUT STD_LOGIC );
END mux2to1 ;
ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
  WITH s SELECT
    f <= w0 WHEN '0',
        w1 WHEN OTHERS ;
END Behavior ;

Figure 6.27  VHDL code for a 2-to-1 multiplexer

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY mux4to1 IS
PORT ( w0, w1, w2, w3 : IN STD_LOGIC ;
       s : IN STD_LOGIC_VECTOR(1 DOWNTO 0) ;
       f : OUT STD_LOGIC );
END mux4to1 ;
ARCHITECTURE Behavior OF mux4to1 IS
BEGIN
  WITH s SELECT
    f <= w0 WHEN "00",
        w1 WHEN "01",
        w2 WHEN "10",
        w3 WHEN OTHERS ;
END Behavior ;

Figure 6.28  VHDL code for a 4-to-1 multiplexer
LIBRARY ieee;
USE ieee.std_logic_1164.all;
PACKAGE mux4to1_package IS
COMPONENT mux4to1
PORT ( w0, w1, w2, w3 : IN STD_LOGIC;
       s : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
       f : OUT STD_LOGIC)
END COMPONENT;
END mux4to1_package;

LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY work;
USE work.mux4to1_package.all;
ENTITY mux16to1 IS
PORT ( w : IN STD_LOGIC_VECTOR(0 TO 15);
       s : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
       f : OUT STD_LOGIC)
END mux16to1;
ARCHITECTURE Structure OF mux16to1 IS
SIGNAL m : STD_LOGIC_VECTOR(0 TO 3);
BEGIN
Mux1: mux4to1 PORT MAP( w(0), w(1), w(2), w(3), s(1 DOWNTO 0), m(0));
Mux2: mux4to1 PORT MAP( w(4), w(5), w(6), w(7), s(1 DOWNTO 0), m(1));
Mux3: mux4to1 PORT MAP( w(8), w(9), w(10), w(11), s(1 DOWNTO 0), m(2));
Mux4: mux4to1 PORT MAP( w(12), w(13), w(14), w(15), s(1 DOWNTO 0), m(3));
Mux5: mux4to1 PORT MAP( m(0), m(1), m(2), m(3), s(3 DOWNTO 2), f);
END Structure;

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY dec2to4 IS
PORT ( w : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
        En : IN STD_LOGIC;
        y : OUT STD_LOGIC_VECTOR(0 TO 3))
END dec2to4;
ARCHITECTURE Behavior OF dec2to4 IS
SIGNAL Enw : STD_LOGIC_VECTOR(2 DOWNTO 0);
BEGIN
Enw <= En & w;
WITH Enw SELECT
     y <= "1000" WHEN "100",
         "0100" WHEN "101",
         "0010" WHEN "110",
         "0001" WHEN "111",
         "0000" WHEN OTHERS;
END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY priority IS
PORT ( w : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
       y : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
       z : OUT STD_LOGIC)
END priority;
ARCHITECTURE Behavior OF priority IS
BEGIN
y <= "11" WHEN w(3) = '1' ELSE "10" WHEN w(2) = '1' ELSE "01" WHEN w(1) = '1' ELSE "00";
x <= '0' WHEN w = "0000" ELSE '1';
END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY priority IS
PORT ( w : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
       y : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
       z : OUT STD_LOGIC)
END priority;
ARCHITECTURE Behavior OF priority IS
BEGIN
WITH w SELECT
   y <= "00" WHEN "0000",
       "01" WHEN "0001",
       "10" WHEN "0010",
       "10" WHEN "0011",
       "10" WHEN "0100",
       "10" WHEN "0101",
       "10" WHEN "0110",
       "10" WHEN "0111",
       "10" WHEN OTHERS,
       "01" WHEN OTHERS;
   z <= '0' WHEN "0000" ELSE '1';
END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY priority IS
PORT ( w : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
       y : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
       z : OUT STD_LOGIC)
END priority;
ARCHITECTURE Behavior OF priority IS
BEGIN
WITH w SELECT
   y <= "00" WHEN "0000",
       "01" WHEN "0001",
       "10" WHEN "0010",
       "10" WHEN "0011",
       "10" WHEN "0100",
       "10" WHEN "0101",
       "10" WHEN "0110",
       "10" WHEN "0111",
       "10" WHEN OTHERS,
       "01" WHEN OTHERS;
   z <= '0' WHEN "0000" ELSE '1';
END Behavior;

Figure 6.28 Component declaration for the 4-to-1 multiplexer

Figure 6.29 Hierarchical code for a 16-to-1 multiplexer

Figure 6.30 VHDL code for a 2-to-4 binary decoder

Figure 6.31 A 2-to-1 multiplexer using a conditional signal assignment

Figure 6.32 VHDL code for a priority encoder

Figure 6.33 Less efficient code for a priority encoder
Concurrent vs. Sequential

- All previous statements are called **concurrent assignment statements** because order does not matter.
- When order matters, the statements are called **sequential assignment statements**.
- All sequential assignment statements are placed within a **process statement**.

Process Statement

- Begins with **PROCESS** keyword followed by a **sensitivity list**.
- For a combinational circuit, sensitivity list includes all input signals used in the process.
- Process executed whenever there is a change on a signal in the sensitivity list.
- Statements executed in sequential order.
- No assignments are visible until all statements in the process have been executed.
- If multiple assignments, only last has an effect.
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY mux2to1 IS
  PORT ( w0, w1, s : IN STD_LOGIC;
        f : OUT STD_LOGIC );
END mux2to1;

ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
  PROCESS ( w0, w1, s )
  BEGIN
    IF s = '0' THEN
      f <= w0;
    ELSE
      f <= w1;
    END IF;
  END PROCESS;
END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY mux2to1 IS
  PORT ( w0, w1, s : IN STD_LOGIC;
        f : OUT STD_LOGIC );
END mux2to1;

ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
  PROCESS ( w0, w1, s )
  BEGIN
    f <= w0;
    IF s = '1' THEN
      f <= w1;
    END IF;
  END PROCESS;
END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY priority IS
  PORT ( w : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        y : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
        z : OUT STD_LOGIC );
END priority;

ARCHITECTURE Behavior OF priority IS
BEGIN
  PROCESS ( w )
  BEGIN
    IF w(3) = '1' THEN
      y <= "11";
    ELSIF w(2) = '1' THEN
      y <= "10";
    ELSIF w(1) = '1' THEN
      y <= "01";
    ELSE
      y <= "00";
    END IF;
  END PROCESS;
  z <= '1' WHEN w = "0000" ELSE '0';
END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY priority IS
  PORT ( w : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        y : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
        z : OUT STD_LOGIC );
END priority;

ARCHITECTURE Behavior OF priority IS
BEGIN
  PROCES5 ( w )
  BEGIN
    y <= "00";
    IF w(1) = '1' THEN
      y <= "01";
    END IF;
    IF w(2) = '1' THEN
      y <= "10";
    END IF;
    IF w(3) = '1' THEN
      y <= "11";
    ELSE
      y <= "00";
    END IF;
  END PROCESS;
  z <= '1' WHEN w = "0000" THEN '0' ELSE '1';
END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY compare1 IS
  PORT ( A, B : IN STD_LOGIC;
        AeqB : OUT STD_LOGIC );
END compare1;

ARCHITECTURE Behavior OF compare1 IS
BEGIN
  PROCESS ( A, B )
  BEGIN
    AeqB <= '0';
    IF A = B THEN
      AeqB <= '1';
    END IF;
  END PROCESS;
END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY compare1 IS
  PORT ( A, B : IN STD_LOGIC;
        AeqB : OUT STD_LOGIC );
END compare1;

ARCHITECTURE Behavior OF compare1 IS
BEGIN
  PROCESS ( A, B )
  BEGIN
    AeqB <= '1';
    IF A = B THEN
      AeqB <= '0';
    END IF;
  END PROCESS;
END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY implied IS
  PORT ( A, B : IN STD_LOGIC;
        AeqB : OUT STD_LOGIC );
END implied;

ARCHITECTURE Behavior OF implied IS
BEGIN
  PROCESS ( A, B )
  BEGIN
    IF A = B THEN
      AeqB <= '1';
    END IF;
  END PROCESS;
END Behavior;

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY implied IS
  PORT ( A, B : IN STD_LOGIC;
        AeqB : OUT STD_LOGIC );
END implied;

ARCHITECTURE Behavior OF implied IS
BEGIN
  PROCESS ( A, B )
  BEGIN
    IF A = B THEN
      AeqB <= '0';
    END IF;
  END PROCESS;
END Behavior;
Figure 6.44: Circuit generated due to implied memory

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY mux2to1 IS
PORT ( w0, w1, s : IN STD_LOGIC;
      f : OUT STD_LOGIC )
END mux2to1;
ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
  PROCESS ( w0, w1, s )
  BEGIN
    CASE s IS
    WHEN '0' => f <= w0;
    WHEN OTHERS => f <= w1;
    END CASE;
  END PROCESS;
END Behavior;

Figure 6.45: A CASE statement that represents a 2-to-1 multiplexer

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
ENTITY alu IS
PORT ( s : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
       A, B : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
       F : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) )
END alu;
ARCHITECTURE Behavior OF alu IS
BEGIN
  PROCESS ( s, A, B )
  BEGIN
    CASE s IS
    WHEN "000" => F <= "0000";
    WHEN "001" => F <= B - A;
    WHEN "010" => F <= A - B;
    WHEN "011" => F <= A + B;
    WHEN "100" => F <= A XOR B;
    WHEN "101" => F <= A OR B;
    WHEN "110" => F <= A AND B;
    WHEN OTHERS => F <= "1111";
    END CASE;
  END PROCESS;
END Behavior;

Figure 6.46: A 2-to-1 multiplexer

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY seg7 IS
PORT ( bcd : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
       leds : OUT STD_LOGIC_VECTOR(1 TO 7) )
END seg7;
ARCHITECTURE Behavior OF seg7 IS
BEGIN
  PROCESS ( bcd )
  BEGIN
    CASE bcd IS -- abcdefg
    WHEN "0000" => leds <= "1111110";
    WHEN "0001" => leds <= "0110000";
    WHEN "0010" => leds <= "1101101";
    WHEN "0011" => leds <= "1111001";
    WHEN "0100" => leds <= "0110011";
    WHEN "0101" => leds <= "1011011";
    WHEN "0110" => leds <= "1011111";
    WHEN "0111" => leds <= "1110000";
    WHEN "1000" => leds <= "1111111";
    WHEN "1001" => leds <= "1110011";
    WHEN OTHERS => leds <= "-------";
    END CASE;
  END PROCESS;
END Behavior;

Figure 6.47: A BCD-to-7-segment decoder

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY alu1 IS
PORT ( s : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
       A, B : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
       F : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) )
END alu1;
ARCHITECTURE Behavior OF alu1 IS
BEGIN
  PROCESS ( s, A, B )
  BEGIN
    CASE s IS
    WHEN "000" => F <= "0000";
    WHEN "001" => F <= A - B;
    WHEN "010" => F <= A + B;
    WHEN "011" => F <= A XOR B;
    WHEN "100" => F <= A OR B;
    WHEN "101" => F <= A AND B;
    WHEN OTHERS => F <= "1111";
    END CASE;
  END PROCESS;
END Behavior;

Figure 6.48: Code that represents the functionality of the 74381 ALU

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY seg7 IS
PORT ( w0, w1, s : IN STD_LOGIC;
      f : OUT STD_LOGIC )
END seg7;
ARCHITECTURE Behavior OF seg7 IS
BEGIN
  PROCESS ( w0, w1, s )
  BEGIN
    IF A = B THEN
      AeqB <= '1';
    END IF;
  END PROCESS;
END Behavior;

Figure 6.49: A CASE statement that represents a 2-to-1 multiplexer

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY seg7 IS
PORT ( w0, w1, s : IN STD_LOGIC;
      f : OUT STD_LOGIC )
END seg7;
ARCHITECTURE Behavior OF seg7 IS
BEGIN
  PROCESS ( w0, w1, s )
  BEGIN
    IF A = B THEN
      AeqB <= '1';
    END IF;
  END PROCESS;
END Behavior;

Figure 6.46: A 2-to-1 multiplexer

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY alu IS
PORT ( s : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
       A, B : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
       F : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) )
END alu;
ARCHITECTURE Behavior OF alu IS
BEGIN
  PROCESS ( s, A, B )
  BEGIN
    CASE s IS
    WHEN "000" => F <= "0000";
    WHEN "001" => F <= B - A;
    WHEN "010" => F <= A - B;
    WHEN "011" => F <= A + B;
    WHEN "100" => F <= A XOR B;
    WHEN "101" => F <= A OR B;
    WHEN "110" => F <= A AND B;
    WHEN OTHERS => F <= "1111";
    END CASE;
  END PROCESS;
END Behavior;

Figure 6.48: Code that represents the functionality of the 74381 ALU

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY alu IS
PORT ( s : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
       A, B : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
       F : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) )
END alu;
ARCHITECTURE Behavior OF alu IS
BEGIN
  PROCESS ( s, A, B )
  BEGIN
    CASE s IS
    WHEN "000" => F <= "0000";
    WHEN "001" => F <= A - B;
    WHEN "010" => F <= A + B;
    WHEN "011" => F <= A XOR B;
    WHEN "100" => F <= A OR B;
    WHEN "101" => F <= A AND B;
    WHEN OTHERS => F <= "1111";
    END CASE;
  END PROCESS;
END Behavior;

Figure 6.48: Code that represents the functionality of the 74381 ALU

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY seg7 IS
PORT ( w0, w1, s : IN STD_LOGIC;
      f : OUT STD_LOGIC )
END seg7;
ARCHITECTURE Behavior OF seg7 IS
BEGIN
  PROCESS ( w0, w1, s )
  BEGIN
    IF A = B THEN
      AeqB <= '1';
    END IF;
  END PROCESS;
END Behavior;

Figure 6.49: A CASE statement that represents a 2-to-1 multiplexer

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY seg7 IS
PORT ( w0, w1, s : IN STD_LOGIC;
      f : OUT STD_LOGIC )
END seg7;
ARCHITECTURE Behavior OF seg7 IS
BEGIN
  PROCESS ( w0, w1, s )
  BEGIN
    IF A = B THEN
      AeqB <= '1';
    END IF;
  END PROCESS;
END Behavior;

Figure 6.46: A 2-to-1 multiplexer

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY alu IS
PORT ( s : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
       A, B : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
       F : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) )
END alu;
ARCHITECTURE Behavior OF alu IS
BEGIN
  PROCESS ( s, A, B )
  BEGIN
    CASE s IS
    WHEN "000" => F <= "0000";
    WHEN "001" => F <= B - A;
    WHEN "010" => F <= A - B;
    WHEN "011" => F <= A + B;
    WHEN "100" => F <= A XOR B;
    WHEN "101" => F <= A OR B;
    WHEN "110" => F <= A AND B;
    WHEN OTHERS => F <= "1111";
    END CASE;
  END PROCESS;
END Behavior;

Figure 6.48: Code that represents the functionality of the 74381 ALU

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY seg7 IS
PORT ( w0, w1, s : IN STD_LOGIC;
      f : OUT STD_LOGIC )
END seg7;
ARCHITECTURE Behavior OF seg7 IS
BEGIN
  PROCESS ( w0, w1, s )
  BEGIN
    IF A = B THEN
      AeqB <= '1';
    END IF;
  END PROCESS;
END Behavior;

Figure 6.49: A CASE statement that represents a 2-to-1 multiplexer
Figure 6.49: Timing simulation for the 74381 ALU code

Summary

- Multiplexers
- Decoders
- Encoders
- Code converters
- Arithmetic comparison circuits
- VHDL for combinational circuits