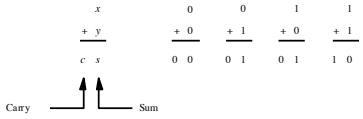
Engr433 - Digital Design

VHDL Examples

Dr Curtis Nelson

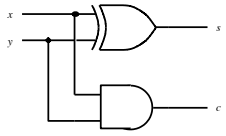
Adder Circuits – Half Adder



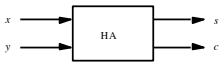
(a) The four possible cases

x	y	Carry c	Sum s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(b) Truth table



(c) Circuit

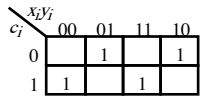


(d) Graphical symbol

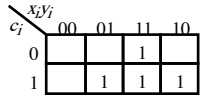
Adder Circuits – Full Adder

c_i	x_i	y_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

(a) Truth table

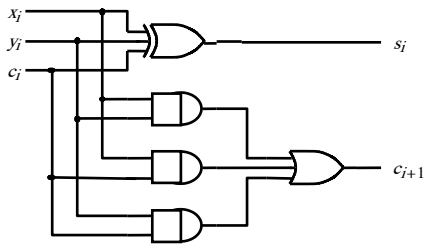


$s_i = x_i \oplus y_i \oplus c_i$



$c_{i+1} = x_iy_i + x_ic_i + y_ic_i$

(b) Karnaugh maps

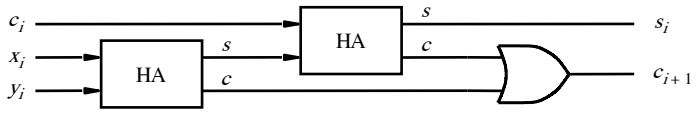


(c) Circuit

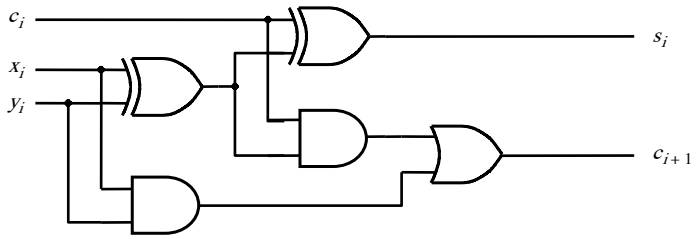
VHDL Code for a Full Adder

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY  fulladd IS
      PORT (cin, a, b          : IN      STD_LOGIC ;
            s, cout            : OUT     STD_LOGIC ) ;
END fulladd ;

ARCHITECTURE Behavior OF fulladd IS
BEGIN
      s <= a XOR b XOR cin;
      cout <= (a AND b) OR (a AND cin) OR (b AND cin);
END Behavior ;
```

Alternative Full Adder



(a) Block diagram



(b) Detailed diagram

Class Assignment

- Write **structural** VHDL code for a 4-bit ripple adder using the full adder as a component.
- Teammates for today only
  - Arlt and Ballance
  - Binder and Birge
  - Christensen and Jessop
  - Jurgensen and Llewellyn
  - Marcondes and Palma
  - Pham and Russell
  - Sukachevin and Terrado
  - Ventura and Wilde

VHDL Code for a 4-bit Ripple Adder

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY  adder4 IS
      PORT (    cin              : IN      STD_LOGIC ;
                x3,x2,x1,x0      : IN      STD_LOGIC ;
                y3,y2,y1,y0      : IN      STD_LOGIC ;
                s3,s2,s1,s0      : IN      STD_LOGIC ;
                x3,x2,x1,x0      : IN      STD_LOGIC ;
                cout             : OUT     STD_LOGIC ) ;
END adder4 ;

ARCHITECTURE Structure OF adder4 IS
      SIGNAL c1,c2,c3 : STD_LOGIC ;
      COMPONENT fulladd
            PORT (    cin, x, y   : IN      STD_LOGIC ;
                      s, cout     : OUT     STD_LOGIC ) ;
      END COMPONENT ;
BEGIN
      stage0: fulladd PORT MAP (cin, x0, y0, s0, c1 );
      stage1: fulladd PORT MAP (c1, x1, y1, s1, c2 );
      stage2: fulladd PORT MAP (c2, x2, y2, s2, c3 );
      stage3: fulladd PORT MAP (
            cin => c3, cout => cout, x=> x3, y => y3, s => s3) ;
END Structure ;
```

VHDL Code for a 2-to-1 Mux

```
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 ;
```

VHDL Using Conditional Signal Assignments

```
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
      f <= w0 WHEN s = '0' ELSE w1 ;
END Behavior ;
```

VHDL Code for a 4-to-1 Mux

```
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 ;
```

Component Declaration for 4-to-1 Mux

```
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 ;
```

Hierarchical Code for a 16-to-1 Mux

```
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 ;
```

Code for a 2-to-4 Decoder

```
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 ;
```

Hierarchical Code for a 4-to-16 Decoder

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY dec4to16 IS
      PORT (    w     : IN      STD_LOGIC_VECTOR(3 DOWNTO 0) ;
                En    : IN      STD_LOGIC ;
                y     : OUT     STD_LOGIC_VECTOR(0 TO 15) ) ;
END dec4to16 ;

ARCHITECTURE Structure OF dec4to16 IS
      COMPONENT dec2to4
            PORT (    w     : IN      STD_LOGIC_VECTOR(1 DOWNTO 0) ;
                      En    : IN      STD_LOGIC ;
                      y     : OUT STD_LOGIC_VECTOR(0 TO 3) ) ;
      END COMPONENT ;
      SIGNAL m : STD_LOGIC_VECTOR(0 TO 3) ;
BEGIN
      G1: FOR i IN 0 TO 3 GENERATE
            Dec_ri: dec2to4 PORT MAP ( w(1 DOWNTO 0), m(i), y(4*i TO 4*i+3) );
            G2: IF i=3 GENERATE
                  Dec_left: dec2to4 PORT MAP ( w(i DOWNTO i-1), En, m ) ;
            END GENERATE ;
      END GENERATE ;
END Structure ;
```

VHDL Code for a Priority Encoder

```
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" ;
      z <= '0' WHEN w = "0000" ELSE '1' ;
END Behavior ;
```

Less Efficient Code for a Priority Encoder

```
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 "0001",
                "01" WHEN "0010",
                "01" WHEN "0011",
                "10" WHEN "0100",
                "10" WHEN "0101",
                "10" WHEN "0110",
                "10" WHEN "0111",
                "11" WHEN OTHERS ;
      WITH w SELECT
            z <='0' WHEN "0000",
                '1' WHEN OTHERS ;
END Behavior ;
```

A BCD-to-7-segment Decoder

```
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 ;
```

Code for a 4-bit Magnitude Comparator

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;

ENTITY compare IS
        PORT (      A, B                    : IN        STD_LOGIC_VECTOR(3 DOWNTO 0) ;
                    AeqB, AgtB, AltB        : OUT       STD_LOGIC ) ;
END compare ;

ARCHITECTURE Behavior OF compare IS
BEGIN
        AeqB <=   '1' WHEN A = B ELSE '0' ;
        AgtB <= '1' WHEN A > B ELSE '0' ;
        AltB <= '1' WHEN A < B ELSE '0' ;
END Behavior ;
```

4-bit Comparator using Signed Numbers

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_arith.all ;

ENTITY compare IS
      PORT (   A, B                  : IN     SIGNED(3 DOWNTO 0) ;
               AeqB, AgtB, AltB      : OUT    STD_LOGIC ) ;
END compare ;

ARCHITECTURE Behavior OF compare IS
BEGIN
      AeqB <= '1' WHEN A = B ELSE '0' ;
      AgtB <= '1' WHEN A > B ELSE '0' ;
      AltB <= '1' WHEN A < B ELSE '0' ;
END Behavior ;
```

Code that Represents the Functionality of the 74381 ALU

```
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 ;
```

Concurrent vs. Sequential VHDL Code

- All previous VHDL statements shown 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 one has an effect.

2-to-1 Mux using an if-then-else Statement

```
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 ;
```

Alternative Code for a 2-to-1 Mux

```
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 ;
```

Priority Encoder Using if-then-else

```
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 <= '0' WHEN w = "0000" ELSE '1' ;
END Behavior ;
```

Alternative Code for the Priority Encoder

```
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
        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" ; END IF ;

        z <= '1' ;
        IF w = "0000" THEN z <= '0' ; END IF ;
    END PROCESS ;
END Behavior ;
```

Code for a 1-bit Equality Comparator

```
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 ;
```

CASE statement for a 2-to-1 Mux

```
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 ;
```

Case Statement for a 2-to-4 Binary Decoder

```
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
BEGIN
        PROCESS ( w, En )
        BEGIN
                IF En = '1' THEN
                        CASE w IS
                                WHEN "00" =>            y <= "1000" ;
                                WHEN "01" =>            y <= "0100" ;
                                WHEN "10" =>            y <= "0010" ;
                                WHEN OTHERS =>          y <= "0001" ;
                        END CASE ;
                ELSE
                        y <= "0000" ;
                END IF ;
        END PROCESS ;
END Behavior ;
```

Implied Memory

```
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 ;
```

Code for a Gated D Latch

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY latch IS
      PORT (   D, Clk     : IN        STD_LOGIC ;
               Q          : OUT     STD_LOGIC) ;
END latch ;

ARCHITECTURE Behavior OF latch IS
BEGIN
      PROCESS ( D, Clk )
      BEGIN
            IF Clk = '1' THEN
                  Q <= D ;
            END IF ;
      END PROCESS ;
END Behavior ;
```

Code for a D Flip-flop

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
      PORT (   D, Clock  : IN      STD_LOGIC ;
               Q          : OUT   STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
      PROCESS ( Clock )
      BEGIN
           IF Clock'EVENT AND Clock = '1' THEN
                  Q <= D ;
           END IF ;
      END PROCESS ;
END Behavior ;
```

Code for a D Flip-flop Using WAIT UNTIL

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY flipflop IS
      PORT (   D, Clock  : IN       STD_LOGIC ;
               Q         : OUT    STD_LOGIC ) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
     PROCESS
     BEGIN
          WAIT UNTIL Clock'EVENT AND Clock = '1' ;
          Q <= D ;
     END PROCESS ;
END Behavior ;
```

D Flip-flop with Asynchronous Reset

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
     PORT (   D, Resetn, Clock    : IN       STD_LOGIC ;
              Q                   : OUT    STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
     PROCESS ( Resetn, Clock )
     BEGIN
          IF Resetn = '0' THEN
               Q <= '0' ;
          ELSIF Clock'EVENT AND Clock = '1' THEN
               Q <= D ;
          END IF ;
     END PROCESS ;
END Behavior ;
```

D Flip-flop with Synchronous Reset

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
      PORT (   D, Resetn, Clock     : IN        STD_LOGIC ;
               Q                    : OUT       STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
      PROCESS
      BEGIN
            WAIT UNTIL Clock'EVENT AND Clock = '1' ;
            IF Resetn = '0' THEN
                  Q <= '0' ;
            ELSE
                  Q <= D ;
            END IF ;
      END PROCESS ;
END Behavior ;
```