Advanced Examples in VHDL for Max+plus II

Daniel Amyot

damyot@site.uottawa.ca

Full Adder (*fulladd*)

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

ENTITY fulladd IS
    PORT (  Cin, x, y    : IN     STD_LOGIC ;
            s, Cout      : OUT    STD_LOGIC ) ;
END fulladd ;

ARCHITECTURE LogicFunc OF fulladd IS
BEGIN
    s <= x XOR y XOR Cin ;
    Cout <= (x AND y) OR (x AND Cin) OR (y AND Cin) ;
END LogicFunc ;
```

4 bits Adder (using *fulladd*)

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

ENTITY adder4 IS
    PORT (  Cin      : IN    STD_LOGIC ;
            X, Y     : IN    STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            S        : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            Cout     : OUT   STD_LOGIC ) ;
END adder4 ;

ARCHITECTURE Structure OF adder4 IS
    SIGNAL C : STD_LOGIC_VECTOR(1 TO 3) ;
    COMPONENT fulladd
        PORT (  Cin, x, y    : IN    STD_LOGIC ;
                s, Cout      : OUT   STD_LOGIC) ;
    END COMPONENT ;
BEGIN
    stage0: fulladd PORT MAP ( Cin , X(0), Y(0), S(0), C(1) ) ;
    stage1: fulladd PORT MAP ( C(1), X(1), Y(1), S(1), C(2) ) ;
    stage2: fulladd PORT MAP ( C(2), X(2), Y(2), S(2), C(3) ) ;
    stage3: fulladd PORT MAP (
        x => X(3), y => Y(3), Cin => C(3), s => S(3), Cout => Cout ) ;
END Structure ;
```

Remarks

- Vector of bits
  - `STD_LOGIC_VECTOR(3 DOWNTO 0)`
  - `X(0)`
- Intermediate signal
  - `SIGNAL C : STD_LOGIC_VECTOR(1 TO 3) ;`
- Connecting ports
  - `fulladd PORT MAP (Cin,X(0),Y(0),S(0),C(1))`
  - `fulladd PORT MAP (x => X(3), y => Y(3), Cin => C(3), s => S(3), Cout => Cout )`
- Stages

Logic and Arithmetic Operators

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

ENTITY adder IS
    PORT (  Cin      : IN    STD_LOGIC ;
            X, Y     : IN    STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            S        : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            Cout     : OUT   STD_LOGIC ) ;
END adder ;

ARCHITECTURE Behavior OF adder IS
    SIGNAL Sum : STD_LOGIC_VECTOR(4 DOWNTO 0) ;
BEGIN
    Sum <= ('0' & X) + Y + Cin ;
    S <= Sum(3 DOWNTO 0) ;
    Cout <= Sum(4) ;
END Behavior ;
```

Generic entity, process, loops

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

ENTITY NANDn IS
    GENERIC ( n : INTEGER := 4 ) ;
    PORT (  X   : IN    STD_LOGIC_VECTOR(1 TO n) ;
            f   : OUT   STD_LOGIC ) ;
END NANDn ;

ARCHITECTURE Behavior OF NANDn IS
    SIGNAL Tmp : STD_LOGIC ;
BEGIN
    PROCESS ( X )
    BEGIN
        Tmp <= X(1) ;
        AND_bits: FOR i IN 2 TO n LOOP
            Tmp <= Tmp AND X(i) ;
        END LOOP AND_bits ;
        f <= NOT Tmp ;
    END PROCESS ;
END Behavior ;
```

Utilization of generic entities

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

ENTITY adderlpm IS
    PORT (  Cin      : IN    STD_LOGIC ;
            A, B     : IN    STD_LOGIC_VECTOR(15 DOWNTO 0) ;
            Sum      : OUT   STD_LOGIC_VECTOR(15 DOWNTO 0) ;
            Cout     : OUT   STD_LOGIC ) ;
END adderlpm ;

ARCHITECTURE Structure OF adderlpm IS
BEGIN
    instance: lpm_add_sub
        GENERIC MAP (LPM_WIDTH => 16)
        PORT MAP (  cin => Cin, dataa => A, datab => B,
            result => Sum, cout => Cout ) ;
END Structure ;
```

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 ;
```

Selection

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

ENTITY problem IS
    PORT (  Input    : IN    STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            Output   : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0) ) ;
END problem ;

ARCHITECTURE LogicFunc OF problem IS
BEGIN
    WITH Input SELECT
        Output <=   "0001" WHEN "0101",
                    "0010" WHEN "0110",
                    "0011" WHEN "0111",
                    "0010" WHEN "1001",
                    "0100" WHEN "1010",
                    "0110" WHEN "1011",
                    "0011" WHEN "1101",
                    "0110" WHEN "1110",
                    "1001" WHEN "1111",
                    "0000" WHEN OTHERS ;
END LogicFunc ;
```

Maxplus2 Library

- C:\maxplus2\vhdl93\altera\maxplus2.vhd
  - a_21mux -- 2 to 1 Mux (1 bit)
  - a_2x8mux – 2 to 1 Mux (8 bits)
  - a_16dmux -- 16 to 1 Mux
  - mux (nb_data, nb_sel) – variable size Mux
    - *Library of parameterized module* (LPM)
    - nb_data is the number of input lines
    - nb_sel is the number of selection lines
  - Other members of 74xx family
    - Ex: a_7400 (NAND2), a_7410 (NAND3), …
    - Adders, decoders, comparators, multiplexers, …
  - D, JK, SR, T flip-flop