LECTURE 7: VHDL Standard Cell Library Package
Standard Cell Library

- **Standard Cell simulation libraries**: describe components functionally as well as with timing information.

- **Software** programmers are primarily concerned with functionality (i.e. given the data inputs (mouse, keyboard, etc.) does it give the expected output results?)

- **Software** programmers sometimes spend additional time to make the program work relatively faster than before. (i.e. time complexity $O(n)$ versus $O(n^2)$)

- Hardware designers deal with real-time issues all the time.

- **Functional hardware specifications** include timing requirements. This translates to an output signal must arrive within a certain time period.)
Standard Cell Library

- **Standard Cell libraries**
  - Standard cell libraries do not contain every possible logic gate. For example, AND gates may not be included.
  - CMOS process technology favors NAND and NOR gates over AND and OR gates in number of transistors.
  - High level Synthesis tools will build the AND gate from a NAND and an INVERTOR gate.
  - By using VHDL, the design becomes portable between cell libraries.
Standard Cell Library

- **Standard Cell library**
  - Assume the library contains the following 6 components
    - nandf201: 2 input nand with 1x output drive
    - norf201: 2 input nor with 1x output drive
    - invf101: 1 input not gate with 1x output drive
    - xorf201: 2 input xor gate with 1x output drive
    - xnof201: 2 input xnor gate with 1x output drive
    - dfbf311: D-Flip Flop with D, Reset, Set, Q, QN, Clk
  - Next we want to create a package library to reference it.
LIBRARY IEEE;
USE IEEE.STD_LOGIC_VECTOR;

ENTITY nandf201 IS
  GENERIC(Tdelay: TIME:=10 ns);
  PORT(A1, B1: IN std_logic;
       F1: OUT std_logic);
END ;

ARCHITECTURE nandf201_arch OF nandf201 IS
BEGIN
  F1 <= ( A1 NAND B1 ) after Tdelay;
END;

CONFIGURATION nandf201_cfg OF nandf201 IS
  FOR nandf201_arch
  END FOR;
END;
LIBRARY IEEE;
USE IEEE.STD_LOGIC_VECTOR.ALL;

PACKAGE STDLIB1 IS
  COMPONENT nandf201 IS
    GENERIC(Tdelay: TIME:=5 ns);
    PORT(A1, B1: IN std_logic; F1: OUT std_logic);
  END COMPONENT;

  COMPONENT norf201 IS
    GENERIC(Tdelay: TIME:=5 ns);
    PORT(A1, B1: IN std_logic; F1: OUT std_logic);
  END COMPONENT;

  ...

END;

PACKAGE BODY STDLIB1 IS
END;
**Full Adder: Architecture**

```vhdl
LIBRARY IEEE; USE IEEE.STD_LOGIC_VECTOR.ALL;
LIBRARY WORK; USE WORK.STDLIB1.ALL;

ENTITY full_adder IS
  PORT(x, y, z: IN std_logic;
       Sum, Carry: OUT std_logic);
END;

ARCHITECTURE full_adder_arch_1 OF full_adder IS
  SIGNAL xy: std_logic;
  BEGIN
    -- Note: “Sum <= ( ( x XOR y ) XOR z );”
    xor1: xor201 port map(x, y, xy);
    xor2: xor201 port map(xy, z, Sum);
  END;
```

```vhdl
LIBRARY IEEE; USE IEEE.STD_LOGIC_VECTOR.ALL;
LIBRARY WORK; USE WORK.STDLIB1.ALL;

ENTITY full_adder IS
  PORT(x, y, z: IN std_logic;
       Sum, Carry: OUT std_logic);
END;

ARCHITECTURE full_adder_arch_1 OF full_adder IS
  SIGNAL xy: std_logic;
  BEGIN
    -- Note: “Sum <= ( ( x XOR y ) XOR z );”
    xor1: xor201 port map(x, y, xy);
    xor2: xor201 port map(xy, z, Sum);
  END;
```
LIBRARY IEEE;
USE IEEE.std_logic_vector.all;

LIBRARY WORK;
USE WORK.stdlib1.all;

ENTITY nandf201_testbench IS
END;

ARCHITECTURE nandf201_testbench_arch OF nandf201_testbench IS
SIGNAL A, B, F: std_logic;
BEGIN
  tb1: nandf201 port map(F, A, B);
  -- reminder: absolute time for wave statement
  A <= '0', '1' after 10 ns, '0' after 20 ns, '1' after 30 ns;
  B <= '0', '0' after 10 ns, '1' after 20 ns, '1' after 30 ns;
END;
USE STD.textio.all; --write/read strings

LIBRARY IEEE;
USE IEEE.std_logic_vector.all;
USE IEEE.std_logic_textio.all; --write/read std_logic

LIBRARY WORK;
USE WORK.stdlib1.all; --component libs

ENTITY nandf201_testbench IS
END ;
ARCHITECTURE nandf201_testbench_arch OF nandf201_testbench IS
SIGNAL A, B, F: std_logic;
BEGIN
  tb1: nandf201 port map(F, A, B);
  -- reminder: absolute time for wave statement
  A <= '0', '1' after 10 ns, '0' after 20 ns, '1' after 30 ns;
  B <= '0', '0' after 10 ns, '1' after 20 ns, '1' after 30 ns;
  PROCESS(A, B, F) -- listen to any change in signal
    variable buf: line; -- pointer to a string
    BEGIN
      write(buf, string'("Time= "));
      write(buf, NOW);
      write(buf, string'(" A= "));
      write(buf, A);
      write(buf, string'(" B= "));
      write(buf, B);
      writeln(output, buf);
    END;
  END PROCESS;
END;
Nandf201: sample output text io test bench

• PROCESS(A, B, F) means that every time there is a change in signals A, B, of F: this process will be called
• This is called a sensitivity list because it is listening in on all signals declared in the list

vhdlan -NOEVENT stdlib1.vhd
vhdlan -NOEVENT nandf201_testbench.vhd
vhdlsim nandf201_testbench_cfg
#run
  Time=0 NS F=U A=U B=U
  Time=0 NS F=U A=0 B=0
  Time=10 NS F=1 A=1 B=0
  Time=20 NS F=1 A=0 B=1
  Time=30 NS F=1 A=1 B=1
  Time=40 NS F=0 A=1 B=1
Nandf201: process wave function

The absolute time wave function:
A <= ‘0’, ‘1’ after 10 ns, ‘0’ after 20 ns;
B <= ‘0’, ‘0’ after 10 ns, ‘1’ after 20 ns;

Can be re-written using relative time:
PROCESS
    VARIABLE buf: LINE;
    BEGIN
        A <= ‘0’; B<=‘0’;
        write(buf, string’("Time= ")); write(buf, NOW);
        write(buf, string’(" A=")); write(buf, A);
        write(buf, string’(" B=")); write(buf, B);
        writeln(output, buf);
        wait for 10 ns;
        A <= ‘1’; B<=‘0’; --followed by more write statements
        wait for 10 ns;
        A <= ‘0’; B<=‘1’; --followed by more write statements
        wait for 10 ns;
    END PROCESS;
PROCESS
  VARIABLE Test: TIME:=10 ns;
  VARIABLE Buf: LINE;
BEGIN
  A <= ‘0’; B<=‘0’;
  Write(Buf, string(“Time= “)); Write(Buf, NOW);
  Write(Buf, string(“ A=“)); Write(Buf, A);
  Write(Buf, string(“ B=“)); Write(Buf, B);
  WriteLine(Output, Buf);
  Wait for Test;

  A <= ‘1’; B<=‘0’; --followed by more write statements
  Wait for Test;

  A <= ‘0’; B<=‘1’; --followed by more write statements
  Wait for Test;
END PROCESS;
ARCHITECTURE nandf201_testbench_arch OF nandf201_testbench IS
  SIGNAL A, B, F: std_logic;
BEGIN
  tb1: nandf201 port map(F, A, B);
  PROCESS
  BEGIN
    A <= '0'; B<=‘0’;
    write(buf, string’("Time= ")); write(buf, NOW);
    write(buf, string’(" A=")); write(buf, A);
    write(buf, string’(" B=")); write(buf, B);
    writeln(output, buf);
    wait for 10 ns;
    if F/='0' then --not equal to
      write(buf, string’("test #1 failed"));
      writeln(output, buf);
    end if;
    ...
    wait; --always add a final wait for test benches--
  END PROCESS;
END nandf201_testbench_arch;
Assignment #7

a) Write a package containing all of the six stdlib1 gates described earlier and rewrite your 1-bit ALU using only these components. The stdlib1 package should also contain 1-bit ALU and the generic N-bit ALU.

b) Write a test bench for the N-bit ALU (i.e. N=8) that uses your package and checks your output values with if/then statements. Do forget to construct the worst case test vectors for 8-bit adds.

c) Using a Tdelay for each gate of 10 ns, determine the minimum delay of your adder by varying the test bench time until it passes all the tests constructed.

d) Hand in the source files and vhdsim session using the Unix script command.