ECE 4305 Computer Architecture
Lab #3: VHDL Library
Prepared by Prof. T. Kwon

The objective of this lab is to learn to utilize the library function in VHDL i.e. how to create and use it. Library is a convenient storage for frequently used components and helps reduce the design complexity. It is also an essential tool for a team design in which each member designs a particular set of components and later combine them as the final design.

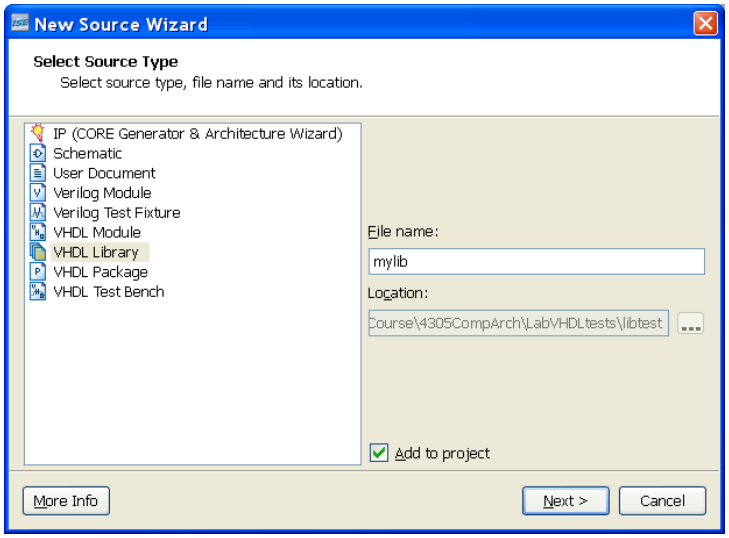
Start this lab by following **Part 1** which is a step-by-step tutorial on how to build and use a library in VHDL, and then finish the **Part 2** on your own.

Part 1: Library implementation

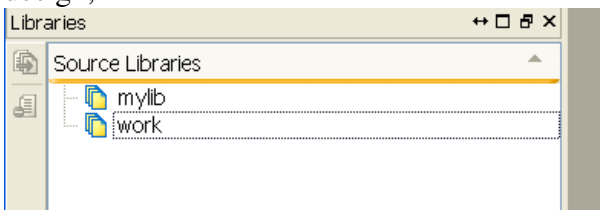
This part is a step-by-step instruction on how to create a new library, add parts (a half adder and an OR gate in this example), and the use them. A full adder is used as an example.

Step 1: Select **File > New Project …** and create a new project with a project name such as "**libtest**".
For the **Top-Level Source Type**, select "**HDL**".
Device properties are the same as the last labs.
Click **Finish** at the Project Summary to complete this step.

Step 2: Select "**New Source**…" by right clicking on the chip name. This time, select **VHDL Library.** In the **File name** textbox, type in "mylib" as shown below. This would be your VHDL library name, and you may choose a different name.



Click **Next** and then click **Finish** to complete the **New Source Wizard**. Now you have created your own VHDL library. If you click on the **Libraries** tab, you should see the "mylib" library you just created. The "work" library is the default library of your current design,



Next, you will add a half_adder to the created library.

Step 3: Go back to the **Design** tab. Create a half adder using the **New Source Wizard**. Select the VHDL module and enter the circuit name "myHA" Specify the port, and then enter the source as:

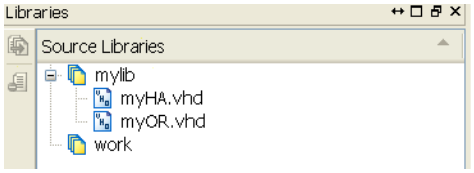
```
entity myHA is
    Port ( x : in  STD_LOGIC;
           y : in  STD_LOGIC;
           sum : out  STD_LOGIC;
           carry : out  STD_LOGIC);
end myHA;

architecture Behavioral of myHA is

begin
 sum <= x xor y;
 carry <= x and y;
end Behavioral;
```

Expand **Synthesize-SXT** and double click on **Check Syntax** to make sure that the "myHA" code does not have any syntax error.

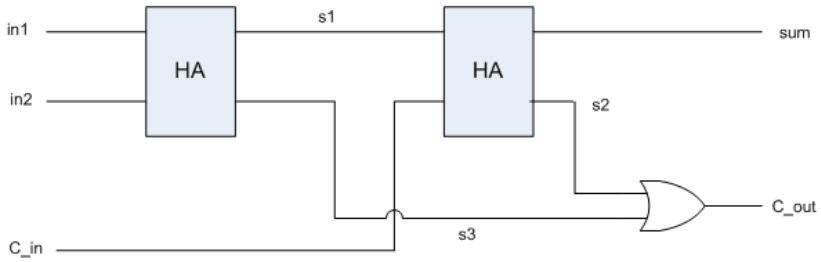
Step 4: Using the **New Source Wizard,** create a VHDL source file for a single OR gate by repeating Step 3. Name this file **myOR.**

```
entity myOR is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           y : out  STD_LOGIC);
end myOR;

architecture Behavioral of myOR is

begin
  y <= a or b;
end Behavioral;
```

Step 5: The myHA and myOr modules are in the work library as a default. Select the **Libraries** tab and add the "myHA.vhd" and "myOR.vhd" files to **mylib**, using right click on the items in the work library and selecting "**Move to Library …**" item. The result should look like:



Step 6: Add a new VHDL source for the final full adder by following below diagram. Make this module (FA) as the top module.



Complete the full adder source code. Two styles codes are given.
Style-1)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
library mylib;
use mylib.ALL;

entity myFA is
    Port ( in1 : in  STD_LOGIC;
           in2 : in  STD_LOGIC;
           c_in : in  STD_LOGIC;
           sum : out  STD_LOGIC;
           c_out : out  STD_LOGIC);
end myFA;

architecture Behavioral of myFA is
  signal s1,s2,s3: std_logic;

begin
  HA1: entity mylib.myHA(Behavioral)
      port map (x=>in1, y=>in2, sum=>s1, carry=>s3);
  HA2: entity mylib.myHA(Behavioral)
      port map (x=>s1, y=>c_in, sum=>sum, carry=>s2);

  OR1: entity mylib.myOR(Behavioral)
      port map (a=>s2, b=> s3, y=>c_out);
end Behavioral;
```

Note that mylib is declared at the beginning of the code. The component from the mylib is defined through entity.

Style-2)
There is another way of writing the code. You can include the component definitions in the architecture to clearly see the ports. This example is given next. Please note the syntax difference in the port map.

```
entity myFA is
    Port ( in1 : in  STD_LOGIC;
           in2 : in  STD_LOGIC;
           c_in : in  STD_LOGIC;
           sum : out  STD_LOGIC;
           c_out : out  STD_LOGIC);
end myFA;

architecture Behavioral of myFA is
  signal s1,s2,s3: std_logic;

  component myHA is
    port (x, y : in std_logic;
          sum, carry : out std_logic);
  end component myHA;

  component myOR is
    port (a, b : in std_logic;
          y : out std_logic);
  end component myOR;

begin
  HA1: myHA
      port map (x=>in1, y=>in2, sum=>s1, carry=>s3);
  HA2: myHA
      port map (x=>s1, y=>c_in, sum=>sum, carry=>s2);

  OR1: myOR
      port map (a=>s2, b=> s3, y=>c_out);
end Behavioral;
```

Note that entity is no longer defined on the port map. This approach is more readable because you can see the ports of the components. If you are using this approach, make sure that you use the same port names and positions.

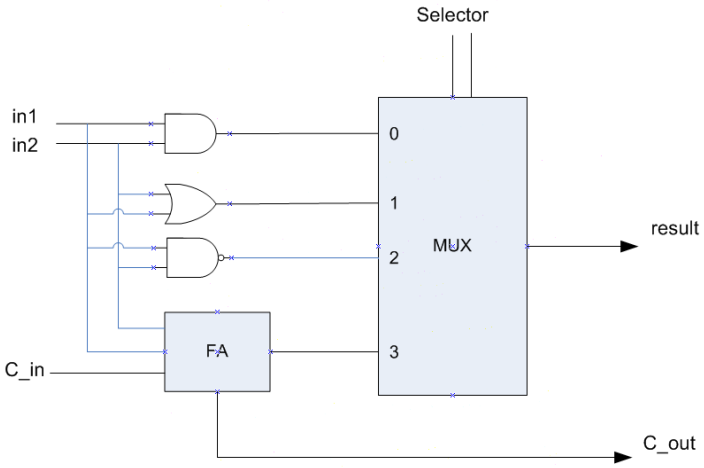
Connect I/O pins as,

C_in (SW2) → K18
C_out (LED 1) →J15
In1(SW 0) → G18
In2 (SW 1) → H18
Sum (LED 0) → J14

Compile to produce the bit file and test whether it is correctly working or not.

Part 2: ALU implementation using your library components

Using the techniques you learned from Part 1, implement a single bit ALU shown below using VHDL codes and components in your library. This is the same ALU used in Lab 2.



You library should include orgate, andgate, nandgate, halfadder, fulladder, and mux. The Mux code is given below.

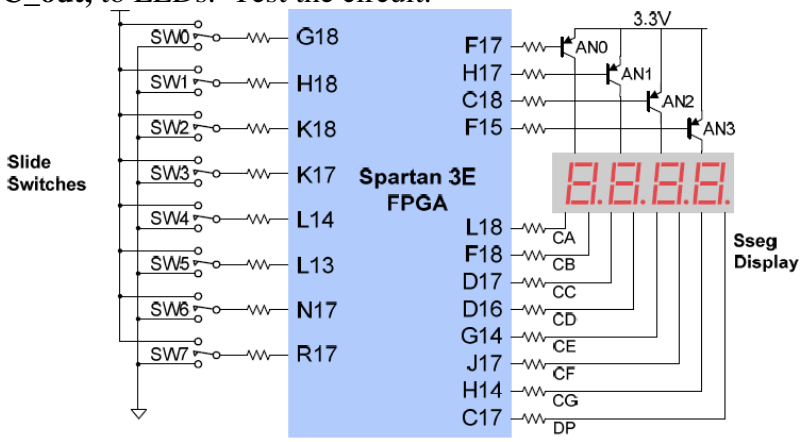
```
entity MUX_4_1 is
   port ( SEL: in std_logic_vector(1 downto 0);
            A, B, C, D: in std_logic;
            Z: out std_logic);
   end MUX_4_1;
architecture Behav of MUX_4_1 is
begin
   process (SEL, A, B, C, D)
   begin
      case SEL is
         when (SEL = "00") => Z <= A;
         when (SEL = "01") => Z <= B;
         when (SEL = "10") => Z <= C;
         when others => Z <= D;
      end case;
   end process;
end Behav;
```

Another way of writing Mux:

```
   process (SEL, A, B, C, D)
   begin
      if (SEL = "00") then Z<= A;
      elsif (SEL = "01") then Z<= B;
      elsif (SEL = "10") then Z<= C;
      else Z<=D;
      end if;
   end process;
```

Write your ALU code using **port map** as shown in the full adder example.

Connect the inputs **in1**, **in2**, **C_in**, and **Selector** to switches and the outputs, **result** and **C_out,** to LEDs. Test the circuit.



Check-off

Show your library along with your **Part 2** implementation.

Report

- What was the purpose of this lab?
- Discuss the advantages and disadvantages of using libraries.
- How would you design your library for large circuits?
- Which approach would be a better, schematic or VHDL coding if you are making a large circuit?
- Attach a screen capture of your Libraries tab.
- What you learned in this lab and your own conclusion
- Attach your VHDL source code for the final ALU implementation.

- **Due date** for the report is the following Monday of the lab week. **Late reports** are not accepted.