CSCE 3730 – Reconfigurable Logic
Lab 5 – 11/13/2008 (Thursday)

Design a one-bit ALU using all the components like AND-gate, OR-gate, NOT-gate, 2 to 1 Mux, 4 to 1 Mux and adder that you have designed in Lab 4. Write the VHDL code for one-bit ALU instantiating all the modules using COMPONENT DECLARATION and PORT MAPING. And also design another one-bit ALU – the MSB bit ALU which has an extra logic when compared to the normal one-bit ALU. Please refer the ALU Descriptions slides for further understanding. Ask T.A. for any help.

Lab 5 - Example to use Components and Port Mapping.

This example shows you, how to design a simple ALU which does just two operations.

1. And operation
2. Or Operation.

So we have two modules And-gate and Or-gate, and we need one 2 to 1 Mux to select the outputs of And_ gate and Or-gate .

Code for And-gate

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity andgate is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           f : out  STD_LOGIC);
end andgate;

architecture Behavioral of andgate is

begin
f <= a and b;

end Behavioral;
```

Code for Or-gate

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity orgate is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          f : out  STD_LOGIC);
end orgate;

architecture Behavioral of orgate is

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

Code for 2 to 1 MUX

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity mux2to1 is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          s : in  STD_LOGIC;
          f : out  STD_LOGIC);
end mux2to1;

architecture Behavioral of mux2to1 is

begin

f<= (a and (not s)) or ( b and s);
end Behavioral;
```

Code for 1 bit ALU which does just And & Or operation

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity alu1bit is
   Port ( a_1bit : in  STD_LOGIC;
          b_1bit : in  STD_LOGIC;
          s_1bit : in  STD_LOGIC;
          f_1bit : out  STD_LOGIC);
end alu1bit;

architecture Behavioral of alu1bit is

------ component Declaration Part --
-- same as the entitiy of that module just the component keyword is used in palce of entity ---

component andgate is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          f : out  STD_LOGIC);
end component andgate;

component orgate is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          f : out  STD_LOGIC);
end component orgate;

component  mux2to1 is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          s : in  STD_LOGIC;
          f : out  STD_LOGIC);
end component mux2to1;

---- internal signals or wires to connect these components ---

signal out_and: std_logic;
```

```
signal out_or: std_logic;

begin

-- portmaping the components --

and_gate: andgate
port map (

a => a_1bit,
b => b_1bit,
f => out_and);


or_gate : orgate
port map (
a => a_1bit,
b => b_1bit,
f => out_or);

mux2 : mux2to1

port map(

a => out_and,
b => out_or,
s => s_1bit,
f => f_1bit);

end Behavioral;
```

So this example shows how to use Components and port Mapping. Internal signals are used to connect the components internally.