Computer Assignment 2

ECGR 2181 - Fall 2009

Assignment Overview

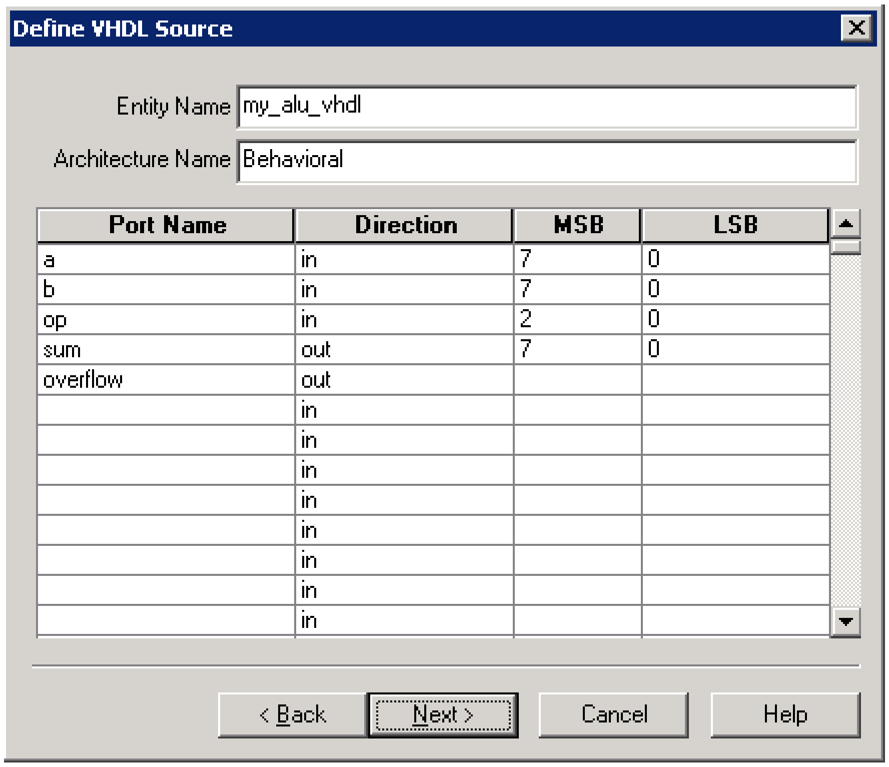
In this assignment you will use Xilinx ISE 7.1i and ModelSim to build and simulate an Arithmetic Logic Unit created from schematics and VHDL. The assignment is based on Homework 8. Refer to Homework 8 for the operations the ALU must be able to perform. Demo the final simulation to one of the TAs during their Office Hours (in Woodward 200). **You are allowed to work in groups of 2 as long as you are both in the same section.** Make sure you both understand everything, when you demo the final solution the TA will ask you both questions regarding the system to make sure you both did the work.

In addition you will need to upload a **single PDF** of the two schematics and two VHDL along with the Grade Rubric. Print each page to a PDF file and use Adobe Acrobat to combine the four pages to a single PDF. Name the PDF <lastname_lastname_cad2>.pdf (so schmidt_conrad_cad2.pdf). Upload the PDF to Moodle under Computer Assignment 2.

The Computer Assignment is due Wednesday, November 4th by 3 PM.

Getting Started

1. Create a new directory to store all of the Computer Assignment Files
2. Open **Xilinx ISE 7.1i** and Create a New Project called: **computer_assignment_2**
3. Create it for the **Nexys2 Board** (Spartan 3E, xc3s500e, fg320, -4)
4. Add 1 New Source File called of type VHDL: **my_alu_vhdl**
   a. **This will be the top level design for your Arithmetic Logic Unit. At the end we will describe what code is necessary in this file. For now simply set the Input and Output ports to the following:**



Entity Name: my_alu_vhdl

Architecture Name: Behavioral

Port Name	Direction	MSB	LSB
a	in	7	0
b	in	7	0
op	in	2	0
sum	out	7	0
overflow	out		
	in		
	in		
	in		
	in		
	in		
	in		
	in		
	in		

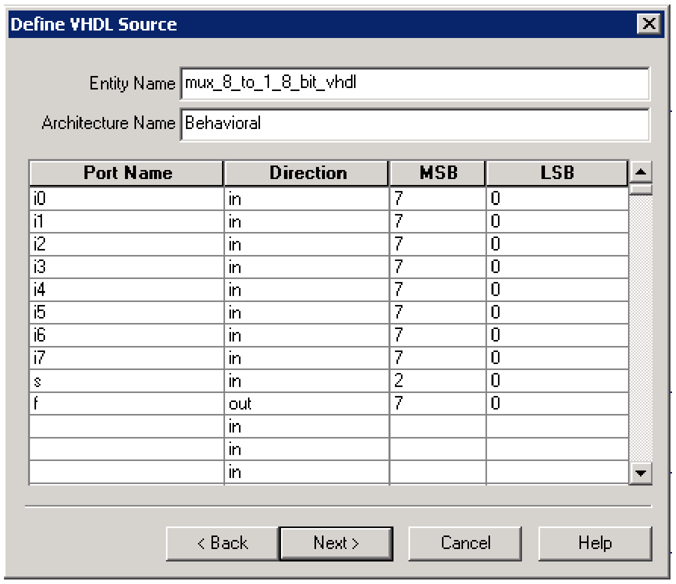
5. Click Next / Finish to complete the New Project Wizard

Full Adder

1. **1-Bit Full Adder**
    a. Add a New Source of type Schematic to the project: **full_adder_1_bit_sch**
    b. Create a schematic of a 1-bit full adder from gates. Look at the Chapter 4 notes if you are not sure how to get started.
    c. Inputs: A, B, Cin
    d. Outputs: S, Cout
    e. Save the schematic and under the Process View for the full_adder_1_bit_sch expand Design Utilities to select and run **Create Schematic Symbol**. This creates the schematic symbol that you will need for part 2 below.
2. **8-Bit Full Adder (full_adder_8_bit_sch)**
    a. Add a New Source of type Schematic to the project: **full_adder_8_bit_sch**
    b. Using the full_adder_1_bit_sch created above create an 8-bit cascaded Full Adder.
    c. In addition to the Full Adder circuit, add the logic needed for Overflow calculation. Look at the Chapter 4 notes if you are not sure how to calculate the Overflow.
    d. Inputs: A(7:0), B(7:0), Cin
    e. Outputs: S(7:0), Cout, Overflow
    f. Save the schematic. It is recommended (although not required) to simulate the 8-bit Full Adder at this point. Good engineering practice is to test components as they are built making the debugging easier as the design becomes more complex.

Multiplexer with VHDL

1. **8-bit 8 to 1 Multiplexer**
    a. Add a New Source of type VHDL Module to the project: **mux_8_to_1_8_bit_vhdl**
    b. Set the Input and Output ports to the following:

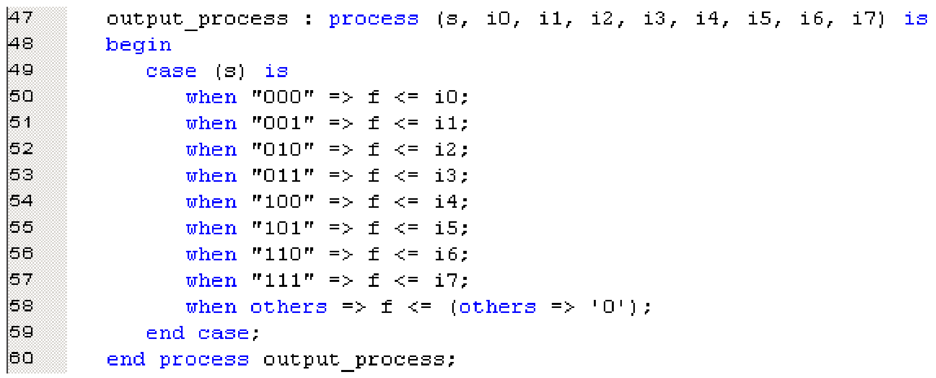


Entity Name: mux_8_to_1_8_bit_vhdl
Architecture Name: Behavioral

Port Name	Direction	MSB	LSB
i0	in	7	0
i1	in	7	0
i2	in	7	0
i3	in	7	0
i4	in	7	0
i5	in	7	0
i6	in	7	0
i7	in	7	0
s	in	2	0
f	out	7	0
	in		
	in		
	in		

    c. Select Finish.
    d. This opens a new VHDL template file which already has some of the code we need.

e. At Line 47 (between "**begin**" and "**end Behavioral;**") create a VHDL Process. This process will set the output "**f**" to the input (i0, i1, i2… i7) based on the value of "**s**". A VHDL process is similar to a function in C++, except when the inputs change the output will automatically change as well.



```
47   output_process : process (s, i0, i1, i2, i3, i4, i5, i6, i7) is
48   begin
49      case (s) is
50         when "000" => f <= i0;
51         when "001" => f <= i1;
52         when "010" => f <= i2;
53         when "011" => f <= i3;
54         when "100" => f <= i4;
55         when "101" => f <= i5;
56         when "110" => f <= i6;
57         when "111" => f <= i7;
58         when others => f <= (others => '0');
59      end case;
60   end process output_process;
```

f. The process is named: **output_process**

g. The Sensitivity List (the inputs to the process) are: **s, i0, i1, i2, i3, i4, i5, i6, i7**

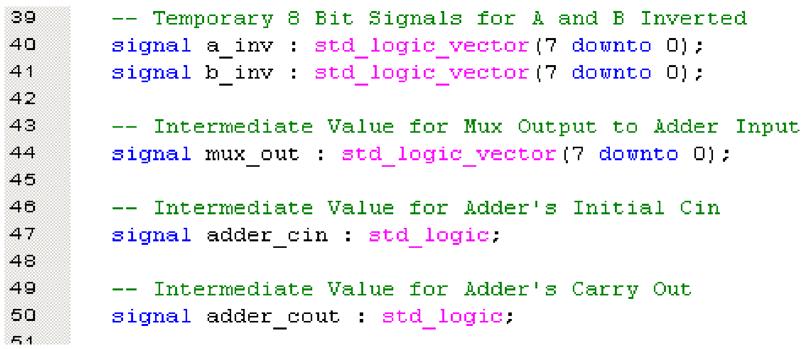
h. The output **"f"** is set to the corresponding input.

    i. For example when s = "011" f is assigned the value of i3.

2. Save the VHDL file and in the Process View select "**Synthesis – XST".** If there are any syntax errors, fix them and re-synthesize until a green check mark appears.
3. It is recommended (although not required) to simulate the 8-bit Mux at this point.

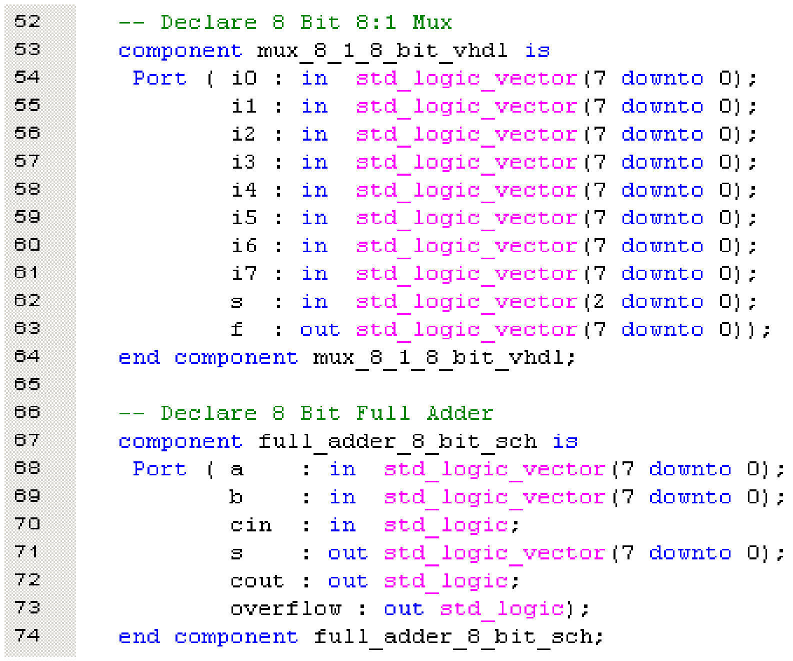
My ALU

1. Open **my_alu_vhdl** file in Xilinx ISE 7.1i (it was the first file you created when you started this project). You will now use VHDL to connect the Multiplexer and Adder together along with the correct Carry-In logic you designed from Homework 8 to complete the ALU. You will also need to look at Homework 8 to see which Inputs are needed for the 8 to 1 Multiplexer. We will provide you with a nearly complete template; you need to connect the correct signals together for the fully functional ALU circuit.
2. At line 39 add the following:



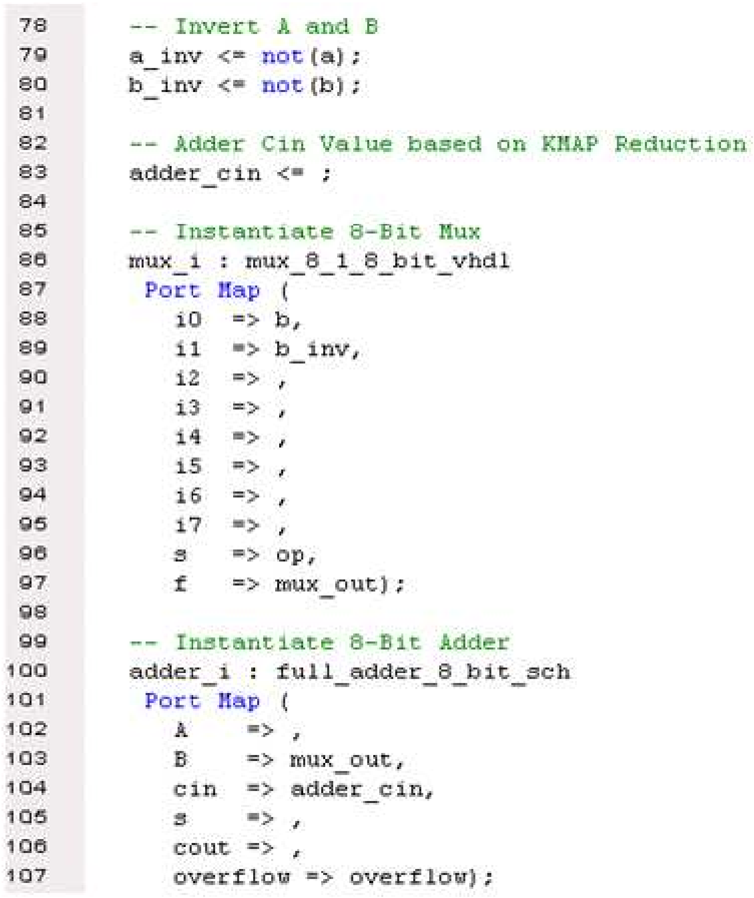
```
39   -- Temporary 8 Bit Signals for A and B Inverted
40   signal a_inv : std_logic_vector(7 downto 0);
41   signal b_inv : std_logic_vector(7 downto 0);
42
43   -- Intermediate Value for Mux Output to Adder Input
44   signal mux_out : std_logic_vector(7 downto 0);
45
46   -- Intermediate Value for Adder's Initial Cin
47   signal adder_cin : std_logic;
48
49   -- Intermediate Value for Adder's Carry Out
50   signal adder_cout : std_logic;
51
```

3. These signals are like "wires" and "buses" in the schematics you have designed. Instead of connecting components together with the wires, we will use VHDL signals (similar to variables in C++). You will start to see the simplicity of using a Hardware Description Language to quickly and concisely create complex systems.
4. Next at line 52 add the following:



```
-- Declare 8 Bit 8:1 Mux
component mux_8_1_8_bit_vhdl is
 Port ( i0 : in  std_logic_vector(7 downto 0);
        i1 : in  std_logic_vector(7 downto 0);
        i2 : in  std_logic_vector(7 downto 0);
        i3 : in  std_logic_vector(7 downto 0);
        i4 : in  std_logic_vector(7 downto 0);
        i5 : in  std_logic_vector(7 downto 0);
        i6 : in  std_logic_vector(7 downto 0);
        i7 : in  std_logic_vector(7 downto 0);
        s  : in  std_logic_vector(2 downto 0);
        f  : out std_logic_vector(7 downto 0));
end component mux_8_1_8_bit_vhdl;

-- Declare 8 Bit Full Adder
component full_adder_8_bit_sch is
 Port ( a    : in  std_logic_vector(7 downto 0);
        b    : in  std_logic_vector(7 downto 0);
        cin  : in  std_logic;
        s    : out std_logic_vector(7 downto 0);
        cout : out std_logic;
        overflow : out std_logic);
end component full_adder_8_bit_sch;
```

5. These are Component Declarations. They tell the VHDL file what components are going to be used within this circuit. Here we are going to use the 8-bit 8 to 1 Multiplexer and the 8-bit Full Adder that you created earlier.
6. The body of the VHDL code is provided below. This template is missing a few parts which you will need to fill in:



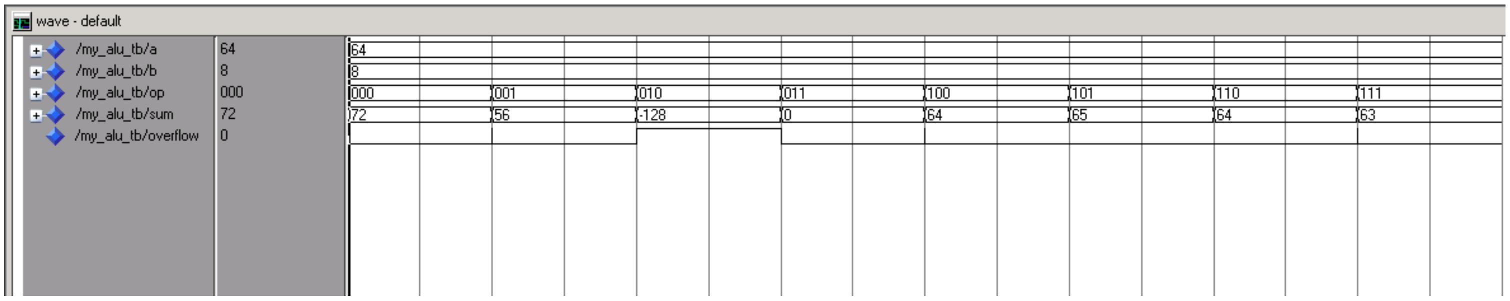
```
-- Invert A and B
a_inv <= not(a);
b_inv <= not(b);

-- Adder Cin Value based on KMAP Reduction
adder_cin <= ;

-- Instantiate 8-Bit Mux
mux_i : mux_8_1_8_bit_vhdl
 Port Map (
   i0  => b,
   i1  => b_inv,
   i2  => ,
   i3  => ,
   i4  => ,
   i5  => ,
   i6  => ,
   i7  => ,
   s   => op,
   f   => mux_out);

-- Instantiate 8-Bit Adder
adder_i : full_adder_8_bit_sch
 Port Map (
   A     => ,
   B     => mux_out,
   cin   => adder_cin,
   s     => ,
   cout => ,
   overflow => overflow);
```

7. Lines 79 and 80 are inverting A and B with the “not” operator. It is a bitwise operator which will invert each of the 8-bits for A and B respectively.
8. Line 83 is incomplete! Add the correct Carry-In Logic before the semi-colon based on the carry-in logic from Homework 8. Use the keywords: **and**, **or**, **not** instead of the symbols *, +, ‘
9. Line 86 is instantiating a single instance of the 8-bit 8 to 1 Multiplexer. The Port Map is like connecting the wires together in the Schematic. A few of the ports have already been connected, fill in the remaining ports. Notice that line 97 “f => mux_out” is connecting the output “**f**” of the multiplexer to the intermediate signal created on line 40 called “mux_out”. Then on line 103 mux_out is connected to the Input port B of the Full Adder. This is like drawing the wire between the output of the multiplexer to the input B of the Adder in a schematic.
10. Line 100 is instantiating the 8-bit Full Adder. Notice how you can combine Schematic designs with VHDL designs so easily! Again, some of the ports connections are missing, fill them in with the signals you think are necessary.
11. Save the VHDL file and Synthesize to check for syntax errors. Fix all errors before continuing to the simulation.

Simulation and Demo

1. Add a New Source of type Test Bench Waveform to the project: **my_alu_tb**
2. Set Clock Information: Combinational (or internal clock)
3. Set Initial Length of Test Bench: 800 ns
4. Click OK
5. Set A to decimal: 64
6. Set B to decimal: 8
7. Set Op to increment from 0 to 7 from the Pattern Wizard
8. Run ModelSim simulation and verify each operation was performed correctly

Expected Simulation Output



```
/my_alu_tb/a         64    64
/my_alu_tb/b         8     8
/my_alu_tb/op        000   000  001  010   011  100  101  110  111
/my_alu_tb/sum       72    72   56   -128  0    64   65   64   63
/my_alu_tb/overflow  0
```

Required Electronic Submission (uploaded to Moodle)

Print the Grading Rubric (the last page of this hangout) and write both names and section number at the top. Then print the two schematics and two VHDL files listed below as individual PDFs (make sure to add both names and section number as text to each schematic and as comments in the VHDL). Combine the four PDFs into a single PDF with both your last names as the filename (I.E. schmidt_conrad_ca2.pdf). Upload the single PDF (one upload per group) to Moodle. Finally, find one of the TAs **during their Office Hours** and demo the outputs of both ModelSim simulations.

1. Schematic of 1-Bit Full Adder
2. Schematic of 8-Bit Full Adder
3. VHDL of 8-bit 8 to 1 Multiplexer
4. VHDL of My ALU

Notes

You do NOT need to checkout a Nexys2 Board for this Computer Assignment. You are only using the software tools to build and test the designs. Future assignments will rely upon these components so make sure to keep a copy of your project's directory.

If you run into a License Error during Simulation save all files and close Xilinx ISE 7.1i. The open Xilinx ISE 7.1i from the Start Menu:

Start -> Mosaic XP -> Engineering -> Electrical -> Xilinx Webpack -> Xilinx ISE 7.1i

If you run into a ModelSim "read only" error on your Test Bench VHW file:

a. To avoid getting any errors with ModelSim in "read mode" you MUST create a new project directory WITHOUT any spaces! Make sure the path to the directory also does not have any spaces. If you are not sure where to save this project to, create the following directory on your desktop:
    i. ecgr2181
    ii. Then within this directory add a new directory for each project
    iii. The Path to this directory will now be: **U:\pc\win_data\Desktop\ecgr2181**

Computer Assignment #2 - Grading Rubric

Student 1 Name: ________________________________ Section Number: ____________

Student 2 Name: ________________________________

Uploaded schematics and VHDL as a Single PDF to Moddle:

☐	Schematic of 1-Bit Full Adder	____________ / 3 Points
☐	Schematic of 8-Bit Full Adder	____________ / 4 Points
☐	VHDL of 8-bit 8 to 1 Multiplexer	____________ / 3 Points
☐	VHDL of My ALU	____________ / 7 Points

Demo of the simulations:

☐	Simulation of ALU in ModelSim	____________ / 8 Points

Total ____________ / 25 Points