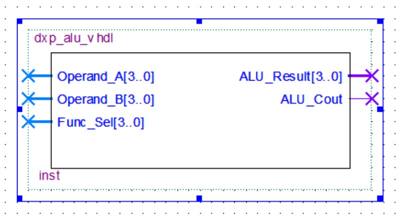
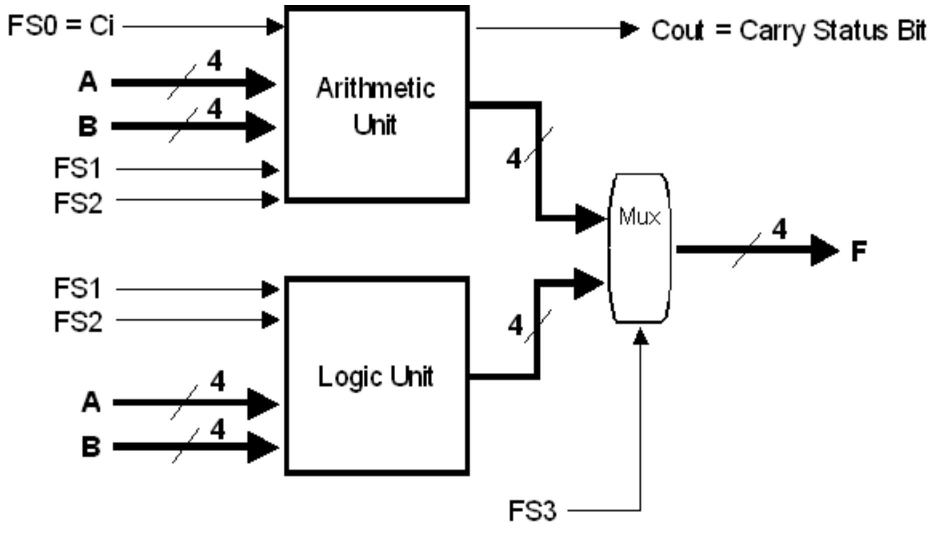
EEEE-220-Lab5

1) From mycourses download on your desktop this pdf file, dxp_alu_vhdl.vhd, and dxp_alu_vhdl_tb.vhd. This lab consists of two parts:
    a. A VHDL behavioral implementation and simulation of an ALU
    b. A Verilog structural implementation and simulation of the same ALU.
2) **Objective**: The objective of this lab exercise is to design and verify using a testbench, i.e. through simulation an **Arithmetic and Logic Unit** (ALU). The ALU is one of the most important blocks in a processor. The ALU is used in 8 of the 16 instructions/commands of the processor you'll be designing in this course and lab.
3) The next section describes the functionality and structure of the ALU.
4) Below is the high level symbol for the ALU:



5) The ALU above has the following Input / Output (I/O) Ports:
    a. Operand_A[3..0] – Input A (4-Bit)
    b. Operand_B[3..0] – Input B (4-Bit)
    c. Func_Sel[3..0] – Function Select Input (4-Bit)
    d. ALU_Result[3..0] – Output Result (4-Bit)
    e. ALU_Cout – Carry Status Bit Output.
6) At the next lower hierarchical level the ALU is structured as shown below. In the next figures FS=Func_Sel, F=ALU_Result, and Cout=ALU_Cout.



7) The functionality of the ALU is captured in the table below:

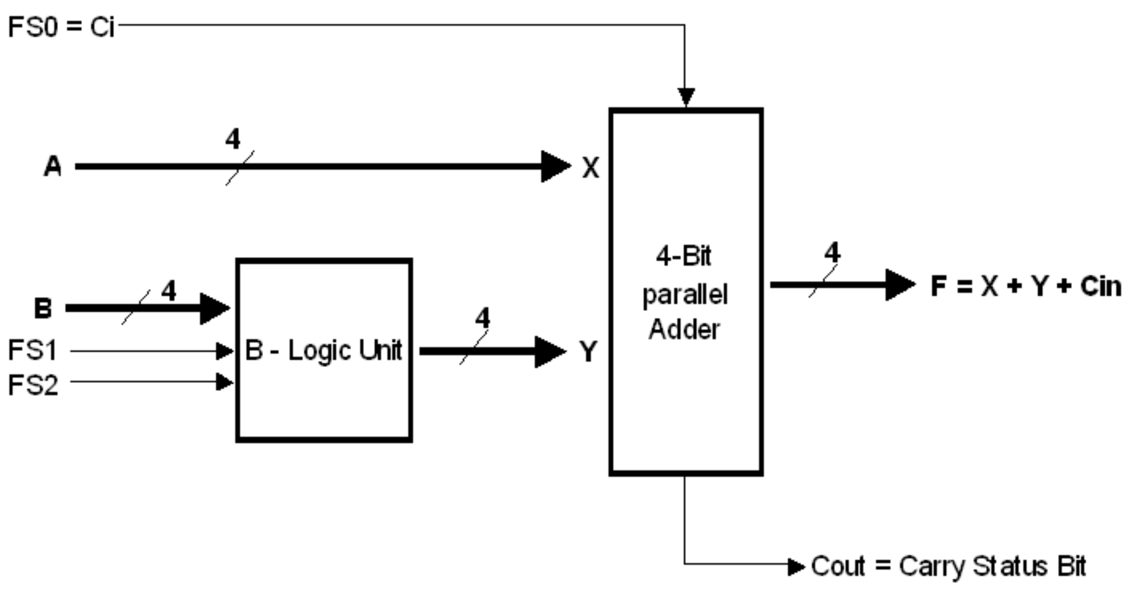
Function Select - FS[3..0]						
FS3	**FS2**	**FS1**	**FS0=C$_i$**		**Function Expression**	**Function Description**
0	0	0	0		F = A	Transfer A
0	0	0	1		F = A + 1	Increment A
0	0	1	0		F = A + B	Addition
0	0	1	1		F = A + B + 1	
0	1	0	0		F = A + (not B)	
0	1	0	1		F = A + (not B) + 1	Subtraction
0	1	1	0		F = A – 1	Decrement A
0	1	1	1		F = A	Transfer A
1	0	0	X		F = not A	NOT
1	0	1	X		F = A AND B	AND
1	1	0	X		F = A OR B	OR
1	1	1	X		F = SHR A	SHR

8) As you can see, the ALU is structurally partitioned in three major functional blocks: an arithmetic unit, a logic unit, and an output mux.

9) The **logic unit** implements the bitwise logic operations described in the table above, and detailed in the table below:

FS2	**FS1**	**Output**	**Operation**
0	0	F = NOT A	NOT
0	1	F = A AND B	AND
1	0	F = A OR B	OR
1	1	F = SHR A	SHR

10) The **arithmetic unit** implements addition, subtraction (using 2'sC), increment, and decrement operations. Below is its structure and truth table.



	Select			B-Logic Output		
FS0 = Ci	FS2	FS1		Y	Operation	Function
0	0	0		0000	F = A	Transfer
0	0	1		B[3..0]	F = A + B	Addition
0	1	0		NOT B[3..0]	F = A + (not B)	
0	1	1		1111	F = A – 1	Decrement
1	0	0		0000	F = A + 1	Increment
1	0	1		B[3..0]	F = A + B + 1	
1	1	0		NOT B[3..0]	F = A + (not B) + 1	Subtract
1	1	1		1111	F = A	Transfer

11) The truth table of the B-Logic Unit is:

FS2	FS1		Output
0	0		0000
0	1		B[3..0]
1	0		NOT B[3..0]
1	1		1111

12) **Part 1:** Download the files *dxp_alu_vhdl.vhd* and *dxp_alu_vhdl_tb.vhd*.
13) Create a project called fml_alu_vhdl. Copy/paste the code from the above two files into a fml_alu_vhdl.vhd and fml_alu_vhdl_tb.vhd. Update the fml in the code.
14) In the architecture of dxp_alu_vhdl a **selected signal assignment** is being used to implement the entire ALU at behavioral level.
15) I used 5-bit wide internal signals for the arithmetic and shift-right to be able to capture the value of the carry out.

16) The shift-right is explicit because I wanted to be able to output the value of the LSbit through the carry out port.
17) Take the time to read the code and make sense of the different statements and constructs. Except for the new selected signal assignment, which is pretty much self-explanatory, we have used all other ones.
18) Unlike the VHDL Testbenches you have seen so far, this one is using a for-loop to iterate through the 16 different combinations of the function select signals.
19) Inside the loop it generates stimuli for the values 0, 5, 10, and 15.
20) To assert the output values we would need to compute the results within the testbench. This would be done behaviorally, i.e. using code almost identical to the one we are trying to test.
21) Therefore, you will visually check each input combination in the waveform window. Most of the input combinations and their associated results are trivial to check.
22) Analyze, synthesize, and verify the design in simulation using the testbench provided.
23) This concludes part 1.
24) **Part 2:** In this part you will design and verify the same ALU using structural and, if you choose for some components to do so, dataflow Verilog.
25) At this point you may want to start accumulating components in your own fml_package.vhd. These components and package you can add and use in every project from here on. For example, you can use a slightly modified and simpler version of the 4-bit ripple adder in the arithmetic block.
26) Design/code each of the blocks shown in the figures above.
27) Based on the waveform results of the VHDL testbench, create a Verilog testbench to check and assert each case covered in the former. I know there are 64 combinations, but many repeat or can be copy/pasted because the differences are minimal.
28) In order to receive partial credit for any part of this design, you'll have to create a testbench for that block and prove that it is working correctly!
29) This concludes part 2.
30) Grading:
    a. 10 points for part 1 – VHDL compilation and simulation
    b. 30 points for part 2 – Verilog design/coding, compilation, and simulation
    c. Possible partial credits for part 2:
        i. 5 points for simulated logic unit
        ii. 5 points for simulated B-logic unit
        iii. 10 points for simulated arithmetic unit.
31) Show your working, i.e. compiled and simulated, designs to the TA. Write your report and upload it along with your project archives in the dropbox on mycourses, as described in the lab policy.
32) This concludes this week's lab.