

ECE 474a/574a - Programming Project 1 (30 points)

Due September 14, 2008 (Monday) at 8:00 pm

ECE 474A DELIVERABLES

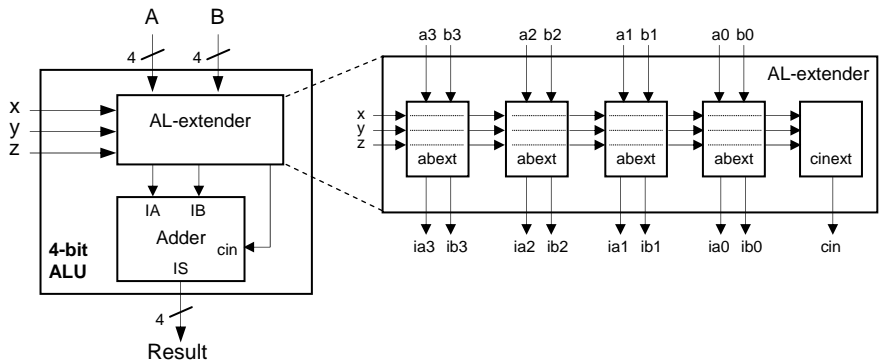
- Verilog code implementing behavioral 4-bit ALU, ALU_beh
- Verilog code implementing structural 4-bit ALU, ALU_struct
- Verilog testbench verifying functionality of both ALU implementations

ECE 574A DELIVERABLES

- Verilog code implementing behavioral 4-bit *modular* ALU, ALU_beh
- Verilog code implementing structural 4-bit *modular* ALU, ALU_struct
- Verilog testbench verifying functionality of both ALU implementations
- Text file indicating the maximum frequency of your structural and behavioral 4-bit ALUs
- Verilog code implementing 8-bit hierarchical ALU from structural 4-bit ALU modules
- Verilog testbench verifying functionality of the 8-bit ALU

PROJECT DESCRIPTION

In this project you will implement a 4-bit ALU using a structural and behavioral methodology. The ALU has two 4-bit inputs, *A* and *B*, with values specified in the two's complement format. Additionally, there are three 1-bit inputs *x*, *y*, and *z* which specify the operation performed by the ALU. The resulting value is outputted as a 4-bit value called *Result*. The ALU block diagram and ALU operation table is specified below.



Inputs			Operation
x	y	z	
0	0	0	Result = A
0	0	1	Result = B
0	1	0	Result = A + B
0	1	1	Result = A - B
1	0	0	Result = A + 1
1	0	1	Result = A'
1	1	0	Result = A AND B
1	1	1	Result = A OR B

The structural implementation should consist only of primitives or module definitions using primitives, with the ALU described as a hierarchical interconnection of those modules/primitives. Implement the underlying ALU structure utilizing an adder and AL-extender, the block diagram is provided above. Make sure to take advantage of hierarchy. Implementing the ALU as a single module is not good design and you will likely have a difficult time debugging your design.

In the behavioral implementation, the ALU specification should consist of arithmetic expressions, procedural assignments, or other Verilog control of flow structures. The behavioral description does not need to adhere to the structure above. Rather, the behavioral description simply implements the functionality of the ALU as defined by the operation chart.

Please make sure to declare your module as shown in the example below to help the grader interface to your module. For the structural ALU implementation replace “_beh” with “_struct”.

```

module ALU_beh (A, B, x, y, z, Result);
    input [3:0] A, B;
    input x, y, z;
    output [3:0] Result;
    ...

```

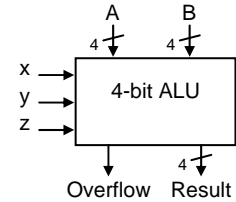
Additionally, you will need to create a testbench to check the functionality of your ALU. You should be able to use the same testbench to test both ALU implementations. Be sure to test a reasonable number of scenarios to be confident in your implementation.

ECE 574A ONLY

Add an extra 1-bit output called *Overflow* to your ALU design to detect when overflow occurs. Set *Overflow* = 1 when the result is not correct due to an overflow/underflow scenario. Otherwise *Overflow* = 0. The 4-bit ALU architecture should also be modular and allow designers to create larger ALUs by utilizing your 4-bit ALU design as a building block. Add any additional signals required to your design.

Declare your behavioral ALU module as shown in the example below to help the grader interface to your module. For the structural ALU implementation replace “_beh” with “_struct”. Add any new signal introduced to create the module design after “*Result*”.

```
module ALU_beh (A, B, x, y, z, Overflow, Result, ...);
  input [3:0] A, B;
  input x, y, z;
  output Overflow;
  output [3:0] Result;
  ...
endmodule
```



Once you have verified the functionality of both the structural and behavior designs, report the maximum frequency of both implementations. Frequency is dependent on the underlying hardware used, for this project assume *Family* = Spartan3E, *Device* = XC3S500E, *Package* = FG320, and *Speed* = -4. Part of being a graduate student is investigation solutions and figuring out how to use tools, thus as part of this project you will need to figure out how to determine the frequency of your circuit. As a hint you may want to take a look at http://www.ece.arizona.edu/~ece274/uploads/Labs/XilinxISE_SynthesisTutorial_Spartan3E.pdf (

Using the ALU_struct module as a building block, create a structural 8-bit ALU implementation. Provide a testbench illustrating the 8-bit ALU's functionality. You **do not** need to create a behavioral 8-bit ALU implementation.