CPRE 381 Lab 3
Module Reuse – Multipurpose Shifter

In this lab you will develop a 32-bit multipurpose shifter based on modules you build in Lab 1.

1-bit Shifter

1. Start by designing a 1-bit shifter module that you will be able to reuse for a larger design. Your module should have three 1-bit inputs (in, left, right), one 2-bit input (selector), and one output (out). When the selector is "01" it shifts to the left, and when "10" it shifts to the right. For other values of the selector no shift is performed. (Note: A left shift means that when looking at a string of values (MSB to the left, LSB to the right) the data will shift toward the MSB. This means that the output of the current bit will be the input value of the bit to the immediate right. A right shift will move the date toward the LSB and the current bit output will be the input value of the bit to the immediate left.) Be careful to draw your design and label your inputs with respect to the proper function.
2. Implement your design in verilog.
3. Verify that your design works as expected.

16-bit Multipurpose shifter

There are several different shifts that can be performed with a single hardware unit. These shifts have slightly different behaviors and are used for various purposes.

A **logical shift** simply takes the input and shifts it to the right or left. The bits that are shifted off the end of the structure are simply discarded. The bits that need to be filled on the other end are set to zero. For your shifter, the shift amount is only one bit, so one bit will be discarded and one bit will be set to 0. A shift-in bit can also be specified as an input and is useful when chaining multiple shifters to make a larger shifter. In this case a shift-out bit should also be specified as an output. The shift-in is used in place of a 0 for the bit to be filled in.

An **arithmetic shift** is similar to a logic shift except that it maintains a sign bit. Thus if the MSB is a 1 before the shift (the input value is negative) it will still be a 1 after the shift regardless of the shift direction. The same holds true for an MSB of 0. The MSB will not change during an arithmetic shift. An arithmetic shift left can have a shift-in bit other than 0, but an arithmetic shift right has no shift-in bit.

A third shift function is a **rotate**. A rotate is just like it sounds and no bits are discarded. The MSB is propagated to the LSB in a shift left and vice-versa in a shift right.

Finally, a shifter can be used as a component in a larger function to develop multiplication and division algorithms. This will be covered in more detail later and in

this lab you can consider the additional inputs to support these algorithms as alternate shiftin values.

Now it is time to reuse the 1-bit shifter to make a more useful functional unit to implement several types of shift operations.

1. Design a 16-bit shifter using a 1-bit shift module to implement the functions as shown in the table below. Verify that you have all of the correct inputs and outputs identified. You should have a 16-bit input (a), a 16-bit output (out), a 4-bit input for selection (sel), three additional 1-bit inputs for special operations (shiftin, dv,ms), and a 1-bit output for cascading (shiftout).
2. Implement your design in verilog.
3. Verify that your design works by testing multiple values for each of the possible functions.

MSB a[n-1]	LSB a[0]	Selector sel[3:0]	Output out[15:0]
Left Shift			
a[n-2]	shiftIn	0000	Logic shift left with shiftIn
a[n-2]	0	0001	Logic shift left
a[n-2]	a[n-1]	0010	Rotate left
a[n-2]	dv	0011	Divide shift left
a[n-1]	shiftIn	0100	Arithmetic shift left with shiftIn
a[n-1]	0	0101	Arithmetic shift left
a[n-1]	a[0]	0110	No shift
a[n-1]	a[0]	0111	No shift
Right Shift			
a[n-1]	a[1]	1000	Arithmetic shift right
a[n-1]	a[1]	1001	Arithmetic shift right
a[n-1]	a[1]	1010	Arithmetic shift right
a[n-1]	a[1]	1011	Arithmetic shift right
shiftIn	a[1]	1100	Logic shift right with shiftIn
0	a[1]	1101	Logic shift right
a[0]	a[1]	1110	Rotate right
ms	a[1]	1111	Multiply shift right

32-bit shifter from 16-bit shifters

1. Design a 32-bit shifter to reuse your 16-bit design.
2. Implement your design in verilog.
3. Verify the 32-bit design works for all functions. Does anything need to be modified?