Simulating the Shifter Sub-Block

Now that you have completely described the behavior of the *Shifter* sub-block in a flowchart view, we should examine the VHDL which is produced. From your *Flow Chart*, generate the VHDL codes for the *Shifter* sub-block.

Since this is your first flowchart view, there is a good chance that you will encounter an error or two the first time that you try to generate. If you do encounter errors, go back to the flowchart view and try to locate and correct them.

Once you have generated the shifter VHDL without errors, click **HDL > View Generated HDL** on the menu.

Take a few minutes to examine the code. Notice that the architecture description for the Shifter does not contain any component declarations or instantiations. Instead, there is a *process block* labeled *process0_proc*. This process block contains the sequence of statements which you entered in the *Decision Boxes* and *Action Boxes* of the flowchart. The declarative region of the process block also contains the declarations of all of the variables that were used in the flowchart to store intermediate results.

Simulating the Shifter

As with the *Logical* sub-block, just creating the design in **FPGA Advantage** is not enough, we must also verify that the design was entered correctly and that, if so, it works as intended. To do this, we will once again load the design into **ModelSim**, manually stimulate the input signals, and then examine the outputs in the Wave window.

First, you will need to compile the *Shifter* sub-block for simulation just as we did the *Logical* sub-block. This can be accomplished by clicking the **ModelSim Flow** button again,  . Don't worry about not finding the "Generate Through Components" option. Click the **No** button when presented with the option of **Enable data capture and show animation?**. This option is not functional since there is no clock signal.

Now we need to test the design. The following table list the test vectors which we will use and the expected results. You will need to create a **ModelSim** do file for your testing ease, configure the Wave window, run the do file, and verify the results.

A	ALUOp	ALUOp	ALUOp	SHAMT	Results		
	(SLL)	(SRL)	(SRA)	SHAMT	SLL	SRL	SRA
X"FEDCBA98"	00	10	11	00000	X"FEDCBA98"	X"FEDCBA98"	X"FEDCBA98"
X"FEDCBA98"	00	10	11	00100	X"EDCBA980"	X"0FEDCBA9"	X"FFEDCBA9"
X"FEDCBA98"	00	10	11	01000	X"DCBA9800"	X"00FEDCBA"	X"FFFEDCBA"
X"FEDCBA98"	00	10	11	10000	X"BA980000"	X"0000FEDC"	X"FFFFFEDC"
X"FEDCBA98"	00	10	11	11000	X"98000000"	X"000000FE"	X"FFFFFFFE"
X"FEDCBA98"	00	10	11	11100	X"80000000"	X"0000000F"	X"FFFFFFFF"
X"FEDCBA98"	00	10	11	11111	X"00000000"	X"00000001"	X"FFFFFFFF"

The above table lists a single test pattern for the input *A* which will be tested with several possible shifts. Although not a completely rigorous testing of the *Shifter* functionality, it should serve to expose almost any error in your design in the limited time we have for this tutorial.

One important case has been left out of the test, however, which you will also need to test. We have tested the right logical and arithmetic shifts with a vector that, if interpreted as two's complement, is a negative number. We have not tested to make sure that the right shifts will work correctly with a positive number.

Change the test pattern for the input *A*, so that the highest order bit is a '0' instead of a '1'. Run this pattern for several different shift amounts of both arithmetic and logical right shift and verify that the behavior is correct.

In the next tutorial, we will create the *Arithmetic* sub-block design.