Colby Banbury, Matt Stout

Lab 2 Report

***Abstract:***

The goal of this lab was to become more familiar with VHDL modeling and design, through the implementation of a 4-1 multiplexor, a 4-bit shift register, an 8-bit shift register, and a 4-bit integer adder/subtractor. Using the GHDL compiler, we were able to design and implement each of these behavioral models, as well as outline detailed schematics for each model.

***Division of Labor***:

The lab itself is comprised of four behavioral models, as well as testbenches and schematics. The areas include:

* 4-1 Multiplexor:
  + Testbench displaying all possible permutations
* 4-bit shift register:
  + Code implements LOAD, HOLD, RIGHTSHIFT, LEFTSHIFT triggered by rising edge
  + Include ENABLE and CLOCK inputs
  + SHIFT LEFTINPUT, SHIFT RIGHTINPUT controlled by a multiplexor
  + Testbench displaying all combinations of I\_SHIFT\_IN, sel, clock, enable, and two values of “I”
  + RTL circuit schematic design drawn by hand
* 8-bit shift register:
  + All functions included in the 4-bit shift register
  + Testbench displaying all combinations of input signals of I\_SHIFT\_IN, shift, clock, enable, load, and two values of “I”
  + RTL circuit schematic design drawn by hand
* 4-bit integer adder/subtractor
  + Testbench testing primary functions
  + RTL circuit schematic design drawn by hand

The design and strategy were discussed cooperatively before each phase, with Colby coding each segment, and Matt creating the testbenches. Frequent verification of the other team member’s work occurred during the process, to allow a fresh set of eyes to catch any errors or requirements that had yet to be implemented. The schematics were also created together, to allow for a visual representation of the task ahead.

***Detailed Strategy:***

When implementing the 4-1 multiplexor, we chose to directly map the inputs to the outputs using a 2-bit selector, which decided the assigned input accordingly based on the four possible values of the selector.

In creating the 4-bit shift register, we implemented it to perform one of four operations: hold, shift\_left, shift\_right, and load. Hold retains the current value to the register, shift\_left and shift\_right place the value in I\_SHIFT\_IN into the leftmost or rightmost bit, respectively, while shifting the other values. Finally, the load operation loads the new value in “I” to the register. A 2-bit selector determines which operation is to be performed, and the “enable” input determines if the register is enabled to shift. To implement the 8-bit shift register, we planned to essentially use the 4-bit shift register from the previous part, and use it twice across 8 bits, then map the two 4-bit outputs to the completed 8-bit output.

For the 4-bit integer adder, we combined 4 full adders. In addition to this, we included flags for overflow and underflow. For subtraction, the second integer would be altered to its negative two’s complement counterpart using XOR.

***Results:***

After extensive testing, our testbenches successfully tested all required and possible outcomes for the behavioral models to be fully functional. Our 4-1 multiplexor testbench uses an array of selection patterns to fully test all possible permutations of inputs. A similar pattern array style was used to test the shift registers, using varying bit combinations to assert the success of the shift, hold, or load. Thorough testing of the 4-bit adder compared the expected output values to the actual outputs across several tests, and found that both addition and subtraction function properly, with correct flagging of overflow and underflow.

***Conclusion:***

The project was successfully completed, but some challenges were faced that made it difficult to complete. Conceptualizing the design of the behavioral models and components, as well as the proper way to thoroughly test each possible permutation, proved challenging and required significant time and effort to overcome. We had sufficient time to fully test all permutations that came to mind, but more time and resources could have allowed us to optimize the models and create more testing scenarios.