

ELEC 5200/6200 – VHDL Modeling Project #2

Due: Friday, February 10

Design a register-transfer level VHDL model of an unsigned binary multiplier, in accordance with the description of the “Third Version of the Multiplication Hardware and Algorithm”, presented in Section 3.4 of the textbook. The requirements for this design are as follows.

1. The circuit is to produce a 16-bit product of two 8-bit unsigned binary integers.
2. The register-level structure, with the exception of the word size, should be that described in the textbook as the “refined version” (Figure 3.7). The control algorithm will be similar to that of Figure 3.6, modified for the “refined version”.
3. The top-level model is to contain only five component instantiations: 8-bit adder (modify your previously-designed 32-bit adder for this), three 8-bit registers, and a controller.
4. The inputs/outputs of the top-level model are:
 - Multiplicand** and **Multiplier** (two 8-bit inputs)
 - Product** (16-bit output)
 - Clock** and **Strobe** (1-bit inputs)
 - Ready** (1-bit output)
5. The multiplier should start in the “Done” state, and remain in that state until the **Strobe** input is activated, at which time it should generate the product of its **Multiplicand** and **Multiplier** inputs. The multiplier should then remain in the “Done” state, with the result on the **Product** output, until another **Strobe** signal is detected, at which time the next product should be computed.
6. The **Ready** output should be 1 when the multiplier is in the “Done” state, and otherwise 0. (Indicating when valid product is available, and when a new multiplication can begin.)
7. Design two new “components” to use in this model: a controller, and a register-level model of an 8-bit parallel-load shift register. Three instances of the shift register are to be used in the top-level model.
8. The shift register is to have an 8-bit (vector) input, an 8-bit vector output, a serial data input, an active-high clock input, and three active-high control inputs:
 - **Shift Right** (the current value in the register shifts one bit to the right, with the left-most bit replaced by the serial input)
 - **Write** (load the 8-bit input into the shift register)
 - **Clear** (set the shift register to all zeros)All register operations are to occur on the rising edge of the clock.

*NOTE: All three functions are not needed for every register in the multiplier, but to reduce the number of new components, simply use **three copies of the same register** and disable any unnecessary functions.*
9. The controller should be a behavioral model of the algorithm of Figure 3.6, modified as necessary for the “refined version” and to incorporate the **Strobe** signal described in step 5 above. Note that the controller must include a counter that can be tested to determine when the operation is finished (don’t create a separate component for this). Also note that in the “Start” state, the two operands should be loaded into registers, the accumulator cleared, and the controller’s counter reset.

TO BE SUBMITTED:

- Insert your new models into your notebook on top of the previous designs.
- Include three separate simulations: the parallel-load shift register, the controller, and the overall multiplier. Use enough test cases to verify the functionality of each.