Average of 2 8-Bit Numbers

Computing Hardware Synthesizable Average Rounding Up

Ethan Gibson

College of Engineering, EE480

University of Kentucky

Lexington, Kentucky, United States

eagi223@g.uky.edu

***Abstract*—This paper provides insight into the implementation of a hardware synthesizable 2 8-bit number average calculator that rounds up using the Verilog HDL.**

***Keywords—average, rounding, hardware, verilog***

# Implementation

In the reference module called refaverage, we took two inputs a and b. The inputs and output are all 8-bit unsigned binary values. To compute the average we add inputs a and b together using the built in addition operator and we also add 1 to this sum, which is saved to a temporary value. Our output is set equal to the the temporary value once it is shifted one place to the right. What this is doing is giving you the next value greater than sum of a and b, and then removing the lowest bit. This method gives you the average rounded to the next highest integer while dealing with any possible overflow.

The next module is called adder, which is just a basic full adder implementation. A 9 bit ripple carry adder is the next module that is implemented, called add9, combines multiple iterations of the add module to implement the addition operator. The full adder and ripple carry adder modules written by Dr. Dietz in the example project 1 were referenced to build this in order to save time.

The final supporting module implements the “>>1” operation. This module shifts all bits to the right one place by initializing a temporary register and shifting all [n+1] index values from the input register to the [n] index of the temporary register. The temporary register is what is output.

For the synthesizable averaging module, the implementation was very similar to that of the reference module. The concept is that the two input integers are added to each other and then to one. The rightmost bit is truncated all all the following bits are shifted a single space to the right. Instead of using the “+” and “>>” operators like the reference did, the manually defined full adder, ripple carry adder and bit shifter were used. All the inputs were converted to 9 bits to account for possible overflow. Originally, the algorithm provided in the project specification ((A^B)+((A&B)<<1)) was going to be used for the implementations since it is supposed to manage overflows automatically, but it would never produce the correct averages. After the procedure of computing the average has completed, the highest bit is truncated so that the result is an 8 bit integer.

The last module that was written was the testbench module. This module simulates the inputs into the other modules and records the output. The purpose of this particular test bench is to run both the reference averaging module and the synthesizable one concurrently and compare the values. After running through all possible averages of 2 8-bit numbers, the result of how many cases passed and how many failed is displayed to the console. This is accomplished using a loop nested inside another loop that will let a and b be all possible combinations of 8-bit values. Each combination is fed into the reference and synthesizable averaging modules individually and the two results are compared with a comparison statement. If they are not equal, the reference is considered the correct value, so the wrong value is flagged and displayed to the console with the correct value.