**Lab Number: 9**

**Section Number: 001**

**Names: Barak Barclay**

**Assigned Date: 04/28/2016**

**Due Date: 04/28/2016**

**Introduction:**

This lab is a series of modules – written in Verilog, and simulated in Modelsim – that demonstrate many commands that design a register transfer level. Part one is a module showing the difference between blocking and non-blocking procedural statements. Part two is a module that demonstrates the arithmetic operators +,-,\*,/,%, and \*\*. Part three is a module demonstrating bitwise and reduction operators ~,&,^, and |. Part four is a module demonstrating shift and concatenation/shmuck operators >>,<<.>>>, and {,}. Part five is a module demonstrating relational operators >,==,!=,===,>=, and <=. Behavioral modeling will be used to create all of these modules. How the code works will be explained thoroughly in the conclusion. All code and snips of the transcript box are in the parts below.

**Part 1:**

module Lab9();

reg a\_nb, b\_nb, c\_nb, a\_b, b\_b, c\_b;

initial begin

a\_nb <= #100 1'b1;

b\_nb <= #100 1'b0;

c\_nb <= #100 1'b1;

$display ("a\_nb=%b", a\_nb);

$display ("b\_nb=%b", b\_nb);

$display ("c\_nb=%b", c\_nb);

end

initial begin

a\_b = #100 1'b1;

b\_b = #100 1'b0;

c\_b = #100 1'b1;

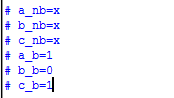
$display ("a\_b=%b", a\_b);

$display ("b\_b=%b", b\_b);

$display ("c\_b=%b", c\_b);

end

endmodule



**Part 2:**

module Lab9();

reg[4:0] a,b,c,d,e,f,g,h;

initial begin

a = #100 4'b0101;

b = #100 4'b0010;

c = #100 a+b;

d = #100 a-b;

e = #100 a\*b;

f = #100 a/b;

g = #100 a%b;

h = #100 a\*\*b;

$display ("a=%d", a);

$display ("b=%d", b);

$display ("a+b=%d", c);

$display ("a-b=%d", d);

$display ("a\*b=%d", e);

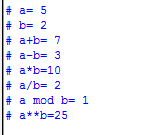
$display ("a/b=%d", f);

$display ("a mod b=%d", g);

$display ("a\*\*b=%d", h);

end

endmodule



**Part 3:**

module Lab9();

reg[3:0] a,b,c,d,e,f;

initial begin

a = #100 4'b1010;

b = #100 4'b0010;

c = #100 ~a;

d = #100 a&b;

e = #100 a^b;

f = #100 a|b;

$display ("a=%b", a);

$display ("b=%b", b);

$display ("~a=%b", c);

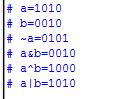
$display ("a&b=%b", d);

$display ("a^b=%b", e);

$display ("a|b=%b", f);

end

endmodule



**Part 4:**

module Lab9();

reg[3:0] a,b;

reg[7:0] c,d,e,f;

initial begin

a = #100 4'b0111;

b = #100 4'b0000;

c = #100 a>>1;

d = #100 a<<1;

e = #100 a>>>1;

f = #100 {a,b};

$display ("a=%b", a);

$display ("b=%b", b);

$display ("Logical right shift of a =%b", c);

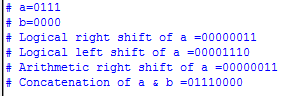
$display ("Logical left shift of a =%b", d);

$display ("Arithmetic right shift of a =%b", e);

$display ("Concatenation of a & b =%b", f);

end

endmodule



**Part 5:**

module Lab9();

reg[3:0] a,b,c,d;

reg e,f,g,h,i,j;

initial begin

a = #100 4'b1010;

b = #100 4'b0000;

c = #100 4'b1zx0;

d = #100 4'b1xx0;

e = #100 a>b;

f = #100 a==b;

g = #100 a!=b;

h = #100 c===d;

i = #100 a>=b;

j = #100 a<=b;

$display ("a = %b", a);

$display ("b = %b", b);

$display ("c = %b", c);

$display ("d = %b", d);

$display ("a>b = %b", e);

$display ("a==b = %b", f);

$display ("a!=b = %b", g);

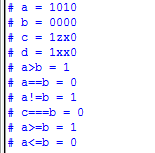
$display ("c===b = %b", h);

$display ("a>=b = %b", i);

$display ("a<=b = %b", j);

end

endmodule



**Conclusion:**

As mentioned above, all of these modules – that design a register transfer level – were written with behavior modeling. None of them have inputs or outputs, but they all have temporary memory locations called registers. These are declared the same way as inputs and outputs are i.e. *reg[3:0] a,b,c,d; reg e,f,g,h,i,j;* Note that if the variable starts, or would become, more than one bit long, vector notation has to be used. After declaring the registers, and in between the *initial begin* and *end,* values for some of the registers were set, the other registers were set to some operation of dealing with one or two of the previous registers, and display commands were written to display the values of all the registers in the transcript box. Here are examples of the three respectfully: *a = #100 4'b1010; e = #100 a>b; $display ("a = %b", a);* The first example makes register *a* 10102 after 100ns. The second uses the operator *>* to test of *a* is greater than *b,* returns true (1) or false (0), and sets *e* equal to the value. The last causes “a = 1010” to appear in the transcript window because *a* was previously set to 10102. All of the modules were created the same way making slight variations for values, operators and vector length when needed.