Name: REIMARC G. CORPUZ **Date: OCTOBER 19, 2022**

Course and Section: BSCPE 3GF **Score:**

PPT Exercise # 1:

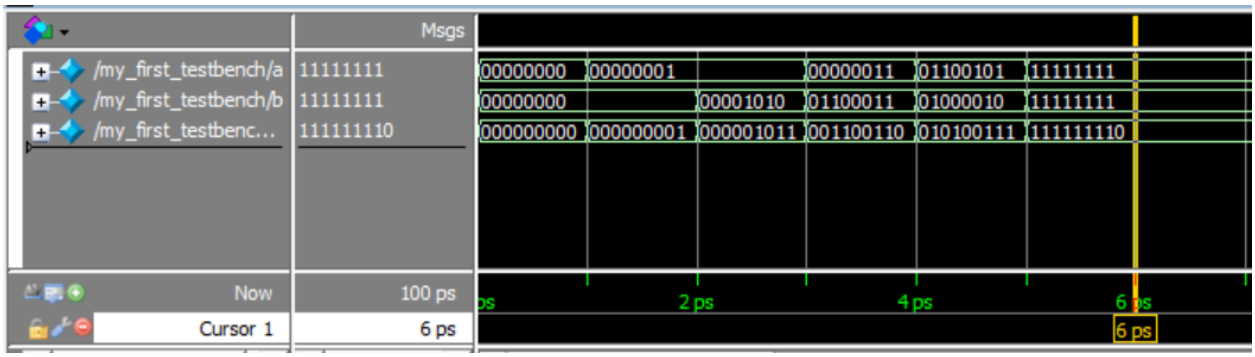
Verilog Code Structure

```
module adder8bit (
  input [7:0] a,
  input [7:0] b,
  output [8:0] sum
  );

    assign sum = a + b;

endmodule
```

Verilog Testbench

```
module my_first_testbench();
  reg [7:0] a = 0;
  reg [7:0] b = 0;

  wire [8:0] sum;

  adder8bit ADDER1(
    .a(a),
    .b(b),
    .sum(sum)
  );

  initial begin
    $monitor("a=%d, b=%d, sum=%d", a,b,sum);
      end
      initial begin
    #1;
    a = 1;
    #1;
    b = 10;
    #1;
    a = 3;
    b = 99;
    #1;
    a = 101;
    b = 66;
    #1;
    a = 255;
    b = 255;
    end

endmodule
```

Output Waveform

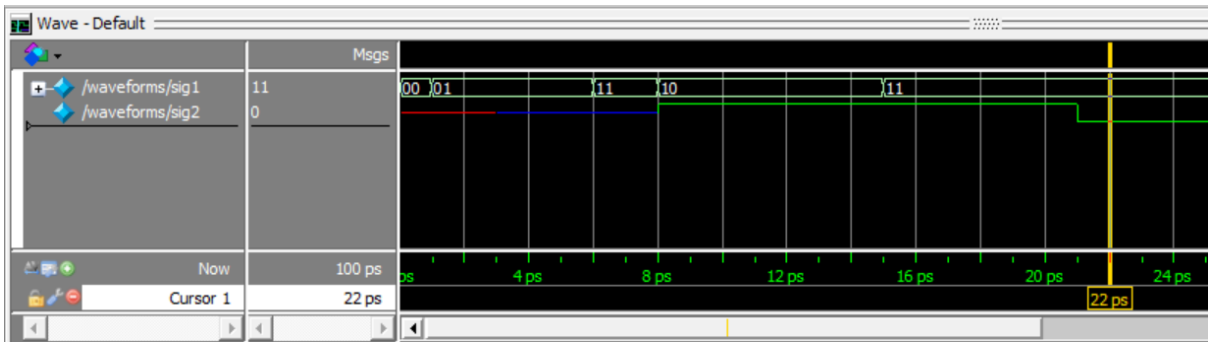


PPT Exercise # 2:

Verilog Code Structure

```
Ln#
1   module waveforms();
2     reg [1:0] sig1;
3     reg       sig2;
4
5       initial begin
6         sig1 = 0;
7       #1;
8       sig1[0] = 1'b1;
9       #2;
10      sig2 = 1'bz;
11      #3;
12      sig1[1] = 1'b1;
13      #2;
14      sig1 = 2'b10;
15      sig2 = 1'b1;
16      #7;
17      sig1[0] = 1'b1;
18      #6;
19      sig2 = ~sig2;
20      #4;
21    end
22  endmodule
23
```

Output Waveform

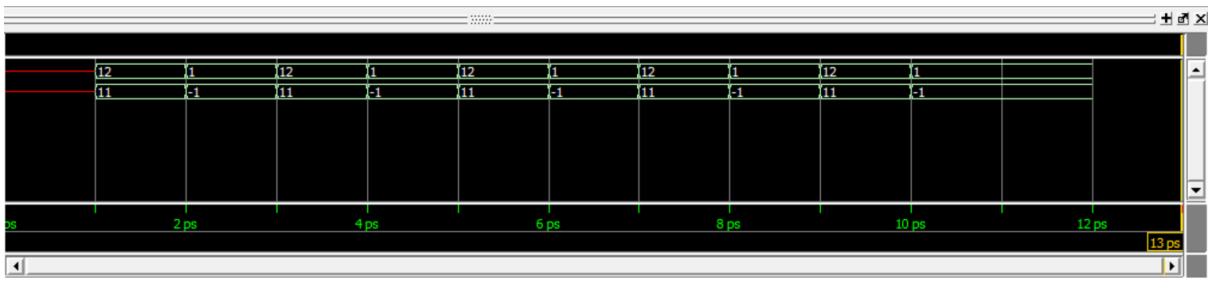


V. Evaluate my Learning

A. A Verilog code is attached as follows. What would be the output of this?

```
Ln#
1   module fancy2;
2   integer i,j;
3   initial repeat(5)
4   begin
5
6     #1 j=0;
7       while(j<=10)
8       begin
9         j=j+1;
10        for(i=0;i<=j;i=i+1) $write(" b");
11        $display("*");
12      end
13    #1 while(j>=0)
14    begin
15      for(i=0;i<=j;i=i+1) $write(" c");
16      $display("*");
17      j=j-1;
18    end
19
20  end
21  initial #12 $stop;
22  endmodule
23
```

Output and Waveform



```
# Reading C:/altera/13.0sp1/modelsim_ase/tcl/vsim/pref.tcl
# Loading project fancy2
ModelSim> vsim -gui work.fancy2
# vsim -gui work.fancy2
# Loading work.fancy2
add wave -position end  sim:/fancy2/i
add wave -position end  sim:/fancy2/j
VSIM 4> run
# bb*
# bbb*
# bbbb*
# bbbbb*
# bbbbbb*
# bbbbbbb*
# bbbbbbbb*
# bbbbbbbbb*
# bbbbbbbbbb*
# bbbbbbbbbbb*
# bbbbbbbbbbbb*
# cccccccccccc*
# ccccccccccc*
# cccccccccc*
# ccccccccc*
# cccccccc*
# ccccccc*
# cccccc*
# ccccc*
# cccc*
# ccc*
# cc*
# c*
# bb*
# bbb*
# bbbb*
# bbbbb*
# bbbbbb*
# bbbbbbb*
# bbbbbbbb*
# bbbbbbbbb*
# bbbbbbbbbb*
# bbbbbbbbbbb*
# bbbbbbbbbbbb*
# cccccccccccc*
# ccccccccccc*
# cccccccccc*
# ccccccccc*
# cccccccc*
# ccccccc*
# cccccc*
# ccccc*
# cccc*
# ccc*
# cc*
```

```
# bbbbbbbbb*
# bbbbbbbbbb*
# bbbbbbbbbbb*
# bbbbbbbbbbbb*
# cccccccccccc*
# ccccccccccc*
# cccccccccc*
# ccccccccc*
# cccccccc*
# ccccccc*
# cccccc*
# ccccc*
# cccc*
# ccc*
# cc*
# c*
# bb*
# bbb*
# bbbb*
# bbbbb*
# bbbbbb*
# bbbbbbb*
# bbbbbbbb*
# bbbbbbbbb*
# bbbbbbbbbb*
# bbbbbbbbbbb*
# bbbbbbbbbbbb*
# cccccccccccc*
# ccccccccccc*
# cccccccccc*
# ccccccccc*
# cccccccc*
# ccccccc*
# cccccc*
# ccccc*
# cccc*
# ccc*
# cc*
```

```
# c*
# bb*
# bbb*
# bbbb*
# bbbbb*
# bbbbbb*
# bbbbbbb*
# bbbbbbbb*
# bbbbbbbbb*
# bbbbbbbbbb*
# bbbbbbbbbbb*
# bbbbbbbbbbbb*
# cccccccccccc*
# ccccccccccc*
# cccccccccc*
# ccccccccc*
# cccccccc*
# ccccccc*
# cccccc*
# ccccc*
# cccc*
# ccc*
# cc*
# c*
# bb*
# bbb*
# bbbb*
# bbbbb*
# bbbbbb*
# bbbbbbb*
# bbbbbbbb*
# bbbbbbbbb*
# bbbbbbbbbb*
# bbbbbbbbbbb*
# bbbbbbbbbbbb*
# cccccccccccc*
# ccccccccccc*
# cccccccccc*
# ccccccccc*
# cccccccc*
# ccccccc*
# cccccc*
# ccccc*
# cccc*
# ccc*
# cc*
# c*
# Break in Module fancy2 at C:/altera/13.0sp1/fancy2.v line 21

VSIM 5>
```

B. Modify the Verilog code to delete b and c in the write statement lines. What would be the output of this? Explain.

- If I delete b and c in the statement lines, the space will only display that satisfies the condition of;
  while(j<=10)
  begin
  j=j+1;
  for(i=0;i<=j;i=i+1) // for b and;

  for(i=0;i<=j;i=i+1)
  j=j-1; // for c

  So, the statement will look like multiple halves of a diamond because it only commands to display $display("*");

Verilog Code

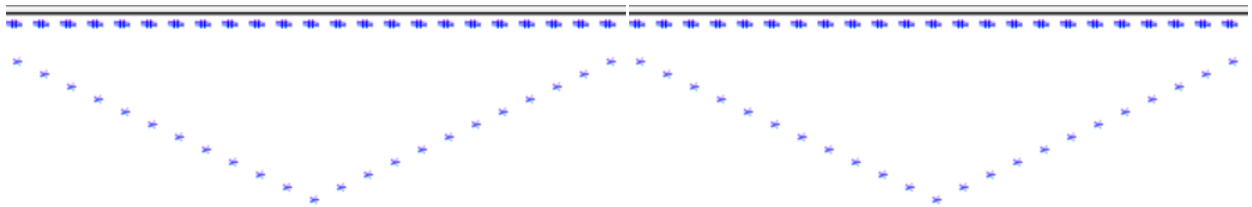
C:/altera/13.0sp1/fancy2.v (/fancy2) - Default

```
module fancy2;
integer i,j;
initial repeat(5)
begin

  #1 j=0;
    while(j<=10)
    begin
      j=j+1;
      for(i=1;i<=j;i=i+1) $write(" ");
      $display("*");
    end
  #1 while(j>=0)
  begin
    for(i=0;i<=j;i=i+1) $write(" ");
    $display("*");
    j=j-1;
  end

end
initial #12 $stop;
endmodule
```

Display Output

```
VSIM 10> run
#   *
#    *
#     *
#      *
#       *
#        *
#         *
#          *
#           *
#            *
#             *
#              *
#             *
#            *
#           *
#          *
#         *
#        *
#       *
#      *
#     *
#    *
```

```
#   *
#    *
#     *
#      *
#       *
#        *
#         *
#          *
#           *
#            *
#             *
#              *
#             *
#            *
#           *
#          *
#         *
#        *
#       *
#      *
#     *
#    *
```



```
#   *
#    *
#     *
#      *
#       *
#        *
#         *
#          *
#           *
#            *
#             *
#              *
#             *
#            *
#           *
#          *
#         *
#        *
#       *
#      *
#     *
#    *
# Break in Module fancy2 at C:/altera/13.0sp1/fancy2.v line 21
```

C. Modify the Verilog code and try other combinations of i and j values. Explain your combinations.

```
module fancy2;
integer i,j;
initial repeat(1)
begin

  #1 j=0;
    while(j<=11)
    begin
      j=j+1;
      for(i=1;i<=j;i=i+1) $write("b");
      $display("*");
    end
  #1 while(j>0)
  begin
    for(i=0;i<=j-1;i=i+1) $write("c");
    $display("*");
    j=j-1;
  end

end
initial #12 $stop;
endmodule
```

- Since in the previous simulation b starts in double b (bb), in the for loop of b I changed the initial value of “i” into 1 to start the counting in single b (b). I made the range of “j” to 11. So, the initial value of “j” will start from 0 up to 11. Since the condition is in a loop “j” will continue to add 1, which is why the last display of b is counted as 12. After that, the last value of “j” which is 12 will satisfy the condition in the loop of “c”. The value of “j” will decrement by 1. In the while loop of “c” (while(j>=0)), if do not delete the “=” operation the decrement will start at 13. As I said, the value of “j” will continue to increment if it satisfies the condition. I also insert minus 1 (- 1) to “j” in the for loop of “c” to make the display without an excess line. Because it satisfies the condition in “i<=j” which is 0 is less than or equal to 0. That is why it will print only the “*” symbol.
- In other words, my modified code will display an equal number of b and c.
- I also change the repeat value to 1 instead of 5 to easily visualize the display output.

D. What would be the output of this? Explain.

```
VSIM 5> run
# b*
# bb*
# bbb*
# bbbb*
# bbbbb*
# bbbbbb*
# bbbbbbb*
# bbbbbbbb*
# bbbbbbbbb*
# bbbbbbbbbb*
# bbbbbbbbbbb*
# bbbbbbbbbbbb*
# cccccccccccc*
# ccccccccccc*
# cccccccccc*
# ccccccccc*
# cccccccc*
# ccccccc*
# cccccc*
# ccccc*
# cccc*
# ccc*
# cc*
# c*
# Break in Module fancy2 at C:/altera/13.0spl/fancy2.v line 21
```

- The display output shows half of the diamond vertically with an equal number of “b” above and “c” below. The loop in “b” is incrementing by 1, while the loop in “c” is decrementing also by 1. The middle part of “b” and “c” are both counted in 12 characters.

E. Modify the Verilog code to replace the “always” statement with an “initial” statement. What would be the output of this? Explain.

Verilog Code

```
Ln#
1   module fancy3;
2   reg[11:0]a;
3   always
4   begin
5     #0 $display("See this: ah=%d, ad=%h, ao=%o, ab=%b",a,a,a,a);
6     #1 $display("How about this? ah=%0d, ad=%0h, ao=%0o, ab=%0b",a,a,a,a);
7       a=a+7;
8
9   end
10  initial
11  begin
12    a=0;
13    #10 $stop;
14  end
15  endmodule
16
```

Display Output

```
VSIM 2> run
# See this: ah=   0, ad=000, ao=0000, ab=000000000000
# How about this? ah=0, ad=0, ao=0, ab=0
# See this: ah=   7, ad=007, ao=0007, ab=000000000111
# How about this? ah=7, ad=7, ao=7, ab=111
# See this: ah=  14, ad=00e, ao=0016, ab=000000001110
# How about this? ah=14, ad=e, ao=16, ab=1110
# See this: ah=  21, ad=015, ao=0025, ab=000000010101
# How about this? ah=21, ad=15, ao=25, ab=10101
# See this: ah=  28, ad=01c, ao=0034, ab=000000011100
# How about this? ah=28, ad=1c, ao=34, ab=11100
# See this: ah=  35, ad=023, ao=0043, ab=000000100011
# How about this? ah=35, ad=23, ao=43, ab=100011
# See this: ah=  42, ad=02a, ao=0052, ab=000000101010
# How about this? ah=42, ad=2a, ao=52, ab=101010
# See this: ah=  49, ad=031, ao=0061, ab=000000110001
# How about this? ah=49, ad=31, ao=61, ab=110001
# See this: ah=  56, ad=038, ao=0070, ab=000000111000
# How about this? ah=56, ad=38, ao=70, ab=111000
# See this: ah=  63, ad=03f, ao=0077, ab=000000111111
# Break in Module fancy3 at C:/altera/13.0spl/fancy3.v line 13
```

- From the display output it shows the decimal, hexadecimal, octal, and binary equivalent value of “a”. The initial value of “a” is 0 and it will increment by 7. It will stop incrementing when it is by ten times.

F. Modify the Verilog code to replace the “a=a+7” statement with “a=a-7”. What would be the output of this? Explain what could happen.

```
Ln#
1   module fancy3;
2   reg[11:0]a;
3   always
4   begin
5     #0 $display("See this: ah=%d, ad=%h, ao=%o, ab=%b",a,a,a,a);
6     #1 $display("How about this? ah=%0d, ad=%0h, ao=%0o, ab=%0b",a,a,a,a);
7       a=a-7;
8
9   end
10  initial
11  begin
12    a=0;
13    #10 $stop;
14  end
15  endmodule
16
```

```
VSIM 4> run
# See this: ah=   0, ad=000, ao=0000, ab=000000000000
# How about this? ah=0, ad=0, ao=0, ab=0
# See this: ah=4089, ad=ff9, ao=7771, ab=111111111001
# How about this? ah=4089, ad=ff9, ao=7771, ab=111111111001
# See this: ah=4082, ad=ff2, ao=7762, ab=111111110010
# How about this? ah=4082, ad=ff2, ao=7762, ab=111111110010
# See this: ah=4075, ad=feb, ao=7753, ab=111111101011
# How about this? ah=4075, ad=feb, ao=7753, ab=111111101011
# See this: ah=4068, ad=fe4, ao=7744, ab=111111100100
# How about this? ah=4068, ad=fe4, ao=7744, ab=111111100100
# See this: ah=4061, ad=fdd, ao=7735, ab=111111011101
# How about this? ah=4061, ad=fdd, ao=7735, ab=111111011101
# See this: ah=4054, ad=fd6, ao=7726, ab=111111010110
# How about this? ah=4054, ad=fd6, ao=7726, ab=111111010110
# See this: ah=4047, ad=fcf, ao=7717, ab=111111001111
# How about this? ah=4047, ad=fcf, ao=7717, ab=111111001111
# See this: ah=4040, ad=fc8, ao=7710, ab=111111001000
# How about this? ah=4040, ad=fc8, ao=7710, ab=111111001000
# See this: ah=4033, ad=fc1, ao=7701, ab=111111000001
# Break in Module fancy3 at C:/altera/13.0sp1/fancy3.v line 13
```

- If I change the operation to “-“, it still decrementing. But compared to normal decrement the given number is to be minus by the larger number. The difference is that the initial value of “a” is 0 minus the greater number, so the equivalent value is greater than the value I get in the incrementation. The result value is shown based on the negative value of number systems.