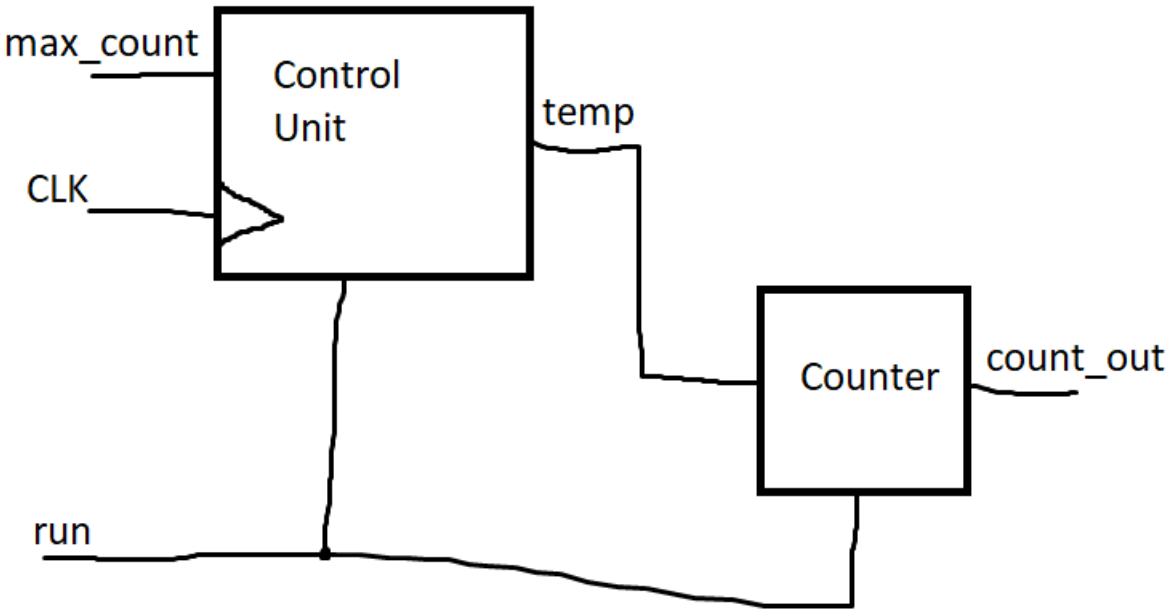
CDA 4203L
Computer System Design Lab

Lab 3 Report
Programmable BCD Counter

Today's Date:	02/10/2022
Team Members:	Thomas Bivins
	Patrick Cook
	Josue Lugo Roldan
Work Distribution:	Briefly explain the tasks completed by each team member Thomas and Patrick – Code writing and video recording Josue – Code troubleshooting and lab report All 3 worked on troubleshooting and synthesizing into board.
No. of Hours Spent:	10
Exercise Difficulty: (Easy, Average, Hard)	Average
Any Other Feedback:	

Problem 1: Draw overall block diagram of the programmable counter. Briefly explain how your design works. For each block, include the Verilog code. *Use as many pages as needed.*

This programmable BCD counter contains a control unit that controls the clock and the run signal which tells the counter when to start counting and how far to count. The following code shows the section of code that controls the clock and the run signals that tell the counter when to start start counting



This module contains the logic to stop a counter when it reaches a designated value. The maximum value is 99.

```
1  // Wrapper to add programmability to a 7-bit counter.
2  // This module contains the logic to stop a counter when it
3  // reaches a designated value. The maximum value is 99 (decimal)
4
5  module prog_count_7(max_count, run, CLK, count_out);
6
7  input [6:0] max_count;
8  input run, CLK;
9  output [6:0] count_out;
10
11 // Wires/Registers required go here.
12 reg clk;
13 reg [6:0] temp;
14 // 7-bit counter instance
15 count_7 counter_1(
16         .run(run),
17         .CLK(clk),
18         .count_out(count_out)
19 );
20
21
22 // TODO: Write logic for Counter control
23 //In your always block, first check if run == 0, then read //max_count. Compare max_count with count_out to stop.
24 always @(CLK or max_count) begin
25       if (run == 0) begin
26          clk = CLK;
27          temp = max_count;
28    end
29    else begin
30       if (temp == count_out || count_out == 99) begin
31          clk = 0;
32       end
33       else if (temp < count_out) begin
34          clk = 0;
35       end
36       else begin
37          clk = CLK;
38       end
39    end
40 end
41
42 endmodule
```

This is the top module for the programmable BCD counter. It implements a programmable 7-bit counter and a binary-to-bcd converter that can output two digits.

```
1  // This is the top module for the programmable BCD counter.
2  // It implements a programmable 7-bit counter and a binary-
3  // to-bcd converter that can output two digits.
4  //
5  // Use of this template is optional
6
7  module bcd_count_7(max_count, CLK, run, digit_1, digit_2);
8
9    input [6:0] max_count;
10   input CLK, run;
11   output [3:0] digit_1;
12   output [3:0] digit_2;
13
14   wire [6:0] cout_bin;
15
16
17   // TODO: Wires and registers for interconnect if needed
18
19   // Programmable 7-bit counter module
20   prog_count_7 counter(
21          .max_count(max_count),
22          .run(run),
23          .CLK(CLK),
24          .count_out(cout_bin)
25   );
26   // Binary-to-BCD Converter for converting count_out to BCD
27   binary_bcd_2 bcd_converter(
28                .bin_in(cout_bin),
29                .digit_1(digit_1),
30                .digit_2(digit_2)
31   );
32
33 endmodule
```

Problem 2: Testbench code: Include your Verilog testbench. *Use as many pages as needed.*

```
// Example BCD Counter Testbench
module bcd_count_tb_example;

   // Inputs
   reg [6:0] max_count;
   reg clk, run;

   // Outputs
   wire [3:0] digit_1, digit_2;

   // UUT - BCD Counter
   bcd_count_7 uut(.max_count(max_count), .run(run), .clk(clk), .digit_1(digit_1), .digit_2(digit_2));

   // Clock Generator
   always begin
      clk = ~clk;
      #5;
   end

   // Simulation
   initial begin
      clk = 0;
      run = 0;
      max_count = 0;
      #100;
      // Set MAX to 73 while run=0
      max_count = 73;
      #10;
      // Wait, then set run to 1
      run = 1;
      #150;
      // Change MAX while run is 1 - should NOT affect the output;
      max_count = 15;
      #850;
      // At this time, the output should be 73. Reset it to zero and give it the new max by setting run to 0
      run = 0;
      #20
      // Count up to 15 by setting run to 1
      run = 1;
      #50;
      // Change max count to > 99
      max_count = 118;
      #500
      run=0;
      #20
      run=1;
      // Should count to 99 and stop
      // *** NOTE*** You will need to simulate 3us to see the entire waveform
   end
endmodule
```

Problem 3: Simulation Waveforms. Include waveforms that demonstrate the functionality. *Use as many pages as needed.*

As the waveforms show, the count does not start until after the run signal goes up to 1 and it stops at the max_count and ignores any input until the run signal goes down to 0, it then resets to 0 and starts the count over until the new max_count is reached, if the max_count is greater to 99, it counts until 99 and then stops.