Assignment 3: Question 2: Converting 2421 code to Binary Coded Decimal BCD

2421-code numbers are weighted differently than BCD numbers.

2421 code is a self-complementary code. The 1's complement of a binary number can be obtained by replacing 0's with 1's and 1's with 0's. For self-complementary codes, the sum of a binary number and

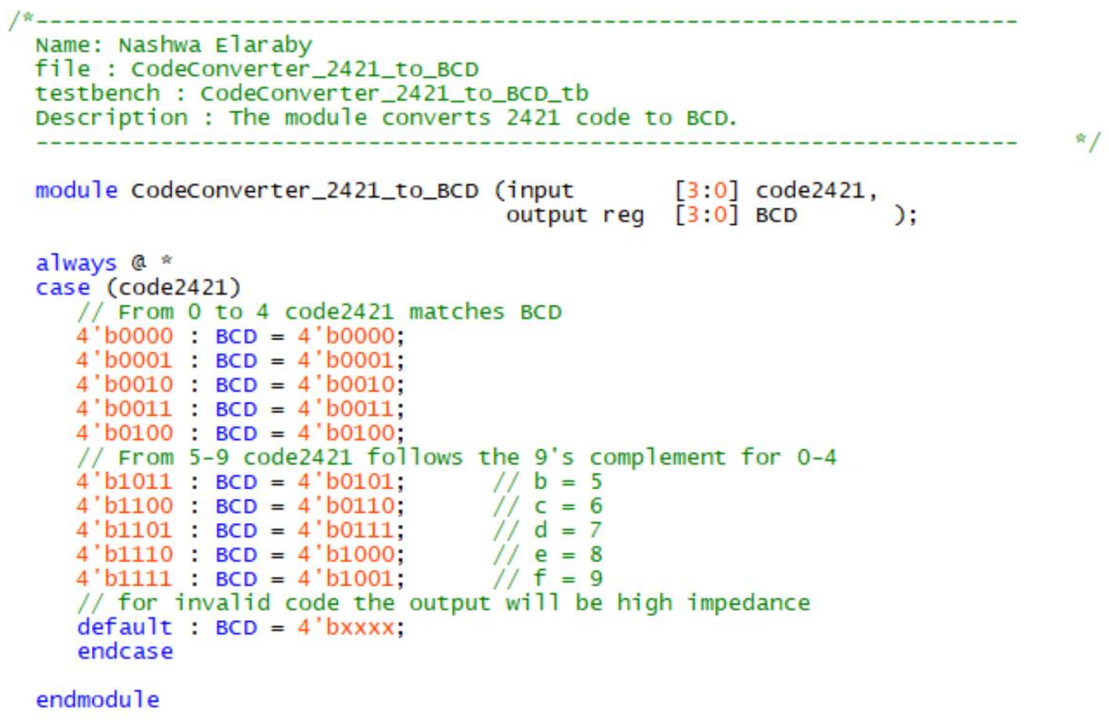
its complement is always equal to decimal 9. It can be observed that in the 2421 code is weighted differently than the binary code.

The code for decimal 9 is the complement of the code for decimal 0, the code for decimal 8 is the complement of the code for decimal 1, the code for decimal 7 is the complement of the code for decimal 2, the code for decimal 6 is the complement of the code for decimal 3, the code for decimal 5 is the complement of the code for decimal 4. These codes are called Reflective Codes. The bit-by-bit complement of the code creates the 9's complement of its value. They always add to 9. This is helpful in the subtraction and addition of decimal numbers. For switching between up and down counting, bit-by-bit complementing of the count is used.

The truth table showing the conversion from BCD 8421 to the weighted 2421 code.

Decimal value	BCD – 8421 weighted code	Weighted 2421 code
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	1011
6	0110	1100
7	0111	1101
8	1000	1110
9	1001	1111

Use a case structure in Verilog code to design the combinational logic for the code2421_2_BCD converter module. Assume that the outputs will only have digits from 0 to 9. Invalid 2421-codes can be considered "don't care" cases. Test your circuit using a testbench and combine your code and testing results in a pdf file. Hint: Remember that the case structure can only be used within an "always" block, and that only registers can be assigned values within the always block, hence you have to define the output of the conversion module as a reg.



```
/*-----------------------------------------------------------------------
  Name: Nashwa Elaraby
  file : CodeConverter_2421_to_BCD
  testbench : CodeConverter_2421_to_BCD_tb
  Description : The module converts 2421 code to BCD.
  -----------------------------------------------------------------------     */

  module CodeConverter_2421_to_BCD (input       [3:0] code2421,
                                    output reg  [3:0] BCD       );

  always @ *
  case (code2421)
     // From 0 to 4 code2421 matches BCD
     4'b0000 : BCD = 4'b0000;
     4'b0001 : BCD = 4'b0001;
     4'b0010 : BCD = 4'b0010;
     4'b0011 : BCD = 4'b0011;
     4'b0100 : BCD = 4'b0100;
     // From 5-9 code2421 follows the 9's complement for 0-4
     4'b1011 : BCD = 4'b0101;      // b = 5
     4'b1100 : BCD = 4'b0110;      // c = 6
     4'b1101 : BCD = 4'b0111;      // d = 7
     4'b1110 : BCD = 4'b1000;      // e = 8
     4'b1111 : BCD = 4'b1001;      // f = 9
     // for invalid code the output will be high impedance
     default : BCD = 4'bxxxx;
     endcase

  endmodule
```

Figure 1. Verilog code implementing the conversion from code 2421 to BCD using a case structure.

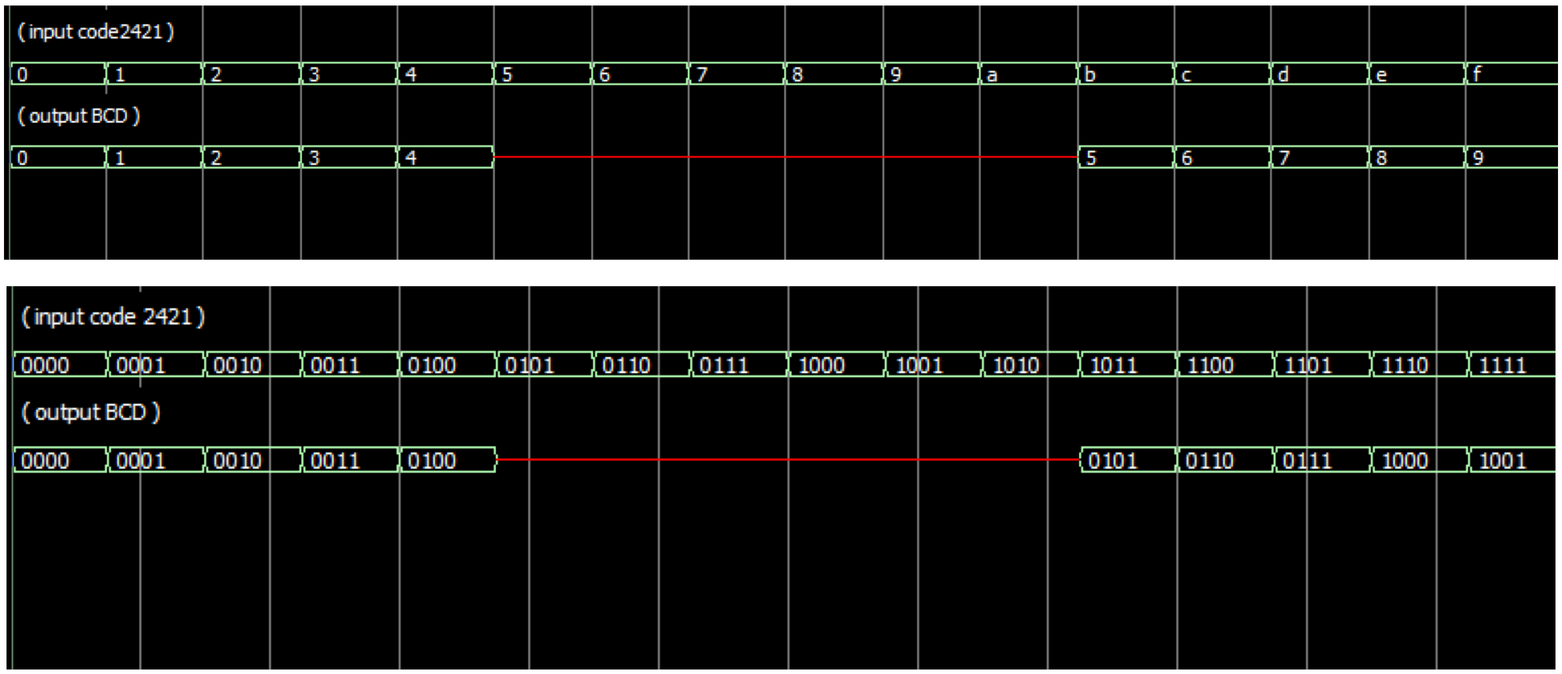
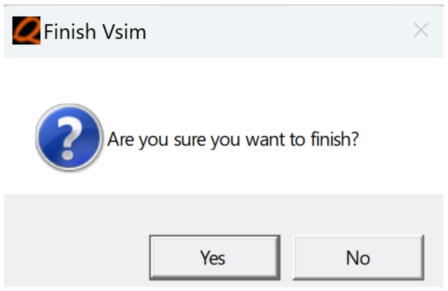


Figure 2. Simulation results waveform showing the conversion for all the possible combinations of 4 bits. Since 0101, 0110, 0111, 1000, 1001, and 1010 are not 2421 codes, the BCD output is showing an unknown signal at the output. A valid output is only seen for valid 2421 codes. The top figure shows the simulation results in hexadecimal format, and the bottom one is showing the code in binary.

```
module CodeConverter_2421_to_BCD_tb ();

reg  [3:0]   code2421;
wire [3:0]   BCD;

CodeConverter_2421_to_BCD UUT (code2421, BCD);

initial
begin
   code2421 = 4'b0000;
   forever
   begin
   #5         // Add time delay to allow for the conversion to be completed accounting for the propagation delay

   // Display the values sampled after 5 time units from changing the value of code2421.
   $display ("code2421 = %b = %h and BCD = %b = %h",code2421,code2421,BCD,BCD);

   #10 code2421 = code2421 + 1;
   end
end

initial
begin
#300 $finish;           // Simulation will stop after 300 time units
end

endmodule
```

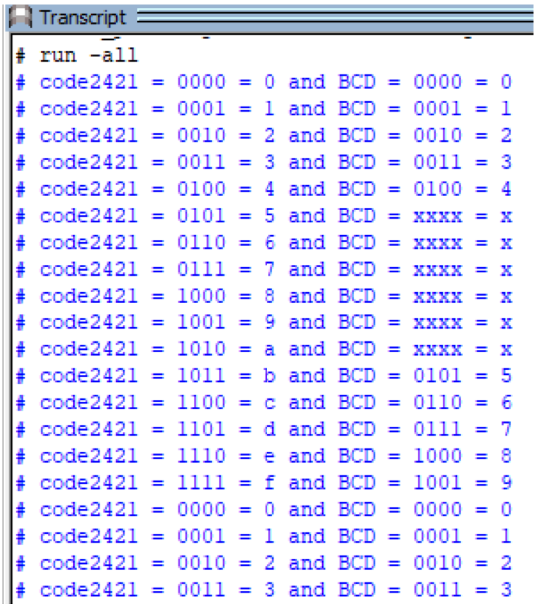
Figure 3. Testbench code. The testbench uses a counter to test all the possible combinations of a 4Bit input. This includes invalid code for the 2421 code.
The code also uses the **$display**. Unlike the $monitor, we specify exactly when we want to sample the values of our inputs and outputs. That's why we had to add some delay #5 to account for the propagation delay through the combinational logic to complete the conversion and have the correct output. If this delay is removed. The BCD displayed will correspond to the previous BCD value, as it did not have any time to update the logic.
The code is also showing the use of **$finish**. After the given time delay #300, the simulation will end. This will cause a pop up window to appear that will ask you if you want to finish. Hit **No** to keep Questa open and review your results. If you hit **Yes**, Questa will close. This command limits the amount of data collected in the simulation, which is helpful in this case since we have a **forever** loop.



Are you sure you want to finish?

Figure 4. Pop-up message when running the RTL simulation. Hit No.

Figure 5 Simulation results printed out in the transcripts window. This is easily matched to the truth-table of the code conversion.



```
# run -all
# code2421 = 0000 = 0 and BCD = 0000 = 0
# code2421 = 0001 = 1 and BCD = 0001 = 1
# code2421 = 0010 = 2 and BCD = 0010 = 2
# code2421 = 0011 = 3 and BCD = 0011 = 3
# code2421 = 0100 = 4 and BCD = 0100 = 4
# code2421 = 0101 = 5 and BCD = xxxx = x
# code2421 = 0110 = 6 and BCD = xxxx = x
# code2421 = 0111 = 7 and BCD = xxxx = x
# code2421 = 1000 = 8 and BCD = xxxx = x
# code2421 = 1001 = 9 and BCD = xxxx = x
# code2421 = 1010 = a and BCD = xxxx = x
# code2421 = 1011 = b and BCD = 0101 = 5
# code2421 = 1100 = c and BCD = 0110 = 6
# code2421 = 1101 = d and BCD = 0111 = 7
# code2421 = 1110 = e and BCD = 1000 = 8
# code2421 = 1111 = f and BCD = 1001 = 9
# code2421 = 0000 = 0 and BCD = 0000 = 0
# code2421 = 0001 = 1 and BCD = 0001 = 1
# code2421 = 0010 = 2 and BCD = 0010 = 2
# code2421 = 0011 = 3 and BCD = 0011 = 3
```