EE417 Advanced Digital Logic with Verilog HDL

L3 Assignment 2: Weighted2421ToBCDConverter

3. Behavioral Design Assignments

Due Jun 2 at 11:59pm **Points** 100 **Questions** 3 **Time Limit** None

Instructions

The main objective of the assignment is to practice and implement different methods for describing combinational logic in Verilog:

1. In this set of assignments, we will practice how to use the assign operator to define the logical expression for an output. In the structural design with basic primitive logic gates assignment, we instantiated AND, OR, NOT, and XOR gates and used interconnecting wires to implement the switching logic. Using the assign operator allows us to describe the hardware in a more compact code. This is implemented in question 1 converting a BCD to Xcess3 code. This example also introduces you to the Xcess code and covers an overview of the advantages of this code.

2. The assignment also covers the implementation of case structures to implement a hardware design while bypassing the K-map step for finding the SOP (Sum of Products) expressions. Question 2 introduces the students to the 2421 code and uses the case structure to implement the conversion.

3. Displaying 4-bit binary input on two seven-segment-displays covering numbers from 0 to 15. This design requires the implementation of comparators, multiplexers (using the conditional assignment) to have the correct numbers displayed.

For each design, you will create a testbench to verify the correct functionality of the design.

This is a group assignment (groups of 2 students). Each group will submit one document listing the contribution of each member.

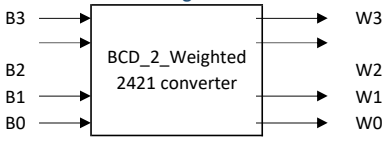
2421-code numbers are weighted differently than BCD numbers.
2421 code is a self complementary code. The 1's complement of a binary number can be obtained by replacing 0's with 1's and 1's with 0's. For self-complementary codes, the sum of a binary number and
its complement is always equal to decimal 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 complemen
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.

Truth Table for BCD to Weighted 2421 Code

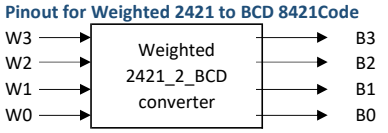
Decimal value	BCD – 8421 weighted code	Weighted 2421 code
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	1011
6	0110	1100
7	0111	1101
8	1000	1110
9	1001	1111

BDC	Weighted2421	Diff
0	0	0
1	1	0
2	2	0
3	3	0
4	4	0
5	11	-6
6	12	-6
7	13	-6
8	14	-6
9	15	-6

Pinout for BCD to Weighted 2421 Code



Pinout for Weighted 2421 to BCD 8421Code



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, and upload it here. ***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.***

Pattern Recognition

It is noticed that for all decimal values 4 and under that W=B

For decimal value 5, W=B+6

For decimal 6, W = single shift left of B, or W = B + 6

For decimal 7, W = B + 6

For decimal 8, W = B + 6

For decimal 9, W = B + 6

dec	B	W
<5	W	B
>=5	W-6	B+6

Module code

```
// Name:  Ron Kalin, Date: 5-30-24, Design: Lesson 3A2: Weighted2421ToBCD Converter
// Group: Ron Kalin/Lamin Jammeh _____
// Weighted 2421 to BCD Converter
```

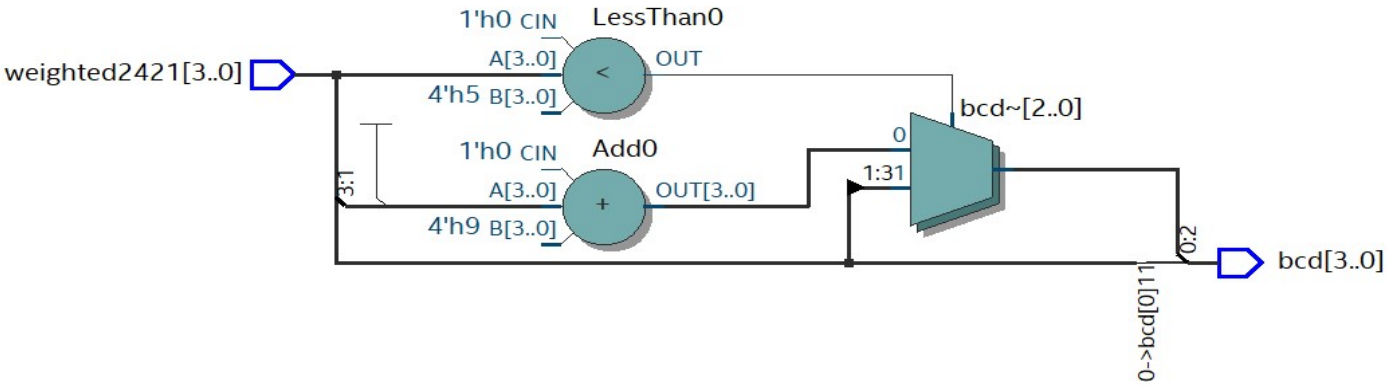
```
module Weighted2421ToBCDConverter (
 input [3:0] weighted2421, // Input in weighted 2421 format
 output reg [3:0] bcd ); // output in BCD format

   always @ ( ) //put input only in sesitivity list
     case (weighted2421)
        4'b0000: bcd = 4'b0000; // 0
        4'b0001: bcd = 4'b0001; // 1
        4'b0010: bcd = 4'b0010; // 2
        4'b0011: bcd = 4'b0011; // 3
        4'b0100: bcd = 4'b0100; // 4
        4'b1011: bcd = 4'b0101; // 5
        4'b1100: bcd = 4'b0110; // 6
        4'b1101: bcd = 4'b0111; // 7
        4'b1110: bcd = 4'b1000; // 8
        4'b1111: bcd = 4'b1001; // 9
        default: bcd = 4'b1111; // Default to unique value so invalid input code can be recognized
      endcase
// Alternative description is given below
//assign bcd = (weighted2421 < 5) ? weighted2421 : weighted2421 - 6;


endmodule
```

Division of Responsibility

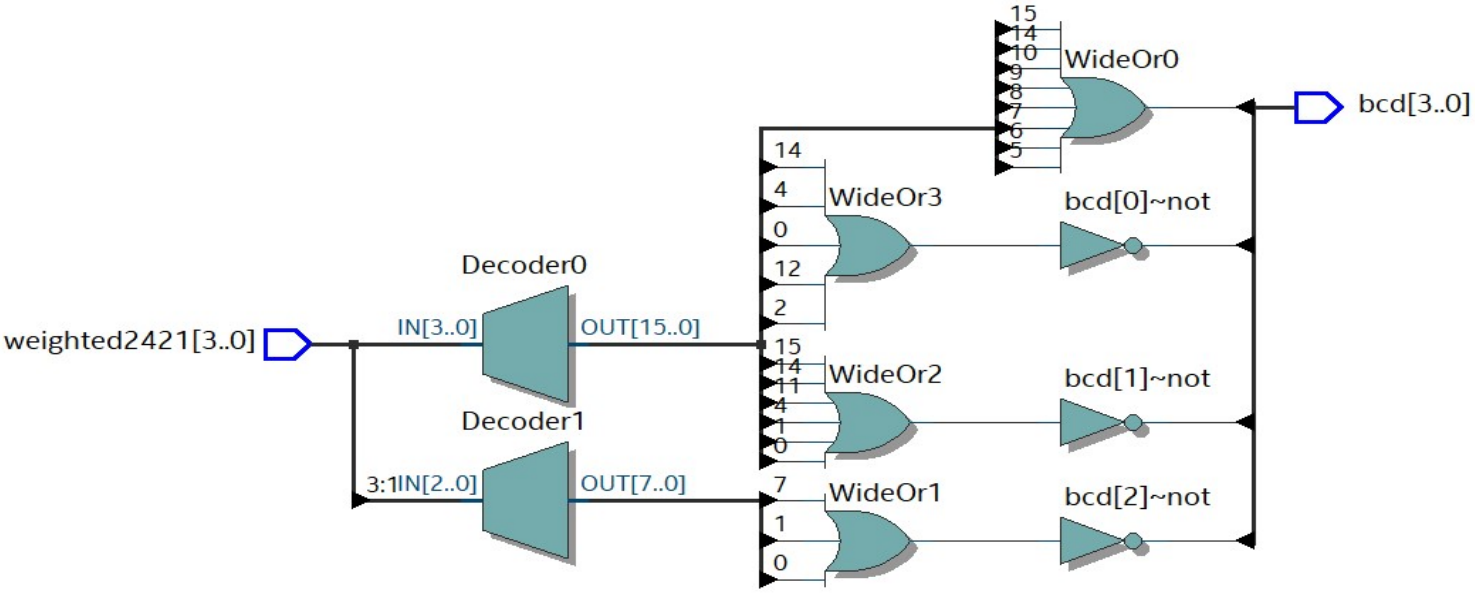
Assignment	Map/ Equations /Code	TestBench
1	Kalin	Jammeh
2	Kalin	Jammeh
3	?	?

Netlist RTL Viewer

Using Assign Statement



Using case statement



testbench Weighted2421toBCD_tb

```
/*-------------------------------------------------------------------------------
Name Lamin Jammeh
CLass: EE417 Summer 2024
Lesson 03 HW Question 1=2
Group: Ron Kalin/ Lamin Jammeh
Main design was done by Ron Kalin
TestBench was done by Lamin Jammeh
-------------------------------------------------------------------------------*/

//Step1 define a name for the test-bench
module Weighted2421ToBCDConverter_tb;

//Step2 define the inputs as registers and outputs as wires
reg [3:0] weighted2421;
wire [3:0] bcd;

//Step3 define the Unit under test UUT with all inputs and outputs
Weighted2421ToBCDConverter UUT (
                                .weighted2421(weighted2421),
                                .bcd(bcd)
                               );

//Step4 open an initial block and define all the possible input combination for the weighted2421 to BCD from 0-9
initial
   begin
            weighted2421 = 4'b0000;              //define the input as 4 bits with all zero binary at the start
       #100 weighted2421 = 4'b0001;
       #100 weighted2421 = 4'b0010;
       #100 weighted2421 = 4'b0011;
       #100 weighted2421 = 4'b0100;
       #100 weighted2421 = 4'b1011;
       #100 weighted2421 = 4'b1100;
       #100 weighted2421 = 4'b1101;
       #100 weighted2421 = 4'b1110;
       #100 weighted2421 = 4'b1111;
       #100;
    end
initial
   begin
          $monitor("Weighted_2421: %b            BCD_8421: %b", weighted2421, bcd);
    end
endmodule
```

Simulation Results