FPGA Application Week 4 Homework

CYEE 10828241 Chen Da-Chuan

March 18, 2023

Contents

List of Figures

List of Tables

1 4 bits calculator with add, subtract, multiply and divide functions

1.1 Objective

Code a 4 bits calculator with add, subtract, multiply and divide functions.

1.2 Operation

This code utilize the 10 switches on board to set the 2 input numbers ranging from -15 to $+15$ and the 2 keys to set the mode of the calculator. The 4 bits calculator will calculate the result and display it on the 7-segment display. The truth table of the 4 bits calculator is shown in Table 1.

Table 1: The truth table of the 4 bits calculator. (S: sign, N: number, X: don't care)

KEY0	KEY1	MODE	INPUT RANGE	OUTPUT RANGE	DISPLAY
unpressed	unpressed	add	± 15	± 30	XXXSNN
pressed	unpressed	subtract	± 15	± 30	XXXSNN
unpressed	pressed	multiply	± 15	± 225	XXSNNN
pressed	pressed	divide	± 15	Q:± 15, R:± 15	SNNXNN

```
28  //=====================================================
29  //  REG/WIRE declarations
30  //=====================================================
31
32  // this is signed 4bits adder/substractor/multiplier/divider
33
34  wire signed [4:0] numA, numB;     // for calculation
35  wire [3:0] numA_ten, numA_digit; // for display
36  wire [3:0] numB_ten, numB_digit;
37
38  wire signed [5:0] num_Add, num_Sub;
39  wire [3:0] num_Add_ten, num_Add_digit;
40  wire [3:0] num_Sub_ten, num_Sub_digit;
41
42  wire signed [9:0] num_Mul;
43  wire [3:0] num_Mul_hundred, num_Mul_ten, num_Mul_digit;
44
45  wire signed [5:0] num_Div_Q, num_Div_R;
46  wire [3:0] num_Div_Q_ten, num_Div_Q_digit;
47  wire [3:0] num_Div_R_ten, num_Div_R_digit;
48
49  reg [3:0] digit0, digit1, digit2, digit3, digit4, digit5;
```

Figure 1: The code of the 4 bits calculator.

1.3 Code

1. Line34-36(Figure 1): Set the input wire of number A and B, and its displaying digits.
2. Line38-40(Figure 1): Set the wire of adding and subtraction, and its displaying digits.
3. Line42-43(Figure 1): Set the wire of multiplication, and its displaying digits.
4. Line45-47(Figure 1): Set the wire of division, and its displaying digits.
5. Line61-66(Figure 2): Convert 4 bits input binary numbers to decimal numbers.
6. Line68-72(Figure 2): Calculate all four functions.
7. Line74-84(Figure 2): Convert calculated decimal numbers to binary numbers.
8. Line88-96(Figure 3): Set digits of the result of $A + B$.
9. Line97-105(Figure 3): Set digits of the result of $A - B$.
10. Line106-114(Figure 3): Set digits of the result of $A \times B$.
11. Line115-135(Figure 3): Set digits of the result of $A \div B$, and display 6 dash lines if $B = 0$.
12. Line138-146(Figure 4): Use provided 7-segment display to display the result.

1.4 Execution Result

This is the link to the execution result video: `https://youtu.be/PYXb0r5au9w`. The table of values being calculated is shown in Table 2.

```
61  assign numA = SW[4]?32-SW[3:0]:SW[3:0];
62  assign numB = SW[9]?32-SW[8:5]:SW[9:5];
63  assign numA_digit = SW[3:0]%10;
64  assign numA_ten   = SW[3:0]/10;
65  assign numB_digit = SW[8:5]%10;
66  assign numB_ten   = SW[8:5]/10;
67
68  assign num_Add = numB + numA;
69  assign num_Sub = numB - numA;
70  assign num_Mul = numB * numA;
71  assign num_Div_Q = numB / numA;
72  assign num_Div_R = numB % numA;
73
74  assign num_Add_ten     = num_Add[5]?(32-num_Add[4:0])/10:num_Add/10;
75  assign num_Add_digit   = num_Add[5]?(32-num_Add[4:0])%10:num_Add%10;
76  assign num_Sub_ten     = num_Sub[5]?(32-num_Sub[4:0])/10:num_Sub/10;
77  assign num_Sub_digit   = num_Sub[5]?(32-num_Sub[4:0])%10:num_Sub%10;
78  assign num_Mul_hundred = num_Mul[9]?(512-num_Mul[8:0])/100:num_Mul/100;
79  assign num_Mul_ten     = num_Mul[9]?((512-num_Mul[8:0])%100)/10:(num_Mul%100)/10;
80  assign num_Mul_digit   = num_Mul[9]?((512-num_Mul[8:0])%100)%10:(num_Mul%100)%10;
81  assign num_Div_Q_ten   = num_Div_Q[5]?(32-num_Div_Q[4:0])/10:num_Div_Q/10;
82  assign num_Div_Q_digit = num_Div_Q[5]?(32-num_Div_Q[4:0])%10:num_Div_Q%10;
83  assign num_Div_R_ten   = num_Div_R[5]?(32-num_Div_R[4:0])/10:num_Div_R/10;
84  assign num_Div_R_digit = num_Div_R[5]?(32-num_Div_R[4:0])%10:num_Div_R%10;
85
```

Figure 2: The code of the 4 bits calculator.

Table 2: The table of values being calculated.

No.	**binary A**	**binary B**	**decimal A**	**decimal B**	**A+B**	**A−B**	**A×B**	**Q:A÷B**	**R:A÷B**
1	00000	00000	+00	+00	+00	+00	+000	—	—
2	01111	01111	+15	+15	+30	+00	+225	+01	+00
3	11111	11111	-15	-15	-30	-00	+225	+01	+00
4	01111	11111	+15	-15	+00	+30	-225	-01	+00
5	11111	01111	-15	+15	+00	-30	-225	-01	+00
6	00101	00001	+05	+01	+06	+04	+005	+05	+00
7	00001	00101	+01	+05	+06	-04	+005	+00	+01

```
 86  always @(KEY[0] or KEY[1])
 87  begin
 88      if (KEY[0] && KEY[1])       // input mode: not pressed key 0,1
 89      begin
 90          digit5 = 4'he;
 91          digit4 = 4'he;
 92          digit3 = 4'he;
 93          digit2 = num_Add[5]?4'hf:4'he;
 94          digit1 = num_Add_ten;
 95          digit0 = num_Add_digit;
 96      end
 97      else if (!KEY[0] && KEY[1]) // add mode: pressed key 0
 98      begin
 99          digit5 = 4'he;
100          digit4 = 4'he;
101          digit3 = 4'he;
102          digit2 = num_Sub[5]?4'hf:4'he;
103          digit1 = num_Sub_ten;
104          digit0 = num_Sub_digit;
105      end
106      else if (KEY[0] && !KEY[1]) // sub mode: pressed key 1
107      begin
108          digit5 = 4'he;
109          digit4 = 4'he;
110          digit3 = num_Mul[9]?4'hf:4'he;
111          digit2 = num_Mul_hundred;
112          digit1 = num_Mul_ten;
113          digit0 = num_Mul_digit;
114      end
115      else if (!KEY[0] && !KEY[1]) // div mode: pressed key 0,1
116      begin
117          if (numA != 0)
118          begin
119              digit5 = num_Div_Q[5]?4'hf:4'he;
120              digit4 = num_Div_Q_ten;
121              digit3 = num_Div_Q_digit;
122              digit2 = 4'he;
123              digit1 = num_Div_R_ten;
124              digit0 = num_Div_R_digit;
125          end
126          else
127          begin
128              digit5 = 4'hf;
129              digit4 = 4'hf;
130              digit3 = 4'hf;
131              digit2 = 4'hf;
132              digit1 = 4'hf;
133              digit0 = 4'hf;
134          end
135      end
136  end
```

Figure 3: The code of the 4 bits calculator.

```
138  SEG7_LUT_6 u_seg(
139      .oSEG0(HEX0),
140      .oSEG1(HEX1),
141      .oSEG2(HEX2),
142      .oSEG3(HEX3),
143      .oSEG4(HEX4),
144      .oSEG5(HEX5),
145      .iDIG ({ digit5, digit4, digit3, digit2, digit1, digit0})
146  );
147
148  endmodule
```

Figure 4: The code of the 4 bits calculator.