Design Block 1

The purpose of Design Block 1 is to activate the LEDs when the corresponding switch is down, and when KEY0 is pressed and held the state of the LEDs toggles. The birthday of one team member is displayed by default, and when KEY1 is pressed and held, the birthday of the other team member is displayed.

Each LED is assigned to its corresponding SW XOR with KEY0. KEYs are Logic 0 when pressed and held, and Logic 1 when KEYs are not being pressed. SWs are Logic 1 when pushed up, and Logic 0 when pushed down. Assigning LEDs to KEY0 XOR SW ensures that LEDs are activated when their corresponding SWs are down, and the state of the LEDs are toggled when KEY0 is pressed and held.

Since the birthdays are displayed in MMDDYY format, all six HEX displays are being utilized for this design block. Six 8-bit registers are declared to hold all six values in the MMDDYY format. If KEY1 is not pressed, all six registered are each assigned an 8-bit integer to display, corresponding to a team member's birthday. A SevenSeg module is called six times, and accepts the HEX display and integer value to display as inputs. SevenSeg converts the integer input into an 8-bit binary value that accurately displays the integer in seven segment display format. If KEY1 is pressed and held, all six registered are each assigned an 8-bit integer to display that corresponds to the other team member's birthday.

Lab1.v is at the top of the hierarchy for Design Block 1. Design Block 1 also contains Birthday.v and SevenSeg.v. Lab1.v calls Birthday.v, and Birthday.v calls SevenSeg.v. The testbench operations test if the LED values correspond to the opposite value of the SWs when KEY0 is both pressed and unpressed. In addition to this, the testbench tests if the correct HEX birthday values are being displayed when KEY1 is both pressed and unpressed.

The demonstration of Design Block 1 works properly.

Design Block 2

The purpose of Design Block 2 is to create a 4-bit add/subtract circuit. SW[7:4] is input1, and SW[3:0] is input2. Input1 and input2 represent 2's complement values. Since the 2's complement sum must fit in 4-bits, there will be arithmetic overflow. Ripple carry logic will be utilized with multiple full adder circuits. The absolute value of input1 is displayed on HEX4, and its sign is displayed on HEX5. The absolute value of input2 is displayed on HEX2, and its sign is displayed on HEX3. The absolute value of input1 + input2 is displayed on HEX0, and when KEY0 is pressed and held, the absolute value of input1 - input2 is displayed on HEX0. The sign is always displayed on HEX1. HEX0 and HEX1 display "0F" if arithmetic overflow occurs.

Lab1DB2.v is at the top of the hierarchy for Design Block 2. Design Block 2 also contains Adder.v, RippleCarry.v, and SevenSeg.v. Lab1DB2 deactivates the LEDs, and calls the RippleCarry.v module. The absolute value and sign of input1 and input2 are determined, and SevenSeg.v is called to display these values. Wires are declared for input1, input2, carry, and output. The carry is a 5-bit wire, and input1, input2, and output are 4-bit wires. The input wires are assigned to the four values of input1 and input2. The carry wire "c" contains carry in and carry out. Carry in, c[0], is assigned not KEY0. If KEY0 is pressed and held, c[0] is value 1 and the two numbers are subtracted. When KEY0 is 0, the two inputs are added together. The Adder.v module is called four times for the respective bits of the inputs and the carry-out from the previous call. The Adder.v logic is one part of a ripple carry adder, and contains a carry-in bit to determine addition or subtraction. The absolute value and sign are fed into SevenSeg.v from the four output bits of the Adder. Arithmetic overflow is determined by the last carry-in input and the carry out. c[3] XOR c[4] is implemented, and overflow occurs in the 2's complement addition or subtraction if only one of the values is 1.

The testbench operations tests different sets of inputs, and their expected outputs when KEY0 is pressed and unpressed. The testbench also tests if the correct values are being outputted on the HEX displays.

The demonstration of Design Block 2 works properly.

Design Block 3

The purpose of Design Block 3 is to compare the values entered by input1 and input2. Input1 corresponds to SW[7:4], and input2 corresponds to SW[3:0]. If input1 = input2, LEDR[2] is activated. If input1 > input2, LEDR[1] is activated. If input1 < input2, LEDR[0] is activated. When SW[9:8] = 2'b11, the two input values are interpreted as 2's complement values. When SW[9:8] = 2'b10, the two input values are interpreted as unsigned values. The absolute value of input1 is displayed on HEX0, and its sign is displayed on HEX1. The absolute value of input2 is displayed on HEX4, and its sign is displayed on HEX5.

SW[9:8], input1, and input2 are fed into a Compare.v module. Four 8-bit registers are declared to hold the absolute value of input1 and sign of input1, and the absolute value of input2 and sign of inpu2. If SW[9:8] = 2'b11, and the third bit of input1 and/or input2 equals 1, the sign of input1 and/or input2 will be assigned an 8-bit integer value that SevenSeg.v will translate to a negative symbol on the HEX display. If SW[9:8] = 2'b10, the sign of input1 and input2 will be assigned an 8-bit integer value that SevenSeg.v will translate to a blank HEX display. If the MSB is 1, the 2's complement value is negative. The 4-bit input1 and input2 values are translated to 8-bit integer values that will be fed as inputs into SevenSeg.v. The signs and absolute values of input1 and input2 are used to determine the proper LED activation. If one input is positive, and the other input is negative, the positive input is greater than the negative input. If both inputs are positive, the integer values of each input are compared and the proper LED is activated. If both inputs are negative, the smaller integer value is the greater input, the larger integer value is the smaller input, and the proper LED is activated.

Lab1DB3.v is at the top of the hierarchy for Design Block 3. Design Block 3 also contains Compare.v and SevenSeg.v. Lab1DB3.v calls Compare.v, and Compare.v calls SevenSeg.v. The testbench operations test if the correct HEX signs and values are being displayed on the HEX seven segment displays. The testbench operations also evaluate if input1 and input2 comparisons are accurate and the appropriate LED is activated.

The demonstration of Design Block 3 works properly.

Integration Design Block

The purpose of Integration is to combine the functionality of Design Block 1, Design Block 2, and Design Block 3. When SW[9:8] = 2'b00, Design Block 1 is implemented. When SW[9:8] = 2'b01, Design Block 2 is implemented. When SW[9:8] = 2'b1x, Design Block 3 is implemented. A single programming file is made that allows the board to demonstrate all three Design Blocks. LEDR[9:8] is utilized to indicate the position of SW[9:8].

Integration functions as a multiplexer (case statement), and decides which Design Block should be implemented on the HEX displays and LEDs based on the input. The three Design Blocks are instantiated with different wire names for outputs, and the same names for inputs. The output of the multiplexer is based on the position of SW[9:8], and then it runs through the cases and set registers, that are connected to the board output, equal to the output wires of the desired design block.

Testbenches are not required for the Integration Design Block. The demonstration of the Integration Design Block works properly.