Lab Code [10 points]
Filename: ChipInterface.sv
AndrewID: jtbell

```
module ChipInterface
  (output logic [6:0] HEX7, HEX6, // magic_constant
  HEX5, HEX4, HEX3, HEX2, HEX1, HEX0,
   output logic [7:0] LEDG,
   input logic [17:0] SW,
   input logic [3:0] KEY,
   input logic CLOCK_50); // needed for enter_9_bcd
  logic [3:0] num1, num2, num3,
  num4, num5, num6,
  num7, num8, num9;
  logic [7:0] magic_constant;
  logic it_is_magic;
  enter_9_bcd e(.entry(SW[3:0]),
  .selector(SW[7:4]),
  .enableL(KEY[0]),
  .zeroL(KEY[2]),
  .set_defaultL(KEY[1]),
  .clock(CLOCK_50),
  .*);
  IsMagic im(.*);
  // Your code here.
  // Output it_is_magic to all 8 bits of LEDG

  assign LEDG = (it_is_magic==1)? 8'b1111_1111: 8'b0000_0000;
  BCDtoSevenSegment seg1 (.bcd({magic_constant[3],magic_constant[2]
  ,magic_constant[1],magic_constant[0]}),.segment(HEX6));
  BCDtoSevenSegment seg2 (.bcd({magic_constant[7],magic_constant[6]
  ,magic_constant[5],magic_constant[4]}),.segment(HEX7));

  // Display magic_constant on the 7 segment display
endmodule : ChipInterface
```

```
Lab Code [10 points]
Filename: IsMagicFinal.sv
AndrewID: jtbell
```

```
`default_nettype none
//add two BCD #'s
module BCDOneDigitAdd
  (input logic [3:0] A, B,
  input logic Cin,
  output logic [3:0] Sum,
  output logic Cout);

  logic [4:0] Sum_temp;

  assign Sum_temp = A + B + Cin;
  assign Cout = (Sum_temp >= 10) ? 1 : 0;
  assign Sum = (Sum_temp >= 10) ? Sum_temp-10 : Sum_temp[3:0];
endmodule: BCDOneDigitAdd

//test BCDAdd one digit
module BCDOneDigitAdd_test();
  logic [3:0] A,B,Sum;
  logic Cin,Cout;
  BCDOneDigitAdd run(.*);
   initial begin
    $monitor($time, "A = %b, B= %b,Sum = %b,Cin = %b,Cout=%b",
    A, B,Sum,Cin,Cout);
    Cin=0;
    A=4'b0001;
    B=4'b0001;
    #10;
    A=4'b0010;
    B=4'b0001;
    #10;
    A=4'b1000;
    B=4'b1000;
    #10;
    A=4'b1111;
    B=4'b1111;
    #10 $finish;
  end
endmodule: BCDOneDigitAdd_test

// Adds two BCD numbers
module Adders
  (input logic [7:0]A,B,
  output logic [7:0] Sum);

  logic [3:0] Aone,Aten,Bone,Bten;
  logic [3:0] sum1,sum2;
  logic carry1,carry2;

  assign Bone = B[3:0];
  assign Bten = B[7:4];
  assign Aone = A[3:0];
  assign Aten = A[7:4];

  BCDOneDigitAdd B1(.A(Aone),.B(Bone),.Cin(1'b0),.Sum(sum1),.Cout(carry1));
  BCDOneDigitAdd B2(.A(Aten),.B(Bten),.Cin(carry1),.Sum(sum2),.Cout(carry2));

  assign Sum = {sum2,sum1};

endmodule: Adders

//test the adder
module Adders_test();
  logic [7:0] A,B,Sum;
  Adders run(.*);
  initial begin
    $monitor($time, "A = %b, B= %b,Sum = %b",
    A, B,Sum);
    A=8'b0000_0001;
    B=8'b0000_0001;
```

```
70      #10;
71      A=8'b0001_0001;
72      B=8'b0000_0001;
73      #10;
74      A=8'b0001_0001;
75      B=8'b0001_0001;
76      #10;
77      A=8'b0001_1001;
78      B=8'b0001_1001;
79    #10 $finish;
80    end
81  endmodule: Adders_test
82
83  // Adds a row of bcd #'s
84  module addRow
85    (input logic [3:0]A,B,C,
86    output logic [7:0]sum);
87
88    logic [7:0] sum1;
89
90    Adders A1(.A(A),.B(B),.Sum(sum1));
91    Adders A2(.A(sum1),.B(C),.Sum(sum));
92  endmodule: addRow
93
94  //Test addrow
95  module addRow_test();
96    logic [3:0] A,B,C;
97    logic [7:0] sum;
98    addRow ar(.*);
99    initial begin
100
101     $monitor($time, "A = %b, B= %b,C = %b,sum = %b",
102     A, B,C,sum);
103
104     A=4'b0001;
105     B=4'b0001;
106     C=4'b0001;
107     #10;
108     A=4'b1001;
109     B=4'b1001;
110     C=4'b1001;
111     #10;
112     A=4'b1001;
113     B=4'b0001;
114     C=4'b0001;
115     #10;
116     A=4'b1111;
117     B=4'b1111;
118     C=4'b1111;
119     #10 $finish;
120   end
121 endmodule: addRow_test
122
123 //These two modules are instaniating the comparator and using the
124 //comparator
125 module Comparator
126   (input logic [3:0] A,
127   input logic [3:0] B,
128   output logic AeqB);
129
130   assign AeqB = (A==B);
131 endmodule: Comparator
132
133 //Compares a Two digit BCD number using comparator
134 module ComparatorTwoDigit
135  (input logic [7:0] A,
136   input logic [7:0] B,
137   output logic AeqB);
138
139   logic onesEqual,tensEqual;
140
```

```
141   Comparator c1(.A(A[3:0]),.B(B[3:0]),.AeqB(onesEqual));
142   Comparator c2(.A(A[7:4]),.B(B[7:4]),.AeqB(tensEqual));
143   assign AeqB = onesEqual & tensEqual;
144 endmodule: ComparatorTwoDigit
145
146 //2 By 4 decoder
147 module decoder
148     (input logic [3:0] in,
149      output logic [15:0] out);
150      always_comb begin
151        case(in)
152        4'd1: out = 16'd1;
153        4'd2: out = 16'd2;
154        4'd3: out = 16'd4;
155        4'd4: out = 16'd8;
156        4'd5: out = 16'd16;
157        4'd6: out = 16'd32;
158        4'd7: out = 16'd64;
159        4'd8: out = 16'd128;
160        4'd9: out = 16'd256;
161        default: out = 16'd0;
162       endcase
163      end
164 endmodule : decoder
165
166 //2 By 4 Deocer test
167 module decoder_test();
168   logic [3:0] in;
169   logic [15:0] out;
170   decoder d(.*);
171   initial begin
172     $monitor($time, "in = %b,out%b",
173                              in,out);
174     in = 4'b001;
175     #10;
176     in = 4'b0011;
177     #10;
178     in = 4'b0101;
179     #10;
180     in = 4'b1001;
181     #10;
182     #10 $finish;
183   end
184
185 endmodule: decoder_test
186
187 //outputs 0 if atleast 1 number is > 9
188 module isGreaterThan9
189   (input logic [3:0] num1,num2,num3,
190   input logic [3:0] num4,num5,num6,
191   input logic [3:0] num7,num8,num9,
192   output logic greaterThanNine);
193
194   logic a,b,c,d,e,f,g,h,i;
195
196   assign a = num1 <= 9;
197   assign b = num2 <= 9;
198   assign c = num3 <= 9;
199   assign d = num4 <= 9;
200   assign e = num5 <= 9;
201   assign f = num6 <= 9;
202   assign g = num7 <= 9;
203   assign h = num8 <= 9;
204   assign i = num9 <= 9;
205   assign greaterThanNine = a & b & c & d & e & f & g & h & i;
206 endmodule : isGreaterThan9
207
208 //test for greaterthan9module
209 module isGreaterThan9_test();
210   logic [3:0] num1,num2,num3,num4,num5,num6,num7,num8,num9;
211   logic greaterThanNine;
```

```
212   isGreaterThan9 igt(.*);
213   initial begin
214     $monitor($time, "num1=%b,num2=%b,num3=%b,num4=%b,num5=%b,num6=%b,num7=%b,\
215     num8=%b,num9=%b,greaterThanNine = %b",
216     num1,num2,num3,num4,num5,num6,num7,num8,num9,greaterThanNine);
217     num1 = 4'b0001;
218     num2 = 4'b0010;
219     num3 = 4'b0100;
220     num4 = 4'b1000;
221     num5 = 4'b1001;
222     num6 = 4'b1011;
223     num7 = 4'b1010;
224     num8 = 4'b1001;
225     num9 = 4'b1111;
226     #10;
227     num1 = 4'b0001;
228     num2 = 4'b0010;
229     num3 = 4'b0100;
230     num4 = 4'b1000;
231     num5 = 4'b1001;
232     num6 = 4'b1001;
233     num7 = 4'b1000;
234     num8 = 4'b1001;
235     num9 = 4'b1001;
236     #10 $finish;
237   end
238 endmodule:isGreaterThan9_test
239
240 //converts bcd to seven segment display
241 module BCDtoSevenSegment
242   (input logic [3:0] bcd,
243   output logic [6:0] segment);
244   always_comb begin
245     case(bcd)
246       4'd0: segment = 7'b000_0001;
247       4'd1: segment = 7'b111_1001;
248       4'd2: segment = 7'b010_0100;
249       4'd3: segment = 7'b011_0000;
250       4'd4: segment = 7'b001_1001;
251       4'd5: segment = 7'b001_0010;
252       4'd6: segment = 7'b000_0010;
253       4'd7: segment = 7'b111_1000;
254       4'd8: segment = 7'b000_0000;
255       4'd9: segment = 7'b001_1000;
256       4'd10: segment = 7'b000_1000;
257       4'd11: segment = 7'b000_0011;
258       4'd12: segment = 7'b100_0110;
259       4'd13: segment = 7'b010_0001;
260       4'd14: segment = 7'b000_0110;
261       4'd15: segment = 7'b000_1110;
262       default: segment = 7'b1111111;
263     endcase
264   end
265 endmodule : BCDtoSevenSegment
266
267 //This module checks for uniqueness and
268 //also checks if a number is greater than 9
269 module checkValue
270   (input logic [3:0] num1,num2,num3,
271   input logic [3:0] num4,num5,num6,
272   input logic [3:0] num7,num8,num9,
273   output logic isMagic);
274
275   logic [15:0]out1,out2,out3,out4,out5,out6,out7,out8,out9;
276
277   logic greaterThanNine;
278   logic [15:0] combinedOut;
279   isGreaterThan9 great(.num1(num1),.num2(num2),.num3(num3),.num4(num4),
280   .num5(num5),.num6(num6),.num7(num7),.num8(num8),.num9(num9),.greaterThanNine
281   (greaterThanNine));
282
```

```
283   decoder d1 (.in(num1),.out(out1));
284   decoder d2 (.in(num2),.out(out2));
285   decoder d3 (.in(num3),.out(out3));
286   decoder d4 (.in(num4),.out(out4));
287   decoder d5 (.in(num5),.out(out5));
288   decoder d6 (.in(num6),.out(out6));
289   decoder d7 (.in(num7),.out(out7));
290   decoder d8 (.in(num8),.out(out8));
291   decoder d9 (.in(num9),.out(out9));
292
293   assign combinedOut = (out1 | out2 | out3 | out4 | out5 | out6 | out7 |
294   out8 | out9);
295
296   always_comb begin
297     if((combinedOut != 9'b1111_11111) || greaterThanNine==0) begin
298       isMagic = 1'b0;
299     end else begin
300       isMagic = 1'b1;
301       end
302     end
303 endmodule : checkValue
304
305 //test for checkValue
306 module checkValue_test();
307   logic [3:0] num1,num2,num3,num4,num5,num6,num7,num8,num9;
308   logic isMagic;
309   logic [15:0] combinedOut;
310   checkValue cv(.*);
311
312   initial begin
313     $monitor($time, "num1=%b,num2=%b,num3=%b,num4=%b,num5=%b,num6=%b,num7=%b,\
314     num8=%b,num9=%b,combinedOut = %b, isMagic = %b",
315     num1,num2,num3,num4,num5,num6,num7,num8,num9,combinedOut,isMagic);
316     num1 = 4'b0001;
317     num2 = 4'b0010;
318     num3 = 4'b0011;
319     num4 = 4'b0100;
320     num5 = 4'b0101;
321     num6 = 4'b0110;
322     num7 = 4'b0111;
323     num8 = 4'b1000;
324     num9 = 4'b1001;
325     #10;
326     num1 = 4'b0011;
327     num2 = 4'b0010;
328     num3 = 4'b0111;
329     num4 = 4'b0100;
330     num5 = 4'b1001;
331     num6 = 4'b0110;
332     num7 = 4'b001;
333     num8 = 4'b1000;
334     num9 = 4'b0101;
335     #10;
336     num1 = 4'b0010;
337     num2 = 4'b0010;
338     num3 = 4'b0110;
339     num4 = 4'b0100;
340     num5 = 4'b1001;
341     num6 = 4'b0110;
342     num7 = 4'b001;
343     num8 = 4'b1000;
344     num9 = 4'b0101;
345     #10;
346     num1 = 4'b0011;
347     num2 = 4'b0010;
348     num3 = 4'b0111;
349     num4 = 4'b0100;
350     num5 = 4'b1011;
351     num6 = 4'b0110;
352     num7 = 4'b001;
353     num8 = 4'b1000;
```

```
    num9 = 4'b0101;
    #10 $finish;
  end
endmodule: checkValue_test

//Checks if all the sums are equal to one another
module checkSums
  (input logic [3:0] a,b,c,d,e,f,g,h,i,
  output logic [7:0] sum);
  logic [7:0] sumRow1,sumRow2,sumRow3,sumCol1,sumCol2,sumCol3;
  logic [7:0] sumDiag1,sumDiag2;
  logic compare1,compare2,compare3,compare4,compare5,compare6;
  logic compare7,compare8;

  ComparatorTwoDigit CR1(.A(sumRow1),.B(sumRow2),.AeqB(compare1));
  ComparatorTwoDigit CR2(.A(sumRow1),.B(sumRow3),.AeqB(compare2));
  ComparatorTwoDigit CR3(.A(sumRow1),.B(sumCol1),.AeqB(compare3));
  ComparatorTwoDigit CR4(.A(sumRow1),.B(sumCol2),.AeqB(compare4));
  ComparatorTwoDigit CR5(.A(sumRow1),.B(sumCol3),.AeqB(compare5));
  ComparatorTwoDigit CR6(.A(sumRow1),.B(sumDiag1),.AeqB(compare6));
  ComparatorTwoDigit CR7(.A(sumRow1),.B(sumDiag2),.AeqB(compare7));

  //This block of code gets all rows and cols and diagnols then compares
  //them all.
  addRow r1(.A(a),.B(b),.C(c),.sum(sumRow1));
  addRow r2(.A(d),.B(e),.C(f),.sum(sumRow2));
  addRow r3(.A(g),.B(h),.C(i),.sum(sumRow3));
  addRow c1(.A(a),.B(d),.C(g),.sum(sumCol1));
  addRow c2(.A(b),.B(e),.C(h),.sum(sumCol2));
  addRow c3(.A(c),.B(f),.C(i),.sum(sumCol3));
  addRow d1(.A(a),.B(e),.C(i),.sum(sumDiag1));
  addRow d2(.A(c),.B(e),.C(g),.sum(sumDiag2));
  assign sum = ((compare1 & compare2 & compare3 & compare4 & compare5 &
  compare6 & compare7) ? sumRow1 : 8'b0000_0000);
endmodule: checkSums

//test for check sums
module checkSums_test();
  logic [3:0] a,b,c,d,e,f,g,h,i;
  logic [7:0] sum;
  checkSums cs(.*);
  initial begin
    $monitor($time, "a = %b, b= %b,c = %b,d = %b, e= %b,f = %b,g = %b, h= %b,\
    i = %b,sum = %b", a, b,c,d,e,f,g,h,i,sum);
    a = 4'b1000;
    b = 4'b0001;
    c = 4'b0110;
    d = 4'b0011;
    e = 4'b0101;
    f = 4'b0111;
    g = 4'b0100;
    h = 4'b1001;
    i = 4'b0010;
    #10;
    a = 4'b0011;
    b = 4'b0010;
    c = 4'b0111;
    d = 4'b0100;
    e = 4'b1001;
    f = 4'b0110;
    g = 4'b001;
    h = 4'b1000;
    i = 4'b0101;
    #10;
    a = 4'b0010;
    b = 4'b0010;
    c = 4'b0110;
    d = 4'b0100;
    e = 4'b1001;
    f = 4'b0110;
    g = 4'b001;
```

```
425     h = 4'b1000;
426     i = 4'b0101;
427     #10;
428     a = 4'b0011;
429     b = 4'b0010;
430     c = 4'b0111;
431     d = 4'b0100;
432     e = 4'b1011;
433     f = 4'b0110;
434     g = 4'b001;
435     h = 4'b1000;
436     i = 4'b0101;
437
438     #10 $finish;
439   end
440 endmodule: checkSums_test
441
442 //The top module of the whole project for simulation
443 module IsMagic
444   (input logic [3:0] num1, num2, num3, //top row, L to R
445   input logic [3:0] num4, num5, num6, //middle row
446   input logic [3:0] num7, num8, num9, //bottom row
447   output logic [7:0] magic_constant, //2 BCD digits
448   output logic it_is_magic);
449   logic [7:0]sum;
450   logic checkVals;
451
452   checkValue CV(.num1(num1),.num2(num2),.num3(num3),.num4(num4),.num5(num5),
453   .num6(num6),.num7(num7),.num8(num8),.num9(num9),.isMagic(checkVals));
454   checkSums CS(.a(num1),.b(num2),.c(num3),.d(num4),.e(num5),.f(num6),.g(num7),
455   .h(num8),.i(num9),.sum(sum));
456   always_comb begin
457     if ((checkVals == 1'b1) && (sum != 8'b0000_0000)) begin
458       it_is_magic = 1'b1;
459       magic_constant = sum;
460     end else begin
461       it_is_magic = 1'b0;
462       magic_constant = 8'b0000_0000;
463     end
464   end
465
466 endmodule: IsMagic
467
468 //Test for simulation or testing the IsMagic module
469 module isMagic_test();
470   logic [3:0] num1,num2,num3,num4,num5,num6,num7,num8,num9;
471   logic [7:0] magic_constant;
472   logic it_is_magic;
473   logic checkVals;
474
475   IsMagic isM(.*);
476
477   initial begin
478     $monitor($time, "num1=%b,num2=%b,num3=%b,num4=%b,num5=%b,num6=%b,num7=%b,\
479     num8=%b,num9=%b,magic_constant = %b, checkvals = %b,it_is_magic = %b",
480     num1,num2,num3,num4,num5,num6,num7,num8,num9,magic_constant,checkVals,
481     it_is_magic);
482     num1 = 4'b1000;
483     num2 = 4'b0001;
484     num3 = 4'b0110;
485     num4 = 4'b0011;
486     num5 = 4'b0101;
487     num6 = 4'b0111;
488     num7 = 4'b0100;
489     num8 = 4'b1001;
490     num9 = 4'b0010;
491     #10;
492     num1 = 4'b0011;
493     num2 = 4'b0010;
494     num3 = 4'b0111;
495     num4 = 4'b0100;
```

```
496     num5 = 4'b1001;
497     num6 = 4'b0110;
498     num7 = 4'b001;
499     num8 = 4'b1000;
500     num9 = 4'b0101;
501     #10;
502     num1 = 4'b0010;
503     num2 = 4'b0010;
504     num3 = 4'b0110;
505     num4 = 4'b0100;
506     num5 = 4'b1001;
507     num6 = 4'b0110;
508     num7 = 4'b0001;
509     num8 = 4'b1000;
510     num9 = 4'b0101;
511     #10;
512     num1 = 4'b0011;
513     num2 = 4'b0010;
514     num3 = 4'b0111;
515     num4 = 4'b0100;
516     num5 = 4'b1011;
517     num6 = 4'b0110;
518     num7 = 4'b0001;
519     num8 = 4'b1000;
520     num9 = 4'b0101;
521     #10 $finish;
522   end
523 endmodule: isMagic_test
524
525 //This module is used for testing in synthesis
526 module enter_9_bcd
527   (input  logic [3:0] entry,
528    input  logic [3:0] selector,
529    input  logic       enableL, zeroL, set_defaultL, clock,
530    output logic [3:0] num1, num2, num3, num4, num5, num6, num7, num8, num9);
531
532   logic enableL_async, enableL_sync;
533   logic zeroL_async, zeroL_sync;
534   logic set_defaultL_async, set_defaultL_sync;
535
536   // 2FF Synchronization
537   always_ff @(posedge clock) begin
538     enableL_async      <= enableL;
539     enableL_sync       <= enableL_async;
540     zeroL_async        <= zeroL;
541     zeroL_sync         <= zeroL_async;
542     set_defaultL_async <= set_defaultL;
543     set_defaultL_sync  <= set_defaultL_async;
544   end
545
546   always_ff @(posedge clock) begin
547     if (~zeroL_sync) begin
548       num1 <= 4'b0000;
549       num2 <= 4'b0000;
550       num3 <= 4'b0000;
551       num4 <= 4'b0000;
552       num5 <= 4'b0000;
553       num6 <= 4'b0000;
554       num7 <= 4'b0000;
555       num8 <= 4'b0000;
556       num9 <= 4'b0000;
557     end
558     else if (~set_defaultL_sync) begin
559       num1 <= 4'b1000;
560       num2 <= 4'b0001;
561       num3 <= 4'b0110;
562       num4 <= 4'b0011;
563       num5 <= 4'b0101;
564       num6 <= 4'b0111;
565       num7 <= 4'b0100;
566       num8 <= 4'b1001;
```

```
567       num9 <= 4'b0010;
568     end
569     else if (~enableL_sync)
570     unique case (selector)
571       4'b0001: num1 <= entry;
572       4'b0010: num2 <= entry;
573       4'b0011: num3 <= entry;
574       4'b0100: num4 <= entry;
575       4'b0101: num5 <= entry;
576       4'b0110: num6 <= entry;
577       4'b0111: num7 <= entry;
578       4'b1000: num8 <= entry;
579       4'b1001: num9 <= entry;
580     endcase
581   end
582
583 endmodule: enter_9_bcd
584
```