**ELEE/CMPE 4303 Lab 2 BCD Display and Addition Spring 2018**

Part 1. Develop a module BCD2LEDS to display a 4-bit binary-coded-decimal (BCD) on 7-segment display.

* Follow the Verilog template below for the sub-module

module BCD2LEDS (bcd, leds) ;

input [3:0] bcd; // 0000 to 1001 assumed

output reg [6:0] leds;

always @(\*)

case (bcd)

4’b0000: leds=7’b 100000; // low active LEDs

….

endcase

endmodule

* Test the module in Quartus II and on the DE-2 board. Use an instance of the submodule inside a top module.

Use SW3-SW0 as the BCD number input.

Use HEX0 as the 7-segment display.

module lab2part1(SW,HEX0);

input [3:0] SW;

output [6:0] HEX0; // why not reg here?

BCD2LEDS U0(SW,HEX0) ; // module instantiation

endmodule

Make sure you import the pin assignment file and compile it to generate circuits before you

program it into the FPGA board.

To see how the BCD to 7 segment code converter works, change switch positions and observe the corresponding 7-segment display.

There are two options regarding module organizations.

Option 1: Project name lab2part1

Single source file: lab2part1.v

with codes for the top module and all sub-modules

module lab2part1(SW,HEX0); // top module on top, same name as file name and project name

…

endmodule

module BCD2LEDS (bcd, leds) ; // all sub-modules next

….

endmodule

Option 2: Project name lab1part

Add onto project two .v source files

One file for each of the top and submodule.

Top module filename lab2part1.v (same as project name)

module lab2part1(SW,HEX0); // top module on top, same name as file name and project name

…

endmodule

Sub module filename bcd2leds.v

module BCD2LEDS (bcd, leds) ; // one sub module, one .v file

….

endmodule

While you may use both modules in a single file, it is recommended to have one module definition for one .v file and use project add file to import submodule files as needed. This way, you may easily reuse the submodules for other projects.

Part 2. Create three instances of the above module BCD2LEDS to display 3-digit BCD number to HEX2-HEX0 with each digit controlled by 4 switches.

module lab2part2(SW,HEX2, HEX1,HEX0);

input [11:0] SW;

output [6:0] HEX2, HEX1, HEX0;

BCD2LEDS U1(SW[3:0], HEX0); // create instance U1 out of module BCD2LED

… // two more instances needed

…

endmodule

Part 3. Design a circuit to perform single-digit addition

* Use SW[3:0] to represent 4-bit binary A0, SW[7:4] for B0
* Display decimal A0 on HEX0, A1 on HEX1
* Perform BCD addition to figure out S1 S0=A0+B0, where S0 is the 4-bit sum, S1 is the carry out bit
* Display two digit sum S1 S0 on HEX3, HEX2 respectively.

Example operation of the circuit

Switch positions

on

off

7 6 5 4 3 2 1 0

SW7-4=1000b=8d SW3-0=0101b=5d

7-Segment LED display



HEX3 HEX2 HEX1 HEX0

8+5=13

Top Module

module lab2part3(SW, HEX0,HEX1,HEX2, HEX3);

input ... SW;

output … HEX0, HEX0, HEX1, HEX2, HEX3;

wire S1; // internal signal for the carry-out bit (tens)

wire [3:0] S0;// internal signal for the BCD sum (ones)

// module instantiation to display bcd numbers on 7 segment displays using BCD2LEDS

… // HEX0

… // HEX1

… (S0, HEX2); // HEX2

BCD2LEDS U3(**{3’b000,S1},** HEX3); // HEX3: **Why {3’b000,S1}?**

// module insanitation to perform single-digit BCD addition

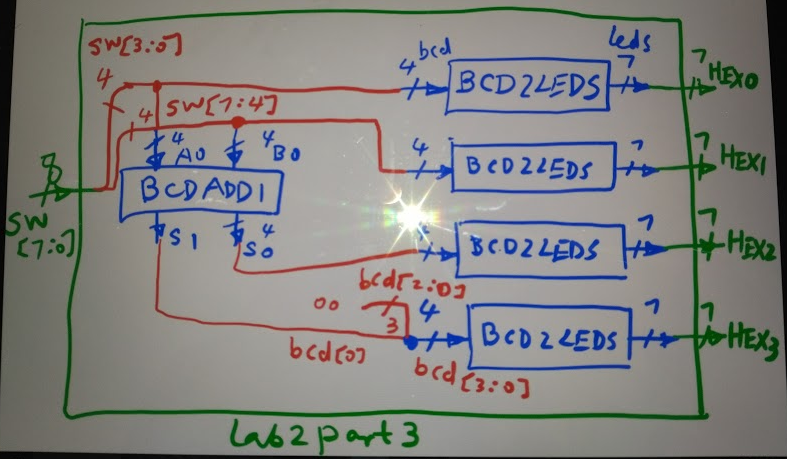
// Refer to module interface definition

// module BCDADD1(A0,B0,S0,S1);

BCDADD1 U4(SW[3:0], SW[7:4], S0, S1);

endmodule

Here is the block diagram of the top module



3’b**000**

Sub-modules to be used

* BCD2LEDS.v as in part 1.
* BCDADD1.v as follows

BCDADD1 to add single-digit BCD to produce BCD sum along with carry out

S1 S0=A0+B0 (S1 is the carry out digit)

// try both methods : 1) use Verilog division/modulus operators and

module BCDADD1(A0,B0,S0,S1);

input [3:0] A0, B0; //4 bit inputs

output [3:0] S0; // 4bit sum ? Why not **reg** type?

output S1;// 1 bit carry-out

assign S0=(A0+B0)/10; // division

assign S1=(A0+B0)%10; // modulus

endmodule

// 2) use always behavior //modeling features inside always block discussed during class

module BCDADD1(A0,B0,S0,S1);

input [3:0] A0, B0; //4 bit inputs

output reg [3:0] S0; // 4bit sum why not **wire** typ?

output reg S1;// 1 bit carry-out

always @(\*)

begin

**{S1,S0}=A0+B0**; // 4-bit add 4-bit to produce 4 bit result and 1 bit carry out

if (A0+B0>9) // otherwise, no correction, S1=0, S0=A0+B0

begin

S0=A0+B0-10;// or S0=A0+B0+6// to have ‘ones’ in S0 as 3 in 13

S1=1;// carry-out

end

end

endmodule

Part 4. Design a module BCDAdd2 to add two-digit BCDs

For example: 57

+ 89

1 4 6

S2 S1 S0= A1A0+B1B0 (two digit adds into another two digits to generate 3-digit sum, where S2 is the carry out digit).

top module

module lab2part4(SW, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5); // how many switches?

sub-module files

BCDADD1FA.v // modify BCDADD1 from a half-adder to a full-adder by adding a carry-in bit

module BCDADD1FA(A0,B0,Cin, S0, Cout); // A0, B0, S0 4bits, Cin, Cout single bits

BCDLEDS.v

BCDADD1.v // optional with 1 half adder 1 full adder instead of two full adders.

Submit your group report online. Signatures needed for attendance during labs.

For each part, include technical details such as

* problem description
* block diagram design
* Truth table / algorithm/pseudo code, etc. your solution before coding
* module source codes
* lab procedure-import pin assignments, compile, simulation, programming, debugging, etc.
* describe lab results with pictures of switches and displays
* conclusion/reflections on what went wrong /what you had learned