Objective

- Based on the experience gained from LAB#1, learn how to design, simulate, synthesize, program on FPGA and test combinational & sequential digital components using Altera Quartus II CAD SW and DE1 FPGA board.
- Learn and become familiar with digital logic design using Verilog Hardware Description Language.
Instructions

1. Your LAB#2 project name should be L2Cyyy, where yyy=your CID (e.g., L2C079 if your CID=079). The golden solution .pof and .sof files are provided. Student should play with golden solution as a reference whenever he/she has a question during design.

2. Use Verilog HDL design. Use the following Verilog top-level module interface code for your design. **No part of this code is allowed to be modified**. The top-level module name must be same as your LAB project name.

   ```verilog
   module L2Cyyy( // where yyy=your CID. e.g., L2C079 if your CID=079
       input [9:0] sw, // ten up-down switches, SW9 - SW0
       input [3:0] key, // four pushbutton switches, KEY3 - KEY0
       input clock, // 24MHz clock source on Altera DE1 board
       output [9:0] ledr, // ten Red LEDs, LEDR9 - LEDR0
       output [7:0] ledg, // eight Green LEDs, LEDG8 - LEDG0
       output reg [6:0] hex3, hex2, hex1, hex0 // four 7-segment, HEX3 - HEX0
   );
   ```

3. Our acceptable timing margin for real-time clock operation is -30 and +30%.
   *For example, for 1-second period required in Part 4&5 of this LAB, a time period between 0.7 sec (= -30%) and 1.3 sec (= +30%) is acceptable as a 1-second period. A time period beyond this range is unacceptable as 1-second period.*

Similar to LAB#1, LAB#2 has a following structure (See each Part for details).
4. LAB#2 Project Operations(**These operations are prerequisite conditions for all Parts**)

When power is turned on, your DE1 board must be in the following initial state:
- all SWs are in DOWN position
- all keys are NOT PRESSED
- all leds(ledg and ledr) are OFF
- No Part of this LAB is enabled

The sw[9:5] is a Part selector. You enable or disable a particular Part by setting the sw[9:5] as follows. No more than one switch on sw[9:5] is allowed to be in UP position (i.e., no more than one Part is enabled at the same time!).

```
IF  sw[9:5]=00000      // all sw are in DOWN position
    Initial state    AND all Parts are disabled

    Part1 is enabled AND all other Parts are disabled

    Part2 is enabled AND all other Parts are disabled

ELSE IF sw[9:5]=00100    // only sw[7] is in UP position
    Part3 is enabled AND all other Parts are disabled

ELSE IF sw[9:5]=01000    // only sw[8] is in UP position
    Part4 is enabled AND all other Parts are disabled

ELSE IF sw[9:5]=10000    // only sw[9] is in UP position
    Part5 is enabled AND all other Parts are disabled
```

**Warning:** Above operations are prerequisite conditions. You will get **zero(0) point** for LAB#2 if you fail above operations **regardless of Parts**.
PART 1 (Basic)  Decimal and Hex Number Display design

******************************************************************************

Design a Decimal and Hex Number Display circuit as follows.

Inputs:  SW[3:0]  // four-bit binary number input
Output:  HEX[3:0]  // displays Decimal and Hex numbers

Operation
   if Part1 is enabled  // see Sec. 4. LAB#2 Project Operations

          HEX[3:2] => displays a Decimal number of SW[3:0].
          HEX[0]  => displays a Hex number of SW[3:0].

******************************************************************************

--------------------------  Hints  -----------------------------------------------

For example,

<table>
<thead>
<tr>
<th>SW[3:0]</th>
<th>HEX[3:2]</th>
<th>HEX[0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>01</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>02</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>03</td>
<td>3</td>
</tr>
<tr>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>1000</td>
<td>08</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>09</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
<td>A</td>
</tr>
</tbody>
</table>
| 1011    | 11       | b      | // <--- use lower case!
| 1100    | 12       | C      |
| 1101    | 13       | d      | // <--- use lower case!
| 1110    | 14       | E      |
| 1111    | 15       | F      |

--------------------------  The End of Part1  -------------------------------
PART 2 (Basic)  Adder/Multiplier design

Design an Adder/Multiplier circuit as follows.

**Inputs:**
- SW[4:3] = operand1 in binary
- SW[2:1] = operand2 in binary
- SW[0] is an operation selector: 0 for Addition, 1 for Multiplication

**Output:**
- HEX[3] = Decimal value of operand1
- HEX[2] = Decimal value of operand2
- HEX[1] = OFF (i.e., no light)
- HEX[0] = Decimal value of Result

**Operation:**
- If Part2 is enabled  // see Sec. 4. LAB#2 Project Operations
- HEX[3:0] displays values defined above Adder/Multiplier circuit

----------------------- The End of Part2 -----------------------

--------------- Hints ---------------

For example,

<table>
<thead>
<tr>
<th>SW[4:0]</th>
<th>HEX[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>00 0</td>
</tr>
<tr>
<td>00010</td>
<td>01 1</td>
</tr>
<tr>
<td>01010</td>
<td>11 2</td>
</tr>
<tr>
<td>10100</td>
<td>22 4</td>
</tr>
<tr>
<td>11100</td>
<td>32 5</td>
</tr>
<tr>
<td>11110</td>
<td>33 6</td>
</tr>
<tr>
<td>....</td>
<td></td>
</tr>
<tr>
<td>00001</td>
<td>00 0</td>
</tr>
<tr>
<td>00011</td>
<td>01 0</td>
</tr>
<tr>
<td>01011</td>
<td>11 1</td>
</tr>
<tr>
<td>10101</td>
<td>22 4</td>
</tr>
<tr>
<td>11101</td>
<td>32 6</td>
</tr>
<tr>
<td>11111</td>
<td>33 9</td>
</tr>
<tr>
<td>....</td>
<td></td>
</tr>
</tbody>
</table>
**PART 3 (Intermediate)  Modulo-16 Up/Down Counter design**

*****************************************************************************

Design a Modulo-16 Up/Down Counter circuit as follows.

**Inputs:**
- KEY[2] for *input*. An input is entered to counter each time the key is *pressed down* (Note that NO input is entered when the key is *released*).
- SW[0] for *reset* operation (0 for normal counting, 1 for *clearing the counter output to zero*).
- SW[1] for selecting direction of counting (0 for *Up*, 1 for *Down* counting) (SW[1] changes the direction of counting at any moment during operation.)

**Output:**
- HEX[2] = counter output in *hex*. All other HEXs = OFF (no light).

**Operation:**
- If Part3 is enabled  // see Sec. 4. LAB#2 Project Operations
  1) The initial value of HEX[2] must be 0 when sw[7] goes up (i.e., when Part3 is enabled)
  2) Your circuit counts the number of pressing on KEY[2] and displays the result on HEX[2]. Therefore HEX[2] increases or decreases each time KEY[2] is pressed depending on SW[1].
  3) SW[1] changes the direction of counting at any moment during operation.
  4) Your counter output should work as Modulo-16 operation.
  5) SW[0] is a *reset* switch. If SW[0]=0, the counter operates normally. If SW[0]=1 then the counter output HEX[2] is cleared to 0 and the counting function is not performed.

*****************************************************************************  The End of Part3  ****************************************************************************

-------------------------------------  Hints  -------------------------------------

For example,
- Case1) When **sw[1]=0**, 0 => 1 => 2 => 3 => ... => d => E => F => 0 => 1 => 2 => 3 =>...
- Case2) When **sw[1]=1**, 0 => F => E => d => ... => 3 => 2 => 1 => 0 => F => E => d =>...
- Case3) A new counting starts with **sw[1]=0**, HEX[2] starts from 0 (by reset), 0 => 1 => 2 => 3 => ... => d => E => F => 0 => 1 => 2 => 3 => here, sw[1]=1 3 => 2 => 1 => 0 => F => E => d => ... => 3 => 2 => 1 => 0 => F => E => d here, sw[1]=0  d => E => F => 0 => 1 => 2 => 3 =>......
**PART 4 (Intermediate)  Real-Time Measurement Circuit design**

Design a Real-Time Measurement Circuit as follows.

**Inputs:**  
SW[0] for reset 

**Output:**  
HEX[3:0] for measurement output (in Modulo-3 operation)  
LEDG[0] for blinking signal 

**Operation**  
If Part4 is enabled // see Sec. 4. LAB#2 Project Operations

1. HEX[3:0] starts displaying the number of seconds passed since the moment when SW[8] goes up (i.e., when Part4 enabled). Each HEX digit displays the counter output in Modulo-3 operation.

2. The LEDG[0] starts blinking every second with 50% duty cycle as follows.

   ![Diagram showing LEDG[0] blinking pattern](image)

3. SW[0] is a reset switch. If SW[0]=0, the timer operates normally. If SW[0]=1 then HEX[3:0] is cleared to 0000, LEDG[0]= OFF (no light), and the time measurement function is not performed.

***************  The End of Part4 ****************************
Bouncing Ball with Moving Message Display design

Design a Bouncing ball with Moving message circuit as follows.

**Inputs:**  SW[0] for **pausing** the operation (0 for resume operation, 1 for **pausing**)

**Output:**  LEDR[9:0] for bouncing ball
          HEX[3:0] for moving message

**Operation**

If **Part5** is enabled  // see Sec. 4. LAB#2 Project Operations

1.  [Bouncing Ball on LEDR[9:0]]
    Starting from LEDR[0] position, a red light ball moves from LEDR[0] to LEDR[9] with a duration of **0.5 second**. When arrived at LEDR[9], the ball moves from LEDR[9] back to LEDR[0] with same duration of **0.5 second**. Therefore the time period of one round trip is **one(1) second**. When returned to LEDR[0], the red light ball keeps repeating the same movement.

2.  [Moving Message on HEX[3:0]]
    A message, "HELLO  Cid <yourCID> ", is moving from right to left repeatedly. For example, "HELLO  Cid 353 " in golden solution.
    The message movement is synchronized to the bouncing ball. The message moves **one letter whenever the bouncing ball hits the LEDR[9](=left edge)**.

3.  SW[0] is a **pause** switch (it's not a **reset** switch!).
    SW[0] = 1 pauses the operation.
    SW[0] = 0 resumes the operation.

***************  The End of Part5  ***********************

---------------  **Hints**  -------------------------------
Knowledge of handling clock operation learned from Part4 may be helpful for this Part also.

---------------  The End of LAB#2  -------------------------------