UCSD CSE140L Fall 2013

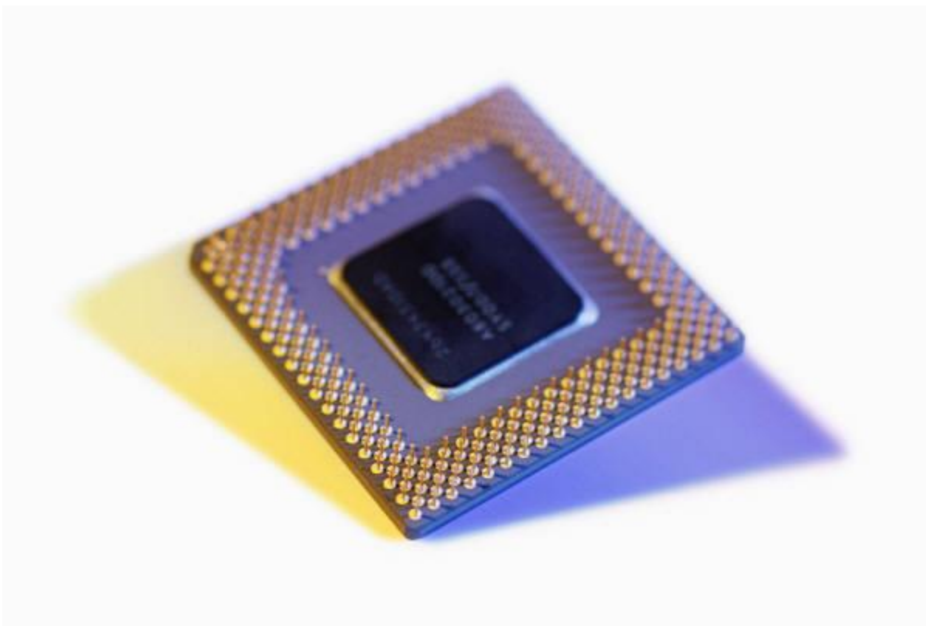
LAB#2

(Due Date & Time: See course web page)

Instructor: Dr. Choon Kim

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

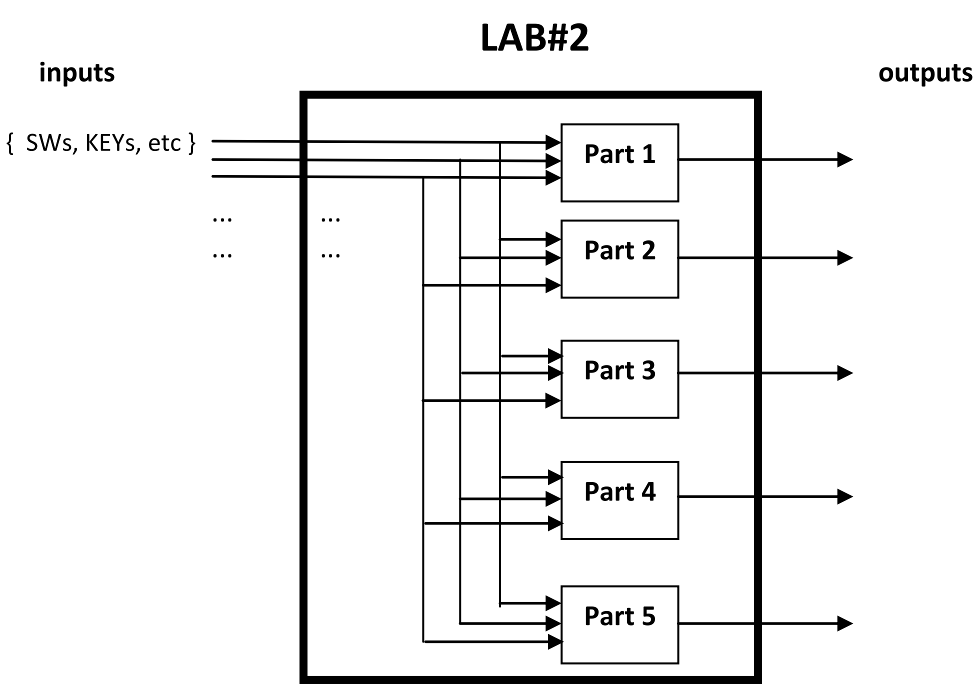
1. Your LAB#2 project name should be L2Cyyy, where yyy=your CID(e.g., L2C079 if your CID=079). The golden solution .pof and .sof files are provided.
Student should play with golden solution as a reference whenever he/she has a question during design.

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.

```
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 Part4&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 <u>prerequisite</u> 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
- hex[3]=OFF(no light), hex[2:0] displays your CID. For example, HEX[3:0]= **097** if your CID=097. The *Golden solution displays HEX[3:0]=* **353** *since it's CID=353.*

The **sw[9:5]** is a <u>Part selector</u>. 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

ELSE IF sw[9:5]=00001 // *only sw[5] is in UP position*
Part1 is enabled AND all other Parts are disabled

ELSE IF sw[9:5]=00010 // *only sw[6] is in UP position*
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[1] => OFF(no light).
HEX[0] => displays a **Hex** number of SW[3:0].

**************** **The End of Part1** ***

------------------------------------ *Hints* --

For example,

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

PART 2 (*Basic*) Adder/Multiplier design

****

Design an <u>Adder/Multiplier</u> 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,

SW[4:0]	HEX[3:0]	
0000**0**	00 0	// addition...
0001**0**	01 1	
0101**0**	11 2	
1010**0**	22 4	
1110**0**	32 5	
1111**0**	33 6	
.....		
0000**1**	00 0	// multiplication...
0001**1**	01 0	
0101**1**	11 1	
1010**1**	22 4	
1110**1**	32 6	
1111**1**	33 9	
.....		

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

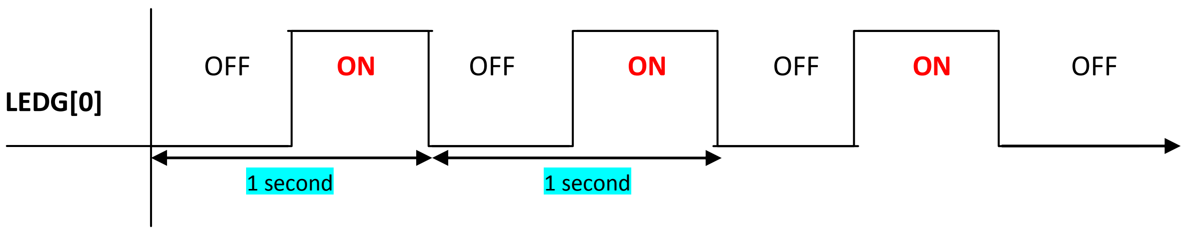
Output: HEX[3:0] for measurement output (in Modulo-3 operation)
LEDG[0] for blinking signal

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.



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** **

-------------------------------------- *Hints* --

1) DE1 User manual sec. 4.4. for **clock operation** may be helpful
2) For example ,

HEX[3:0] = 0000 // <---- when SW[8] goes up here! (i.e., Part4 enabled)
HEX[3:0] = 0001 // after one second passed
HEX[3:0] = 0002 // after another second passed(i.e., two seconds passed)
HEX[3:0] = 0010 // after another second passed(i.e., three seconds passed),
HEX[3:0] = 0011 .....
HEX[3:0] = 0012
HEX[3:0] = 0020
HEX[3:0] = 0021
....
HEX[3:0] = 2222
HEX[3:0] = 0000 // <----back to 0000, Modulo-3 operation!

PART 5 (***Advanced & Challenging!***)

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 --------------------------