The Final Lab Project

Michael Brodskiy
Professor: S. Shazli

April 6, 2023

- In this project, you will utilize the 7-segment displays on the DE1-SoC board to display characters, decimal, and hexadecimal values using object oriented design in C++
- The DE1-SoC has six 7-segment displays controlled by two parallel ports
- Each segment (6 to 0) is controlled by one bit, with the first parallel port controlling the first four displays (HEX3, HEX2, HEX1, & HEX0), and the second parallel port controlling the last two displays (HEX5 & HEX4)
- Data can be written into these two registers, and read back, by using word operations
- The seven segment display table:

#	6	5	4	3	2	1	0	Decimal	Hex
0	0	1	1	1	1	1	1	63	0x3F
1	0	0	0	0	1	1	0	6	0x6
2	1	0	1	1	0	1	1	91	0x5B
3	1	0	0	1	1	1	1	79	0x4F
4	1	1	0	0	1	1	0	102	0x66
5	1	1	0	1	1	0	1	109	0x6D
6	1	1	1	1	1	0	1	125	0x7D
7	0	0	0	0	1	1	1	7	0x7
8	1	1	1	1	1	1	1	127	0x7F
9	1	1	0	1	1	1	1	111	0x6F
A	1	1	1	0	1	1	1	119	0x77
b	1	1	1	1	1	0	0	124	0x7C
C	0	1	1	1	0	0	1	57	0x39
d	1	0	1	1	1	1	0	94	0x5E
e	1	1	1	1	0	0	1	121	0x79
f	1	1	1	0	0	0	1	113	0x71

- The `DE1SoCfpga` class from the previous lab should be converted to a header file called `DE1SoCfpga.h`
- It should be implemented in a file called `DE1SoCfpga.cpp`
- The following functions need to be part of the header:
  - Constructor initializing the memory-mapped I/O
  - Destructor finalizing the memory-mapped I/O
  - Function `RegisterWrite(offset, value)`, which writes a value into a register given its offset
  - Function `RegisterRead(offset)`, returning the value read from a register given its offset

- The class `DE1SoCfpga` will be the base class initializing memory and controlling register access
- There are two offsets for the hex displays:

```
const unsigned int HEX3_HEX0_Base = 0x00000020;
```

and

```
const unsigned int HEX5_HEX4_Base = 0x00000030;
```

- For the seven segment display, a new class should be created, called `SevenSegment`, with the following functionality:
  - Two private data members: `unsigned int reg0_hexValue` and `unsigned int reg1_hexValue` representing the state of the two 7-segment display registers
  - These variables should be updated every time a new value is written to the corresponding registers
  - A class constructor that initializes the private data members
  - A class destructor that calls `Hex_ClearAll()` function to clear the displays
  - A public function called `Hex_ClearAll()` that clears (turns off) all the 7-segment displays
  - A public function called `Hex_ClearSpecific(int index)` that clears (turns off) a specified 7-segment display, specified by `index`, where the index (0 to 5) represents one of the six displays
  - A public function called `Hex_WriteSpecific(int display_id, int value)` that writes a hexadecimal digit specified by a decimal value(0 to 15) to the specified 7-segment display, specified by `display_id`, where the `display_id` (0 to 5) represents one of the six displays
  - A public function called `Hex_WriteNumber(int number)` that writes a positive or negative number on the 7-segment displays
  - Use the `main()` function provided to test the file
- The class `SevenSegment` inherits the class `DE1SoCfpga` and uses the functions `RegisterWrite(...)` and `RegisterRead(...)` as needed
- Create a global `const` array with 16 elements (see the declaration below)
- Each element contains the decimal (or Hex) code from the table above that represents the display segments to be turned ON or OFF when writing to a 7-segment display
- The index corresponds to the digits to write on the 7-segment display

- Include the array in the `SevenSegment.h` file to use when writing to the 7-segment displays. The arrays shoulder have the following format:

```
const unsigned int bit_values[16] = {...};
```

- The seven segment displays should then be integrated with `LEDControl` from the previous lab