

Surname, Name: \_\_\_\_\_

## ***Design of a simple hexadecimal calculator***

*Laboratory session 4  
Departamento de Tecnología Electrónica  
Universidad de Sevilla  
Jorge Juan Chico, December, 2020*

### **1 Material**

---

- Computer with [Xilinx's ISE](#) software installed.
- [Digilent's Basys 2 board and documentation](#).
- Initial design files (lab kit code).

### **2 Description**

---

The objective of this lab is to design and implement a basic hexadecimal calculator on a FPGA prototype board. The calculator takes two positive 4-bit input numbers 'a' and 'b' and does four different operations controlled by a 4-bit signal 'op' as specified by the following table:

<b>op</b>	<b>result</b>
1xxx	a + b
01xx	a – b
001x	b – a
0001	a * b
0000	0

The result of the operation is given to a 7-segment display controller that drives a four digits 7-segment display. From left to right the digits represent: 'a' (1 digit), 'b' (1 digit) and the result (2 digits). The schematic of the calculator is depicted in Figure 1, including the names of the signals and the names of the ports in the Basys2 board where the inputs and outputs should be connected.

The display control module and the calculator's top level module are already designed. The display control modules is provided in already synthesized format (.ngc) with a Verilog wrapper. The student will have to complete the design and test the arithmetic circuit as a pre-lab work.

Once implemented in the Basys2 board, the calculator operates as follows:

1. The value of 'a' is introduced in binary format through switches SW7 to SW4. The value of 'a' is displayed in the first digit (leftmost) of the 7-segment display in hexadecimal format.
2. The value of 'b' is introduced in binary format through switches SW3 to SW0. The value of 'b' is displayed in the second digit (starting from the left) of the 7-segment display in hexadecimal format.
3. The user selects the operation to perform by pressing one of the buttons BTN3 to BTN0. The calculator will display the result of the operation in the two rightmost digits of the 7-segment display in hexadecimal format. For example, if  $a=2$ ,  $b=14$  and button BTN1 is pressed, the display will show "2E0C" because button BTN1 triggers operation  $b-c=14-2=12$ , and 12 is '0C' in hexadecimal with two digits.

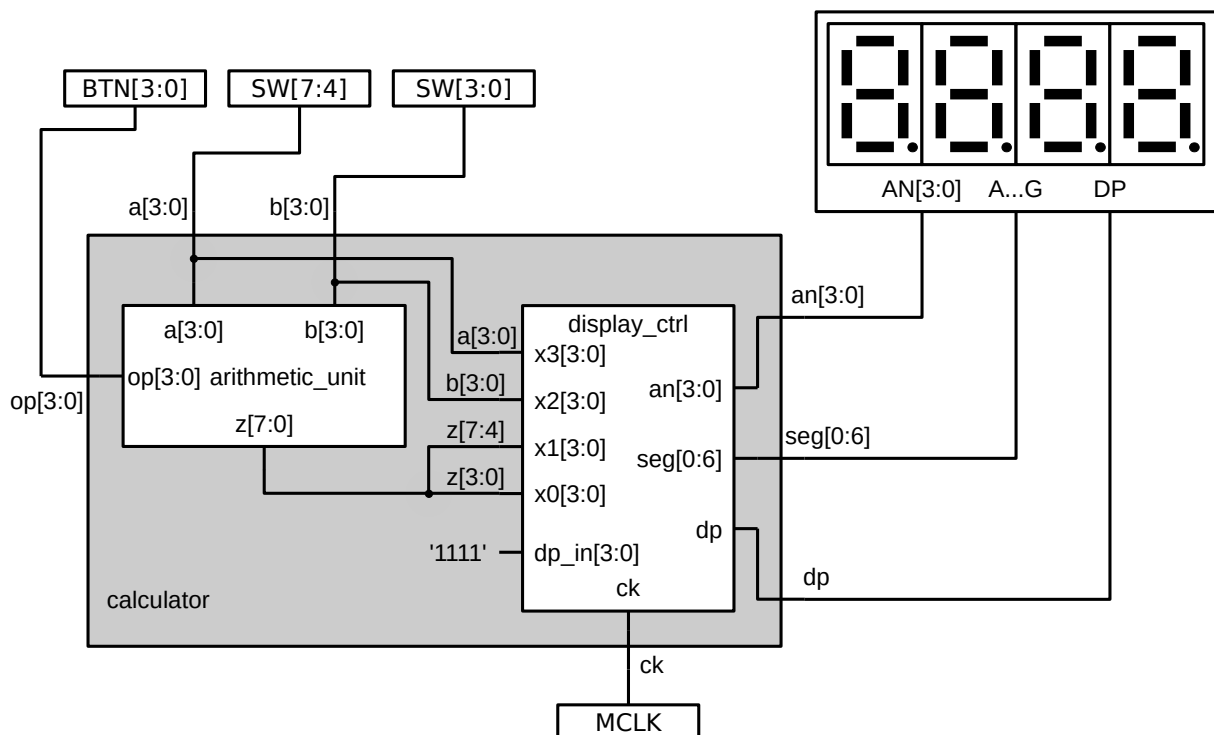


Figure 1: Simple calculator based on an arithmetic unit.

### 3 Pre-lab work

1. Download the `lab4_kit.zip` file from the course's web page. The contents of the package is:
  - `calculator.v`: top-level design file (gray colored in the figure).
  - `calculator_tb.v`: test bench.
  - `arithmetic_unit.v`: arithmetic unit that implements the calculator operations.
  - `display_ctrl_wrapper.v` and `display_ctrl.ngc`: display controller design.
  - `Basys2_100_250_calculator.ucf`: generic UCF file.

Get familiar with the Verilog files and identify in the code the structure of the circuit depicted in Figure 1.

2. Complete the design of the arithmetic unit to implement the operations described in the table above.
3. Use the provided test bench in `calculator_tb.v` to verify the design. With Icarus Verilog you can do:


```
iverilog *.v  
vpp a.out
```

4. Edit the UCF file to make the connections of the signals to the board's devices as depicted in the figure.
5. Upload the files or bring them to the lab following the instruction from the teaching staff.
6. (Optional) See section 5 and write the Verilog code for the extension work if you are interested in doing so.


## 4 Lab work

---

1. Copy all the lab files to a folder in the lab computer.
2. Create a new project in Xilinx's ISE with name "calculator". Use Basys2 project properties: General Purpose, Family: Spartan3E, Device: XC3S100E, Package: CP132, Speed grade: -5.
3. Add all the source files to the project, this includes all the ".v" files, the ".ucf" file and the ".ngc" file.
4. Simulate the test bench in ISIM and check that the results are correct.

 **Ask the instructor to check this step.**

5. Execute the synthesis and implementation of the design. Correct the errors, if any, and repeat.
6. Program the design in the FPGA board and test it.

 **Ask the instructor to check this step.**

7. Modify the design to make the decimal point between the second and third digits to be on.

 **Ask the instructor to check this step.**

8. Modify the design to make LD0 to activate when the result is negative after a subtraction operation (in this case the displayed value is not correct).

 **Ask the instructor to check this step.**

## 5 Extension (optional)

---

We want to change our calculator to operate with 4-bit numbers in two's complement representation. Now, the output 'z' of the arithmetic unit is only 4 bits and the operations are:

op	result
1xxx	$a + b$
01xx	$a - b$
001x	$a + 2b$
0001	$a - 4$
0000	0

The arithmetic unit has an additional output bit 'v' that activated when there is an overflow condition. The calculator will represent the output value in the rightmost digit of the 7-segment display and will activate LED0 when there is overflow.

1. Modify the arithmetic unit as described in the table. Do not forget to use 'signed' type for the input and output numbers this time.
2. Modify the test bench, if necessary, and simulate the new arithmetic unit. Check that the operation is correct, including the cases with overflow.
3. Modify the 'calculator' module to include the modifications: now 'z' is only connected to one digit and the new overflow output of the arithmetic unit should be connected to LED0.
4. Implement, program and test the circuit in the board.