Laboratory 2

(Due date: **002/003/007/010**: Feb. 8th, **004**: Feb. 9th, **006/008**: Feb. 10th, **011**: Feb. 12th)

OBJECTIVES

✓ Use the Structural Description on VHDL.
✓ Test arithmetic circuits on an FPGA.

VHDL CODING

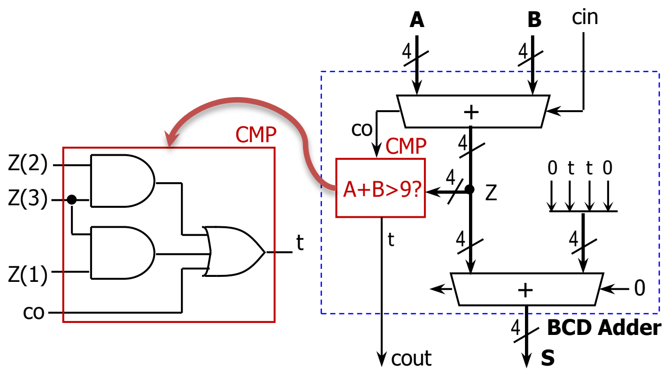
✓ Refer to the Tutorial: VHDL for FPGAs for a list of examples.

FIRST ACTIVITY (100/100)

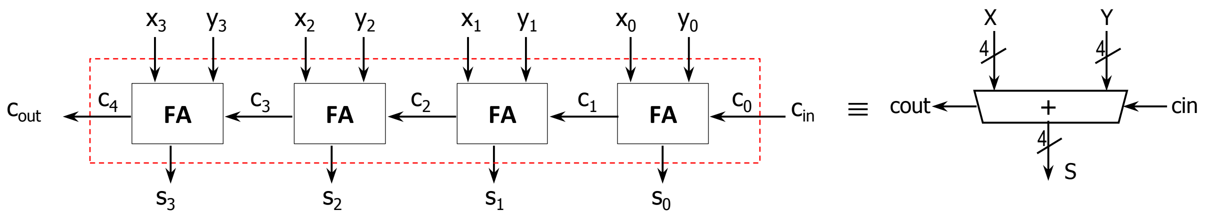
DESIGN PROBLEM

- BCD Addition of two numbers. The operands (A and B) are 4-bit unsigned numbers represented in BCD (where only numbers from 0 to 9 are allowed). The result S is also represented in BCD. There is also a BCD carry out ($cout$). If any of the operands is greater than 9, the result S is invalid.
  Example: $7 + 8 = 15$. Here, $cout = 1$, and $S = 0101 = 5$.

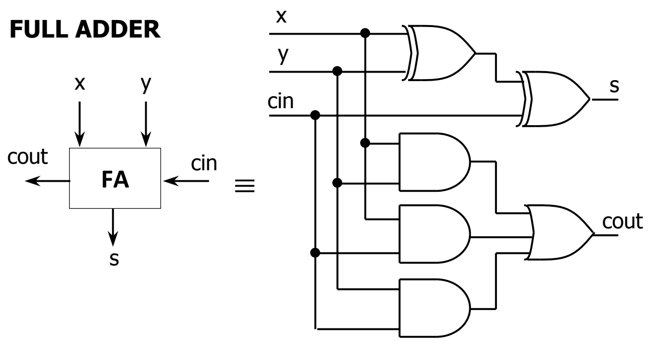
  This circuit can be built out of two 4-bit binary adders and a few logic gates as depicted in the figure ⇒
  - ✓ Operation: If $A + B > 9 \rightarrow S = 6 + Z$, $cout = 1$. Here, by adding 6, we "correct" the binary sum to make it look as BCD code. If $A + B \leq 9 \rightarrow S = Z$, $cout = 0$.



- The figure below depicts the internal architecture of the 4-bit binary adder. The full adder circuit is also shown.



FULL ADDER



PROCEDURE

- **Vivado: Complete the following steps**:
  - ✓ Create a new Vivado Project. Select the corresponding Artix-7 FPGA device (e.g.: the XC7A50T-1CSG324 FPGA device for the Nexys A7-50T).
  - ✓ Write the VHDL code for the BCD Adder.
    - ▫ Use the Structural Description: Create a separate `.vhd` file for the Full Adder, the 4-bit adder, the 'CMP' block, and the top file (BCD Adder).
  - ✓ Write the VHDL testbench to test the circuit for the following cases:
    - ▫ **A**=0x6, **B**=0x8, cin=1 → cout=1, **S**=0101
    - ▫ **A**=0x2, **B**=0x7, cin=0 → cout=0, **S**=1001
    - ▫ **A**=0x9, **B**=0x8, cin=0 → cout=1, **S**=0111
    - ▫ **A**=0x6, **B**=0x4, cin=0 → cout=1, **S**=0000
    - ▫ **A**=0x4, **B**=0x8, cin=1 → cout=1, **S**=0011

Instructor: Daniel Llamocca
TAs: Sandeep Kumar, Ala'aldin Hijaz, Moath Algharibeh

- ✓ Perform Functional Simulation and Timing Simulation of your design. **Demonstrate this to your TA**.
  - ▫ Functional Simulation: Add the internal signals (Z, co) to the waveform view. Go to: SCOPE window: testbench → UUT. Then go to Objects Window → Signal(s) → Add to Wave Window. Finally, re-run the simulation.
    - ⌁ This step is extremely useful when debugging your circuit. Your circuit might be cleared of syntax errors, but there might still be errors that can be difficult to spot. By tracing the internal signals, we can determine where the error is located in the circuit.

    - ⌁ For example: In this circuit, for the given set of input values, we know the expected output values and we can compute the internal signal values. Then, we compare those values with those provided by the simulation:
      - If the output `S` is incorrect (simulation results do not match the expected values), we then look at the value of the internal signal Z.
        - If the value of Z is correct (i.e., simulation results match the calculated values), then the error is located in the lower adder.
        - If the value of Z is incorrect, then the error is on the upper adder.

      - If the output `cout` is incorrect (simulation results do not match our calculated values), we then look at the value of the internal signals Z and co.
        - If the value of Z and co are correct (i.e., simulation results match our calculated values), then the error is in the CMP circuit.
        - If the value of Z or co is incorrect, then the error is on the upper adder.

  - ▫ For the following set of inputs, complete the expected values of the listed internal signals. Then, run the simulation and compare the values in the simulation waveform with the ones you computed. This will help you figure out where the errors (if any) are located at.

A	**B**	cin	co	Z	cout	S
0010	0111	0			0	1001
1001	1000	0			1	0111

- ✓ I/O Assignment: Generate the XDC file associated with your board.
  - ▫ Suggestion:

Board pin names	SW8	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0	LED4	LED3	LED2	LED1	LED0
Signal names in code	cin	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	cout	S_3	S_2	S_1	S_0

  - ▫ The board pin names are used by all the listed boards (Nexys A7-50T/A7-100T, Basys 3, Nexys 4/DDR). The I/Os listed here are all active high.

- ✓ Generate and download the bitstream on the FPGA and test. **Demonstrate this to your TA**.

▪ Submit (as a .zip file) the six generated files: VHDL code (4 files), VHDL testbench, and XDC file to Moodle (an assignment will be created). DO NOT submit the whole Vivado Project.
- ✓ Your .zip file should only include one folder. Do not include subdirectories.
  - ▫ It is strongly recommended that all your design files, testbench, and constraints file be located in a single directory. This will allow for a smooth experience with Vivado.

```
lab2
 ├── top.vhd      ┐
 ├── add4.vhd     │ Design files
 ├── fa.vhd       │
 ├── cmp.vhd      ┘
 ├── top_tb.vhd     Testbench file
 └── lab2.xdc       Constraints file
```

TA signature: ______________________________ **Date:** ___________________________