CPEG 422 Project 1
Add/subtractor with carry ripple and CLA implementation

During this project, you will implement a 24-bit add/subtractor in VHDL in two ways. The first implementation is a carry-ripple adder, while the second implementation is a two-level carry look-ahead (CLA) adder. You will perform a side-by-side comparison of the two structures in terms of their hardware utilization and timing, and submit a report about your findings.

Goals of this project:

Be familiar with VHDL programming and syntax
Learn to build up hierarchical design
Learn to use Vivado IDE
Learn to write test bench

Minimal Hardware and Toolkit:

Zybo Z7-10 board
Vivado IDE

Your Tasks:

The project is divided into four tasks. Each of them contributes 25% to the project grade.
Task 1: implement a 24-bit carry-ripple adder/subtractor
Task 2: implement a 24-bit CLA adder/subtractor.
Task 3: write a behavioral testbench for the two 24-bit adders
Task 4: compare the hardware cost, power, and delay of the carry-ripple and CLA adder/subtractor side-by-side.

Implementation details:

Your implementation should follow the entity definition shown to the right.

```
entity addsub_24 is
 Port (
    A :   in STD_LOGIC_VECTOR (23 downto 0);
    B :   in STD_LOGIC_VECTOR (23 downto 0);
    M:   in STD_LOGIC;
    S :   out STD_LOGIC_VECTOR (23 downto 0);
    Overflow : out STD_LOGIC
    );
end addsub_24;
```

The 24-bit ripple adder/subtractor should have three entity definitions and follow a three-level structure: full_adder (FA) → 4-bit carry ripple (optional)→ 24-bit carry ripple. Similarly, the 24-bit cla adder/subtractor also follows a hierarchical structure: CLA generator → 4-bit CLA (optional)→ 16-bit (or 12-bit) CLA → 24-bit CLA. In both designs, you can convert adder to adder/subtractor at any level.

All computations should be at the logic gate level, and you cannot explicitly use operator '+' and '-'. You should also write your own testbench to simulate your design and verify its correctness. Please submit both your VHDL adder implementation and the testbench.

Test cases:

Organize your test procedure in three phases:

Phase	Goal	Modules	Test bench	Run simulation	Run synthesis & implementation
1	Behavioral correctness	Your design + LFSR	Yes	Yes	No
2	Report design hardware, power and schematics	Your design	No	No	Yes
3	Report timing	Your design + FF_wrapper	No	No	Yes

Final report content requirement:

1. Design summary: summarize your design idea and design framework briefly, include design block diagram if necessary.
2. Design simulation results: briefly describe how you test your design correctness, include the behavioral simulation waveform for each of the adder design.
3. Design implementation report: summarize the design implementation report offered in Vivado. For each adder design, please report the following:
    1) Hardware utilization of post-implementation (FF, IO, LUT, BUF…) in table format.
    2) Power report (dynamic vs. static, also signal, logic and I/O).
    3) Timing report (critical path delay).
    4) Design schematic snapshot.
4. Design comparison: compare the two adder implementations based on the information you obtained from the design report.

Grading criteria:

Any of the following may account for deduction of your score:

- Incorrect/unexpected operation or function
- Inconsistent with implementation requirement
- Sloppy, undocumented program code
- Late for your submission (please refer to the late policy in course syllabus)

Due Dates:

- VHDL codes of design and test bench (.vhd source files): 03/06 at midnight
- Final project report: 03/09 at midnight