CPEG 422 Project 2
24-bit Booth's Multiplier

During this project, you will implement a 24-bit Booth's multiplier. The two inputs and the output are 2's complement numbers. The multiplier starts to operate upon receiving a start signal, and produces results in 48 cycles.

Goals of this project:

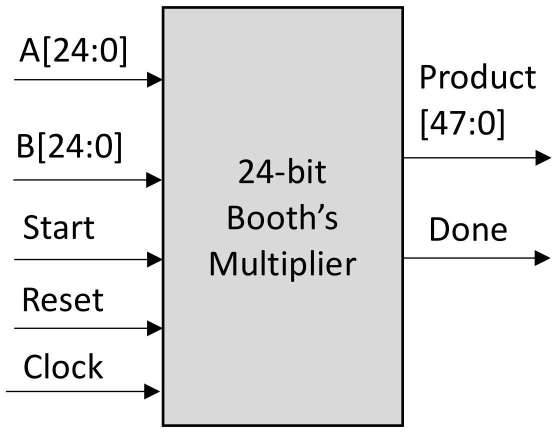
- Learn to implement sequential logic
- Learn to build up hierarchical design
- Learn to write testbench
- Learn to observe waveform
- Be familiar with VHDL programming and syntax

Your tasks:

- Task1: Implement a 24-bit Booth's multiplier for 2's complement number. (40%)
- Task2: Write a test-bench for the 24-bit multiplier. (30%)
- Task4: Submit your design report. (30%)

Minimal Hardware and Toolkit:

Zybo
Vivado 2017.4



Top module inputs:

- A *Clock* signal;
- A synchronous *Reset* signal;
- A *Start* signal to start the multiplier;
- Two 24-bit numbers in 2's complement form: A [23:0] and B [23:0].

Top module outputs:

- A 48-bit multiplication result： Product [47:0];
- A *Done* signal, indicating that the result is ready (at the next clock rising edge) and the multiplier can take the next input.

General functional implementation requirement:

- You should strictly follow the signal declaration and their order when defining your top-level entity.
- All computation should be based on logic gate, and you cannot explicitly use operator '+', '-' and '*'.
- The top module should include 4 components:
  1. A 24-bit CLA adder that computes partial product.
  2. Three universal registers that support load and shift, used to store the multiplicand, the multiplier, and the partial product. When init=1, A and B are loaded while the product register is cleared.
  3. A 5-bit counter for counting shift iterations. count_up becomes high when the counter overflows, indicating that 24 shifts have been performed.

4. A control unit implemented as a Moore FSM. The signals and state machine are given in Lecture 8 slides.

- The sequential behavior of the 24-bit multiplier should be the following: (1) The **Reset** clears all components, it is only valid (high) for one cycle, then remains low; (2) The **Start** signal initiates computation. It is one-cycle width pulse signal. It becomes high after reset; (3) The **A/B** inputs should be ready and stable once the start becomes high. In the testbench, both A and B can be generated with LFSR, and start signal can serve as *enable* to the LFSR; (4) The **Done** signal indicates that computation is finished and the multiplier can take another set of inputs. Initially it is high, indicating that the multiplier is ready. It becomes low when the multiplication is in process. It becomes high again when the result is ready at the next rising edge of the clock; (5) The **Product**: result is meaningful and valid **only if** done is valid.
- You should write your own testbench to simulate design and verify its correctness. Make sure that your design is robust and stable. Specifically, your testbench need to test three cases: (1) when the multiplier is done with the last computation (the *done* signal is high), it should be able to process new inputs and repeat correct multiplication once the start signal becomes high again; (2) when the inputs *start* changes during the multiplication process (before done becomes high), it does not have any effect; (3) when the inputs *A* or *B* change during the multiplication process, it does not affect the correctness of the final product.

Report requirements:

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, how to write your testbench, and include the simulation waveforms of the three cases (one stable, two unstable) described above. Show at least three consecutive multiplications process with different values of A and B.
3. Design implementation report: summarize the design implementation report offered in Vivado. Please report the following (snapshot is fine):
   1) Hardware utilization (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.

Grading criteria:

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

- Incorrect/unexpected operation or function
- Unfamiliarity with any aspect of your project
- Late for your submission
- Inconsistent with implementation requirement
- Sloppy, undocumented program code

Due Dates:

Design source code/testbench submission:	04/06 at midnight
Report submission:	04/08 at midnight