Project 1 COMBINATIONAL ARITHMETIC CIRCUITS

PROJECT DESCRIPTION & REQUIREMENTS

1. Each group will implement the arithmetic circuit assigned to them (See the table at pages 2-3), according to the references provided and requirements given below.

2. Unless otherwise specified, circuits will perform **signed integer arithmetic** and will have two 32-bit inputs A and B, a 32-bit output C, and a single bit output “carry”.

3. Give a brief explanation of the algorithm/topology that’s assigned to you.

4. Explain your design steps clearly. A reader should be able to follow your work along.

5. Code the combinational design using Verilog. Add (* DONT_TOUCH = “TRUE” *) inline attribute in front of your **top module definition** to prevent it to be optimized by the synthesizer (see below).

   ```
   (* DONT_TOUCH = "TRUE" *)
   module a_top_module( input x, input y, output z );
   ```

6. For testing the circuit, you are tasked to write a script/program that will create 100 random input pairs for your circuit and write them into a text file. Using this, create one such stimulus file that will be used in your testbench. You may use any programming language you like.

7. Prepare a testbench for your circuit. It should be able to read all the test values from your stimulus file created at step 6, apply them to the circuit one by one, then write the corresponding results to another output text file. For each input pair, display A, B and C in binary or hex, followed by their decimal equivalents; then continue with the expected result. Lastly, include a check statement that will show if the circuit output and the expected output matches. Use newline when you are going to print-out the results related to the next input pair. An example output is shown below (You do not have to use the exact same format, but the required elements should be there.):

   ```
   A=”bin=0100,dec=4”; B=”bin=0111, dec=7”; C_circ=”bin=1011, dec=11”; C_exp=”bin=1011, dec=11”;  status=TRUE
   A=”bin=0110,dec=6”; B=”bin=0111, dec=7”; C_circ=”bin=1111, dec=15”; C_exp=”bin=1101, dec=13”;  status=FALSE
   ```

   Print these results to the Tcl console as well (Hint: $display, $write, $monitor…).

8. Implement the design. Obtain the total LUT usage and the average power consumption of the circuit. What is the maximum clock frequency that your circuit can work correctly?

9. Intuitively discuss which path in your circuit would be the critical path. Did the result match your expectations?

10. Prepare a report that will include all your work & items asked in previous steps. You are expected to go by the experiment report format and rules given before in Ninova. Include a work package table to show specific tasks performed by each group member.

11. Your project submissions should include:
    - Archived Vivado project (must include all design codes, testbench codes, stimulus text file, test output text file…)
    - Test input generation code
    - Report in PDF format

ASSIGNMENTS

NOTE: You may use extra references (videos, websites, research papers, books etc.), just be sure to list them in your report.

Group Number	Assignment
Originals	Barrel Shifter with Logical/Arithmetic Right Shift and Rotate **References: [2]** **Exclusive Requirements** • B[4:0] should determine shift amount. B[6:5] should be used for extra control signals. Rest of B can be discarded. • Use logical operators and MUX modules with varying sizes. • "Carry" output should be set if the resulting number becomes zero due to a logical right shift • Be sure to include a test pair for each functionality at least once.
LSB	Barrel Shifter with Logical/Arithmetic Right Shift and Rotate **References: [2]** **Exclusive Requirements** • B[4:0] should determine shift amount. B[6:5] should be used for extra control signals. Rest of B can be discarded. • Use logical operators and MUX modules with varying sizes. • "Carry" output should be set if the resulting number becomes zero due to a logical right shift • Be sure to include a test pair for each functionality at least once.
MSB	Conditional Sum Adder **References: [1] pages 141-143, [5]** **Exclusive Requirements** • "Carry" output should give the carry-out bit of the last addition stage (traditional C_{out}) • Add an "overflow" output that will be set when overflow occurs in the circuit. In your testbench output text file and Tcl console, along with "C" output, monitor the value of overflow condition as well.
OD	Carry Select Adder **References: [1] pages 138-141** **Exclusive Requirements** • "Carry" output should give the carry-out bit of the last addition stage (traditional C_{out}) • Add an "overflow" output that will be set when overflow occurs in the circuit. In your testbench output text file and Tcl console, along with "C" output, monitor the value of overflow condition as well. • Discuss the effect of block bit sizes to the speed of the circuit. What size would you pick for the fundamental blocks? You may select different sizes for each block.
KARAM	Wallace Tree Multiplier **References: [1] pages 158-164, 215-218** **Exclusive Requirements** • Inputs A and B will be 16 bits long, and will be interpreted as unsigned numbers. • For additions, you may use any adding algorithm you like. Do not use "+" operator. • If viable, discuss the reason for the choice of the addition topology for the given bit lengths, in terms of overall performance.
DB	Brent-Kung Adder **References: [3],[4]** **Exclusive Requirements** • "Carry" output should give the carry-out bit of the last addition stage (traditional C_{out}) • Add an "overflow" output that will be set when overflow occurs in the circuit. In your testbench output text file and Tcl console, along with "C" output, monitor the value of overflow condition as well.

BOSS	Hybrid Carry-Lookahead/Carry-select Adder **References: [1] pages 138-141, 143-145** **Exclusive Requirements** • "Carry" output should give the carry-out bit of the last addition stage (traditional C_{out}) • Add an "overflow" output that will be set when overflow occurs in the circuit. In your testbench output text file and Tcl console, along with "C" output, monitor the value of overflow condition as well. • Discuss the effect of block bit sizes to the speed of the circuit. What size would you pick for the fundamental blocks? You may select different sizes for each block.
Nane-Limon	Array Multiplier **References: [1] pages 158-162, 226-230** **Exclusive Requirements** • Inputs A and B will be 16 bits long, and will be interpreted as unsigned numbers. • For additions, you may use any adding algorithm you like. Do not use "+" operator. • If viable, discuss the reason for the choice of the addition topology for the given bit lengths, in terms of overall performance.
MUSE	Carry Skip Adder **References: [1] pages 132-138** **Exclusive Requirements** • "Carry" output should give the carry-out bit of the last addition stage (traditional C_{out}) • Add an "overflow" output that will be set when overflow occurs in the circuit. In your testbench output text file and Tcl console, along with "C" output, monitor the value of overflow condition as well. • Discuss the effect of block bit sizes to the speed of the circuit. What size would you pick for the fundamental blocks? You may select different sizes for each block.
Leyla	Dadda Tree Multiplier **References: [1] pages 158-164, 215-218** **Exclusive Requirements** • Inputs A and B will be 16 bits long, and will be interpreted as unsigned numbers. • For additions, you may use any adding algorithm you like. Do not use "+" operator. • If viable, discuss the reason for the choice of the addition topology for the given bit lengths, in terms of overall performance.
Bitstream	Parallel Counter **References: [1] pages 158-162, 164-167** **Exclusive Requirements** • There is only one input A and input A will be 32 bits long, and will be interpreted as unsigned number. • Explain how many full adder stages are necessary for different bit-lengths (e.g. 16-32-64).

References:

[1] B. Parhami, Computer arithmetic - algorithms and hardware designs, Oxford University Press, (2010)

[2] Pillmeier, Matthew & Corporation, Unisys & Schulte, Michael & George, Eldose & Iii, Walters. Design Alternatives for Barrel Shifters. Proceedings of SPIE - The International Society for Optical Engineering. 4791. 10.1117/12.452034. (2002).

[3] Daphni, Samraj & Grace, Kasinadar. Design and Analysis of 32-bit Parallel Prefix Adders for Low Power VLSI Applications. Advances in Science, Technology and Engineering Systems Journal. 4. 10.25046/aj040213. (2019).

[4] B.Mounika, & A. RajKumar., Design of Efficient 32-Bit Parallel Prefix BrentKung Adder, Advances in Computational Sciences and Technology ISSN 0973-6107 Volume 10, Number 10, pp. 3103-3109 (2017)

[5] http://www.cs.columbia.edu/~cs4823/handouts/handout-16-F16.pdf