PROJECT DESCRIPTION & REQUIREMENTS

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

1. Each group will implement the combinational circuit assigned to them (See the tables at pages 2-4), according to the references provided and requirements given below.
2. Give an explanation of the algorithm/topology that's assigned to you.
3. Explain your design steps clearly. A reader should be able to follow your work along.
4. 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 );
```

5. 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 text file** that will be used in your testbench. You may use any programming language you like to create stimulus file.
6. Prepare a testbench for your circuit. It should be able to read all the test values from your stimulus file created at step 5, apply them to the circuit one by one, then write the corresponding results to another **output text file**. For each input pair, display them 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):

```
A="bin=0100,dec=4"; B="bin=0111, dec=7"; SUM="bin=1011, dec=11"; SUM_exp="bin=1011, dec=11";  status=TRUE
A="bin=0110,dec=6"; B="bin=0111, dec=7"; SUM="bin=1111, dec=15"; SUM_exp="bin=1101, dec=13";  status=FALSE
```

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

7. Implement the design. Obtain the total LUT usage and the delays of the circuit. What is the maximum clock frequency that your circuit can work correctly?
8. 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**.
9. Your project submissions should include:
   - Archived Vivado project (must include all design codes and testbench codes)
   - Test input generation code, stimulus text file, test output text file
   - Report in PDF format

Group Name	ASSIGMENT
untitled	32-bit Carry Select Adder: • It will perform signed integer arithmetic (both addition and subtraction) and will have two 32-bit inputs A and B, a 32-bit output SUM, and a single bit carry output COUT. • Add an "overflow" output that will be set when overflow occurs in the circuit. Along with the "SUM" output, monitor the value of overflow condition and output the exact value {cout, SUM} if overflow is set. References: [1] Behrooz Parhami, *Computer Arithmetic Algorithms and Hardware Designs*, 2nd ed. 2010, pp. 138–141. [2] "Carry-select adder," *Wikipedia*. https://en.wikipedia.org/wiki/Carry-select_adder
Betis cena	32-bit Carry Skip Adder: • It will perform signed integer arithmetic (both addition and subtraction) and will have two 32-bit inputs A and B, a 32-bit output SUM, and a single bit carry output COUT. • Add an "overflow" output that will be set when overflow occurs in the circuit. Along with the "SUM" output, monitor the value of overflow condition and output the exact value {cout, SUM} if overflow is set. References: [1] Behrooz Parhami, *Computer Arithmetic Algorithms and Hardware Designs*, 2nd ed. 2010, pp. 132–138. [2] "Carry-skip adder," *Wikipedia*. https://en.wikipedia.org/wiki/Carry-skip_adder
LUTs go	32-bit Kogge-Stone Adder: • It will perform signed integer arithmetic (both addition and subtraction) and will have two 32-bit inputs A and B, a 32-bit output SUM, and a single bit carry output COUT. • Add an "overflow" output that will be set when overflow occurs in the circuit. Along with the "SUM" output, monitor the value of overflow condition and output the exact value {cout, SUM} if overflow is set. References: [1] I. Koren, *Computer Arithmetic Algorithms*, 2nd ed. 2002, pp. 106–111. [2] "Kogge–Stone adder," *Wikipedia*. https://en.wikipedia.org/wiki/Kogge%E2%80%93Stone_adder
Bersel	32-bit Brent-Kung Adder: • It will perform signed integer arithmetic (both addition and subtraction) and will have two 32-bit inputs A and B, a 32-bit output SUM, and a single bit carry output COUT. • Add an "overflow" output that will be set when overflow occurs in the circuit. Along with the "SUM" output, monitor the value of overflow condition and output the exact value {cout, SUM} if overflow is set. References: [1] I. Koren, *Computer Arithmetic Algorithms*, 2nd ed. 2002, pp. 106–111. [2] "Brent–Kung adder," *Wikipedia*. https://en.wikipedia.org/wiki/Brent%E2%80%93Kung_adder

Group Name	ASSIGMENT
Paris	Barrel Shifter with Logical&Arithmetic Right Shift and Rotate • It will perform logical and arithmetic right shift, besides rotate operations. It will have 32-bit input A and 7-bit input B; 32-bit output RESULT. A will be the 32-bit number, B[4:0] should determine shift amount. B[6:5] should be used for extra control signals. RESULT will be the shifted version of A specified by B. • Use logical operators and MUX modules with varying sizes. • Be sure to include a test pair for each functionality at least once. References: [1] M. R. Pillmeier, M. J. Schulte, and E. G. Walters, "Design alternatives for barrel shifters," *Proceedings of SPIE, the International Society for Optical Engineering/Proceedings of SPIE*, vol. 4791, pp. 436–436, Dec. 2002, doi: https://doi.org/10.1117/12.452034.
500T	Hamming Weight Calculator using Parallel Counters • It will take the sum of each digit of 32-bit binary number (number of 1s, Hamming Weight). It will have 32-bit input DATA and 6-bit output RESULT. DATA will be the 32-bit number, RESULT will be the Hamming weight of the DATA. • Slice the 32-bit DATA 4 or 8 bits (4x8, 8x4) and implement the counter with separate blocks by using Full Adders and Half adders. Then take the sum of these separate blocks using an adder (+ operator can be used). References: [1] Behrooz Parhami, Computer Arithmetic Algorithms and Hardware Designs, 2nd ed. 2010, pp. 164–167. [2] "Hamming weight," *Wikipedia*. https://en.wikipedia.org/wiki/Hamming_weight
Alkolik Bilbao	Modular Leading Zero Counter • It will count the leading zeros of 32-bit binary number. It will have 32-bit input DATA and 6-bit output RESULT which provides the number of leading zeros of the given DATA. • Use Boundary Nibble Encoder (BNE) and Nibble Local Count (NLC) circuits which are specified in the reference [1]. References: [1] N. Z. Milenković, V. V. Stanković, and M. Lj. Milić, "Modular Design Of Fast Leading Zeros Counting Circuit," *Journal of Electrical Engineering*, vol. 66, no. 6, pp. 329–333, Nov. 2015, doi: https://doi.org/10.2478/jee-2015-0054.
AND	Booth's Encoder Multiplier • It will perform signed multiplication and will have two 8-bit inputs A and B; besides, a 16-bit output PRODUCT which provides the multiplication result of the given numbers. It will use Booth's Radix-4 Encoding algorithm to reduce the number of partial products by half. • Generate/modify the partial products according to Booth's encoding table. Then take the sum of these reduced newly generated partial products using adders (+ operator can be used). The circuit should be purely combinational. References: [1] Behrooz Parhami, Computer Arithmetic Algorithms and Hardware Designs, 2nd ed. 2010, pp. 187-189, 200–202.

Group Name	ASSIGMENT
EM2	Hamming (7,4) Encoder/Decoder • It will perform Hamming (7, 4) Encoding Algorithm [1] to encode/decode input DATA. The encoder part has 4-bit input RAW_DATA and 7-bit output ENC_DATA; the decoder part has 7-bit input ENC_DATA and 4-bit output DEC_DATA. • Use logical operators and precalculated codewords to design. • Be sure to include test pairs for both encoding and decoding. • Do research and explain the applications and methods of error correction codes. References: [1] "Hamming code," *Wikipedia*. https://en.wikipedia.org/wiki/Hamming_code [2] "Design of (7, 4) Hamming Encoder and Decoder Using VHDL," Sep. 2015, Available: https://www.researchgate.net/publication/281462120_Design_of_7_4_Hamming_Encoder_and_Decoder_Using_VHDL
-	Block Interleaver/Deinterleaver • It will perform Block Interleaving/Deinterleaving [1] to a serial data. There will be 2-bit two parameters ROW and COL. ROW will indicate the number of rows, and COL will indicate the number of columns to perform interleaving (00->4, 01->1, 10->2, 11->3). The size of the inputs and outputs should be determined by the parameter SIZE (ROWxCOL). • The interleaver part should have an input RAW_DATA and an output INT_DATA with the lengths of SIZE; the deinterleaver part should have an input INT_DATA and an output DINT_DATA with the lengths of SIZE. • Be sure to include test pairs for all parametric cases and both interleaving and deinterleaving. • Do research and explain the applications and methods of burst error correction codes. References: [1] "Burst error-correcting code," *Wikipedia*. https://en.wikipedia.org/wiki/Burst_error-correcting_code#Interleaved_codes

Contact me, if you have questions about the definitions of your assignment.

SERDAR DURAN