Homework #3

Start assignment as soon as possible. Work individually, but you can ask your classmates for help when you get stuck without sharing any code. Please send me email if something isn't clear and I will update the assignment.

Notes:

- Clearly state whether your design is fully functional, and state the failing sections if any exist.
- Make sure your design and code are easily readable and understandable (clear and well commented).
- For all problems, perform the required test(s) and submit a table printed by your verilog testbench module listing all inputs and corresponding outputs.

Keep “hardware” modules separate from testing code. Instantiate a copy of your processing module(s) in your testing module (the highest level module) and drive the inputs and check the outputs from there.
1. [35 pts] The purpose of this problem is to familiarize you with the synthesis and place and route process and to give you a rough feeling for the size of a few simple circuits. Turn in a detailed report with the results for each part.

- For all designs register the inputs and outputs, and add timing constraint for the clock (e.g. 550 MHz). Report slack (set-up and hold time if available). If it couldn’t make the timing constraint, slow down the clock until it meets the timing.
- Synthesize and place and route the following blocks and report their total slice count, and other FPGA resource counts that are listed in Summary Report (this includes all logic utilization numbers under Device Utilization Summary). Include registers (flip-flops) and constrain the timing path as explained in the class. Assume words are all 2’s complement signed unless stated otherwise. No need to simulate, only turn in the source verilog, but your verilog must compile correctly (Use “Check Syntax” in Synthesis flow).
- Report delay for each block using Place and Route Static Timing Analysis (under Place and Route). Slack must be a minimum positive number (zero or near zero e.g 0.1 ns).
- Using the XPower analyzer tool (under Place and Route), report the power dissipation of each block for your clock specification that the design can operate. Note that for power number, your design clk must be set to the number that you found in previous step, and place and route must be done again.
- Report results for all circuits in a table. Report the numbers in a single table so it can be used as a note sheet in the future.
- Include verilog files for each circuit in the report.

**Blocks**

a) [10 pts] two input 10-bit adder (11-bit output). Use "+" in verilog.

b) [10 pts] an adder which adds 25 6-bit numbers using verilog "+" (i.e., something like, assign out = in0 + in1 + in2 + ...) and produces a 6-bit sum.

c) [15 pts] 16-bit x 16-bit unsigned multiplier (32-bit output). Use "*" in verilog with two different options
   i. Make the synthesis use LUT for implementation and report timing, slice count and power results
   ii. Make the synthesis use Multiplier Blocks for implementation and report timing, slice count and power results
2. [110 pts] Design a circuit that compute the product of 128 element-vectors, a and b; that is a vector p such that \( p_i = a_i * b_i \). The elements of a and b are stored in separate SSRAMS and the result is to be written into a third SSRAM. Assume that computation is started by a control signal, go being one during a clock cycle and output control signal, done is to be set to one during the cycle when the computation is complete. Assume a and b are 16-bit values.

**Design**

i. [10 pt] Draw the detailed block diagram with naming the signals

ii. [40 pt] Design a finite state machine in verilog for controlling the signals. Report 1. Control Sequence/steps (page 40 of State Machine slides), 2. Table for control signals (Page 41) and 3. State diagram or Transition Function (Page 44 and 49) based on the required steps in separate tables (as shown in the class).

iii. [20 pt] Write a verilog model for the SSRAM memories and write the verilog for the complete circuit.

iv. [20 pt] Modify your design in iii to Xilinx Core Generator to generate the SSRAM memories as shown in the class. Instantiate the cores in your verilog design and use the same top file that you used in iii.

**Behavioral Simulations**

v. [20 pt] Test the design in part iii and iv separately using your testbench and include all your test results.
3. [80 pt] This problem was explained in the class: Design a FIFO to store up to 256 data items of 16-bits each, using 256x 16-bit dual-port SSRAM for the data storage. Assume the FIFO will not be read when it is empty, not to be written when it is full, and that the write and read ports share a common clock.

vi. [10 pt] Draw a detailed block diagram of your design.

vii. [20 pt] Develop a verilog model for the FIFO example. You can use a state machine for the design.

viii. [20 pt] Write a testbench and test the design and identify the cases that can verify the FIFO works.

ix. [30 pt] Develop the FIFO design using Xilinx IP Core Generator and test the FIFO using the same testbench that you developed in ii.