Homework 6

1. Normally, the program counter (PC) must count up 4. But, it must also have branch and jump control inputs as shown in lecture slides page 28 of "ITU_VLSI_II_RF_Datapath_SingleCycle_Control.pdf" slides.
    a. Design branch controller as shown in lecture slides page 28 of "ITU_VLSI_II_RF_Datapath_SingleCycle_Control.pdf" slides.
    b. Design PC module as shown in lecture slides page 28 of "ITU_VLSI_II_RF_Datapath_SingleCycle_Control.pdf" slides.
    c. Synthesize the top-level code of your PC within OpenLane flow. By using the generated post synthesis reports, obtain total cell usage, estimated area, critical path delay and estimated power consumption of the design.
    d. Perform behavioral simulation for your PC by using iverilog and gtkwave. Aim to test all functionalities of your PC. Clearly explain your simulation results. Having a self-checking testbench with ample amount of test operands is encouraged.
    e. Include all Verilog codes, testbench codes related to your PC in your report.
2. Design Instruction Decoder (ID) block of your processor which translates the instructions to the control world as shown in lecture slides page 28 of "ITU_VLSI_II_RF_Datapath_SingleCycle_Control.pdf" slides.
    a. Design ID block as shown in lecture slides page 28 of "ITU_VLSI_II_RF_Datapath_SingleCycle_Control.pdf" slides.
    b. Synthesize the top-level code of your ID within OpenLane flow. By using the generated post synthesis reports, obtain total cell usage, estimated area, critical path delay and estimated power consumption of the design.
    c. Perform behavioral simulation for your ID by using iverilog and gtkwave. Aim to test all functionalities of your ID. Clearly explain your simulation results. Having a self-checking testbench with ample amount of test operands is encouraged.
    d. Include all Verilog codes, testbench codes related to your PC in your report.
3. Design control unit (CU).
    a. Integrate the PC and the Instruction Decoder in order to design the CU. As stated in step 3, instruction memory will be connected over from a testbench, so include any required memory I/O ports in your CU as well.
    b. Synthesize the top-level code of your CU within OpenLane flow. By using the generated post synthesis reports, obtain total cell usage, estimated area, critical path delay and estimated power consumption of the design.
    c. Create a testbench and connect your CU and Instruction memory. Prepare an initialization file for your instruction memory that will include at least one instruction from all types of instructions. By using $readmemb function, load this file to Instruction memory.
    d. Perform behavioral simulation for your CU by using iverilog and gtkwave. Clearly explain your simulation results. Having a self-checking testbench with ample amount of test operands is encouraged.
    e. Include all Verilog codes, testbench codes related to your PC in your report.
4. Convert your single cycle processor to a 5-stage pipelined RISC-V processor by using the structure shown on page 18 of lecture slides "ITU_VLSI_II_Computer_Organization_Pipeline.pdf" as basis. Note that your design might be different from this example. You will first design your processor without any hazard handling.
    a. Explain your pipeline design and all other work that you have done for implementing the 5-stage pipelined structure.
    b. Draw a detailed structure of your entire design. You may use horizontal page layout.
    c. Write an assembly code for the algorithm shown in Fig. 1.
    d. Convert your assembly code to machine code.
    e. By using $readmemb function, load this file to Instruction memory.

f. Perform behavioral simulation for your pipelined core by using iverilog and gtkwave. Clearly explain your simulation results. Having a self-checking testbench with ample amount of test operands is encouraged.

g. Because you do not have any hazard handling in your design, the simulation result may not be correct. Explain your findings and try to solve the problem (if there is any) by introducing NOP operations where you think that will be useful.

Note: If there are needed changes in your FU design from the previous homework, please correct them in this homework.

Please write your reports as a group homework. Do not forget to add cover page and page numbers. Try to write the report regularly and well organized! You may use tables, explanations, figures, drawings, etc.

Left-to-right modular multipcation algorithm

Input: $A=(a_{k-1},a_{k-2},\cdots,a_1,a_0)_2$, $B=(b_{k-1},b_{k-2},\cdots,b_1,b_0)_2$, $N=(n_{k-1},n_{k-2},\cdots,n_1,n_0)_2$

Output: $C=AB \bmod N=(c_{k-1},c_{k-2},\cdots,c_1,c_0)_2$

Step1: C=0
Step2: for i=k-1:0
Step3: C=2C
if C≥N
Step4: C=C-N
Step5: if b_i=1
Step6: C=C+A
if C≥N
Step7: C=C-N

Figure 1