**Module Description:**

The microprocessor is a mini MIPS (Microprocessor without Interlocked Pipelined Stages) designed to process four instruction pair of which are R type and I type. The processor is an 8 bit word format where three MSBs are the opcode and two bits forms the weightage value of both destination and source registers while the last 3 LSBs are not used in R type while forms the offsets for I type. The following are the TopLevel module which working concurrently forms the complete processor.

1. Adder8Bit UUT1: (UUT denotes the unit under test, used while naming the instantiation).

The Adder used is an 8 bit adder used to add the address value from the Program counter to increment by 1 unit. Adder8Bit module is a clock initiated module that would increase the address at each clock cycle.

1. PC UUT2:

PC or Program Counter is a register that passes the data from the input to the output at each clock cycle of the system. The PC or the Program counter is used to initialize the address of the Instrutction memory and point at each clock cycle which instruction to be initialized. The PC is a reset and enable module that marks the beginning of the Microprocessor design.

1. InstructionMemory UUT3:

Instrctionmemory is a memory unit that defines here as a simple ROM function. Each memory allocates a unique instruction that is pointed out by the PC, hence at each clock cycle the particular instruction needs to be encountered. The four unique instruction stored here are Store word into a memory , Load Word into a Register , ADD and Subtract the values of two registers and store in a register.

1. Control\_Unit UUT4:

Control unit is heart of the processor that send particular signal to each module to act according to the instruction required. Here the main important signals that manages the low and high signals following the opcode are write\_enable, aluop, alusrc, mem\_write, memtoreg. Write enable makes the RegData available to write the values from the instruction pointer to the registers (here temporary registers are defined as Temp0 and Temp1) whose output is being read at the readdata1 and readdata2. The aluop signal is used as a select line for and ALU to ADD or SUB when it is either low or high. The alusrc is a signal for a Multiplexer (Mux2to1\_8Bit) which when allows the instruction grouped to gather to perform such as ADD and SUB is R type made available when the alusrc is high and when Store and Load(SW and LW) Instruction type or I type are encountered when alusrc is 0.

1. RegData UUT5:

RegData is a module comprises of Registers which store and load data as when required by the control signals. Here initially the RegData acts as a Decoder by converting the binary unit value of either one or zero to its weightage value in 8 bit binary. The write enable signal when high makes the register equivalent values available at the output of two register regdata1 and regdats2.

1. SignExtend UUT51

SignExtedn module is required only when I type instruction is being encountered. SignExtend module extends the 3 LSBs of the word which are called as offsets to make available by converting into 8 bits. These 8 bits are formed by concatenating the offsets binary numbers with their Sign bit extended to 8 bits. The out of the SignExtend makes the other input of a Multiplexer near Alu in order to make the difference between the R and I type.

1. Mux2to1\_8Bit UUT52:

The multiplexer is used at two places inside the processor. The first time it acts as the select line for R and I type instruction that needs to flow through ALU for either adding/ subtracting two values or loading/storing the data into the selected memory address.

1. ALU\_8bit UUT6:

Arithmetic and Logical unit does the addition and subtraction of two values. At once instance for R type the two inputs are the two values while for I Type that while storing and loading the values the two inputs become the address where the second input value needs to be stored.

1. Data\_Memory UUT7:

Data\_Memory is the main Memory of the processors where the values can be stored or retrieved (Load) into the different registers as when required, here the mem\_write signal does the two operation when it is low the signal is read and send to the output while when it is high the data from the input is written inside the memory with the address allocated from the Alu.

1. Mux2to1\_8Bit UUT8:

The mux used at the output of Data\_Memory works on the signal select line memtoreg which when high reads the value from the memory output and send as the write data into the RegData, if it is low the readdata2 value would be retrieved from the multiplexer again to the Regdata.

Operation:

The following are the fours instructions that the processor is responsible to process and show the results as required. For the following instruction $t0 is register 1 and $t1 s register two. The instructions are as follows

1. sw $t0,1($t1)
2. lw $t1,1($t1)
3. add $t0,$t1,$t0
4. sub $t1,$t0,$t1

1) sw $t0,1($t1): this is an I type instruction, here the register apparent value of $t1 is added with 1 which is of three bits only. Therefore the sign extension is required to convert it into 8 bits. Now this value is added to the address generated by register 2. The value of register 1 is stored at this memory address in the memory. Here the following signals are active write\_enable=1'b0; aluop=1'b0; alusrc=1'b1; mem\_write=1'b1; memtoreg=1'b1;

2) lw $t1,1($t1): This is also another type of I instruction where the process is similar to the store word, but here the last multiplexer after the memory is active and the data after relaying on the memory address gets transferred to the register in the register memory i.e. DataReg. Here as compared to the previous instruction the storing value is written into the register 1. The following signals from the control unit are active as shown. write\_enable=1'b0; aluop = 1'b0; alusrc=1'b1; mem\_write=1'b1; memtoreg=1'b0

3) add $t0,$t1,$t0 : this is the first type of R instruction, here the register values of first register is added with second register and stored in the first register itself. Careful manipulating of the RegData is considered here and the following signals are kept intact

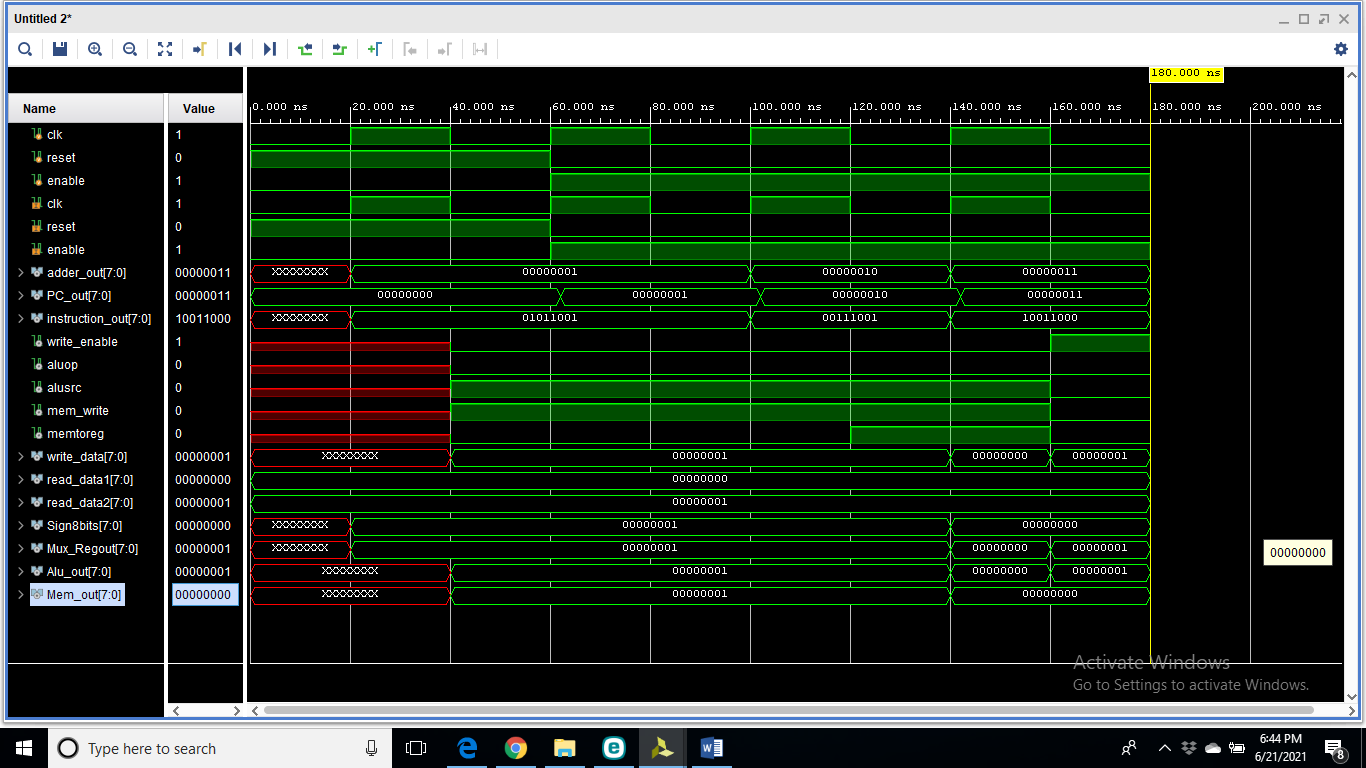
write\_enable= 1'b1; aluop = 1'b0; alusrc=1'b0; mem\_write=1'b0; memtoreg=1'b0;

4)sub $t1,$t0,$t1: this is similar to the add instruction lest used in subtracting the value of register two which gets subtracted from register one and the result being stored in register two. Except aluop which is high for subtraction all the reaming signals are same as shown.

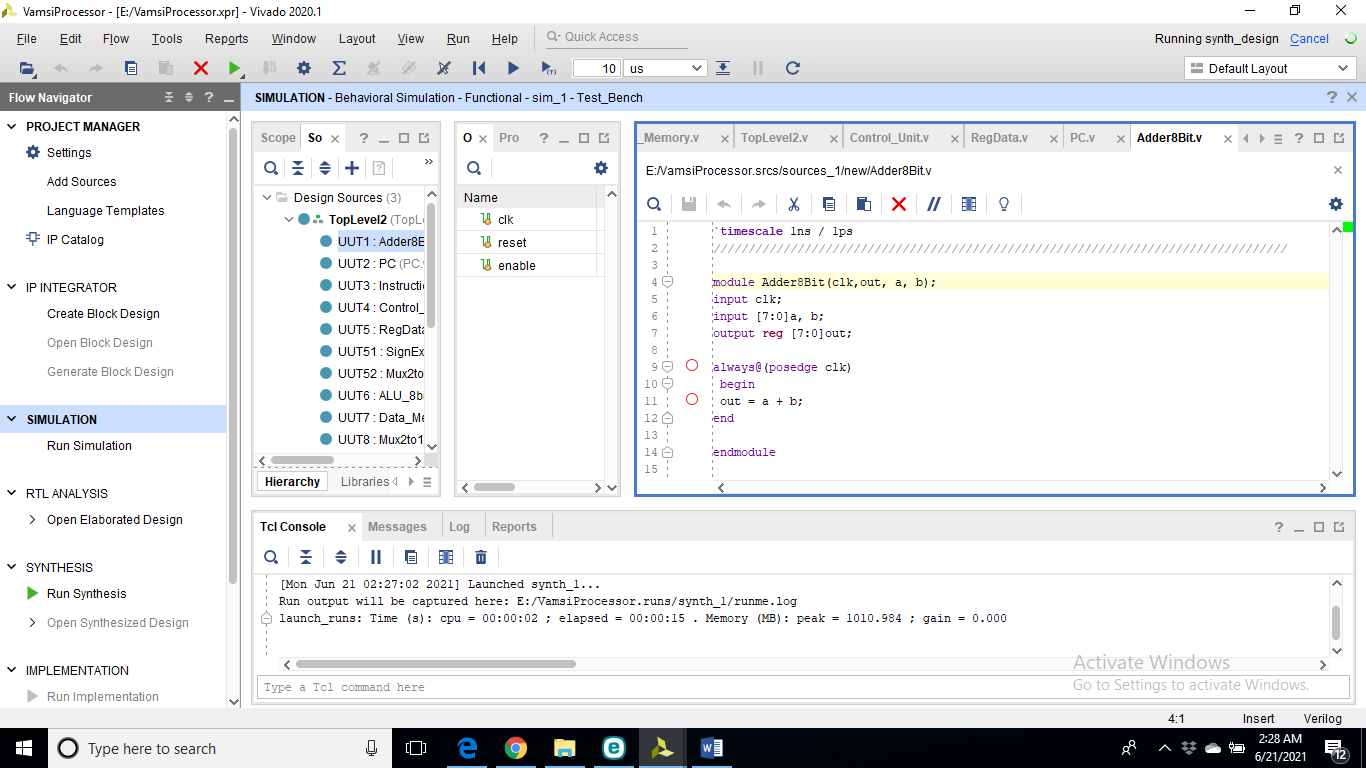
write\_enable=1'b1; aluop=1'b1; alusrc=1'b0; mem\_write=1'b0; memtoreg=1'b0;

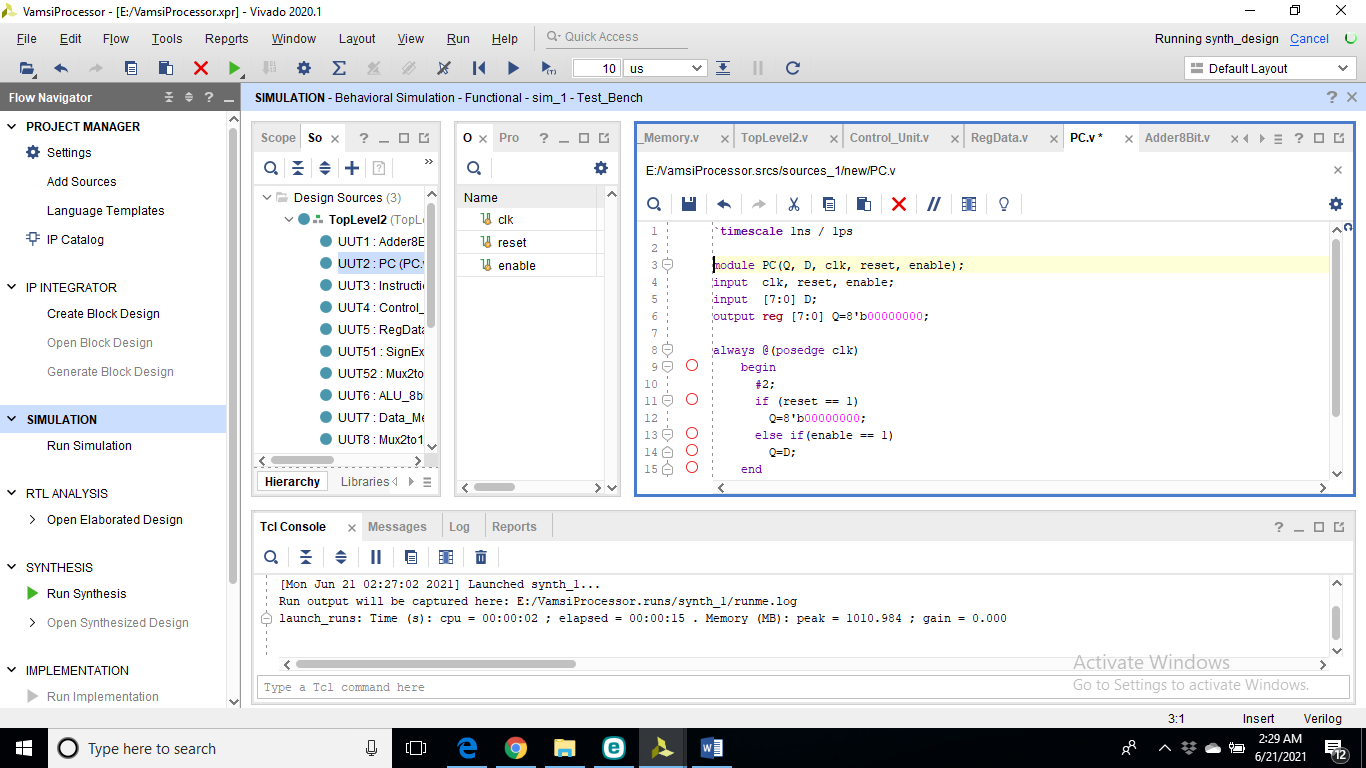
Simulation and Schematics:

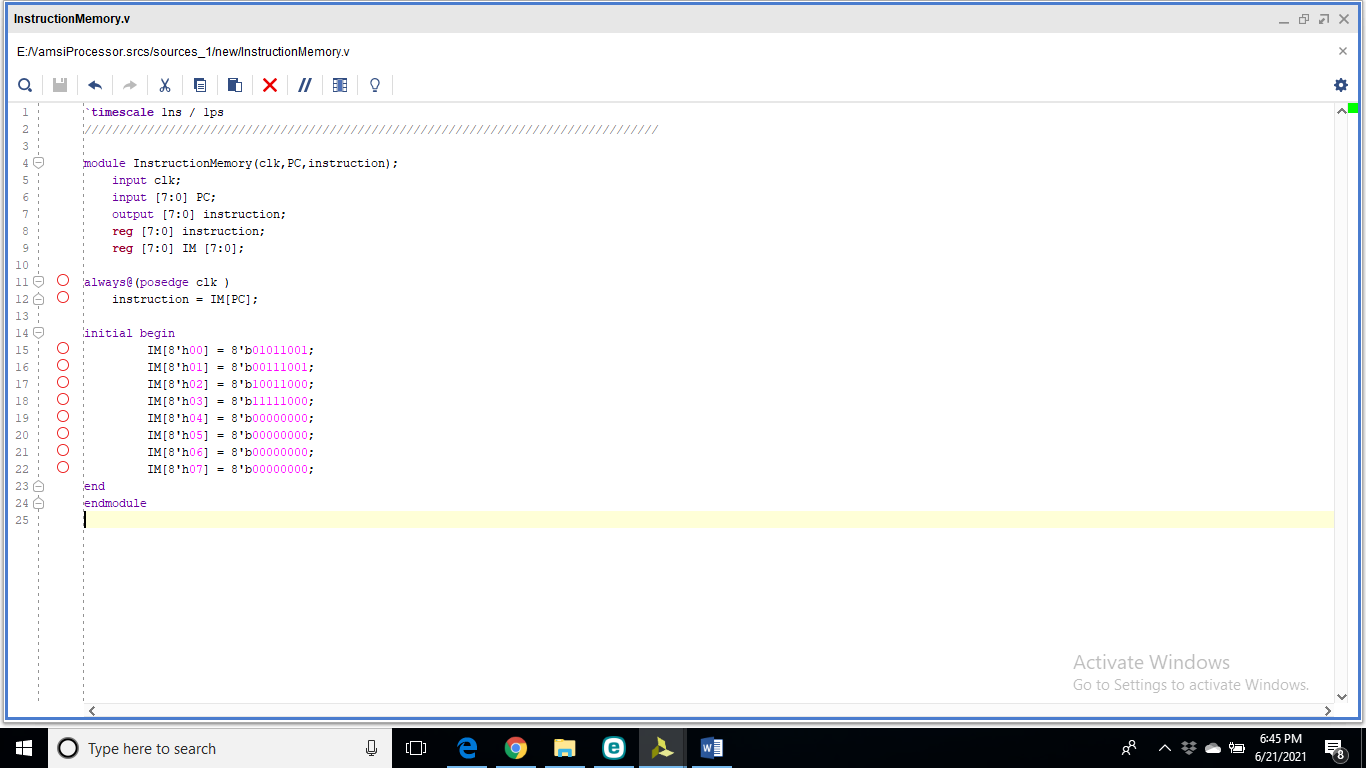
Simulation:

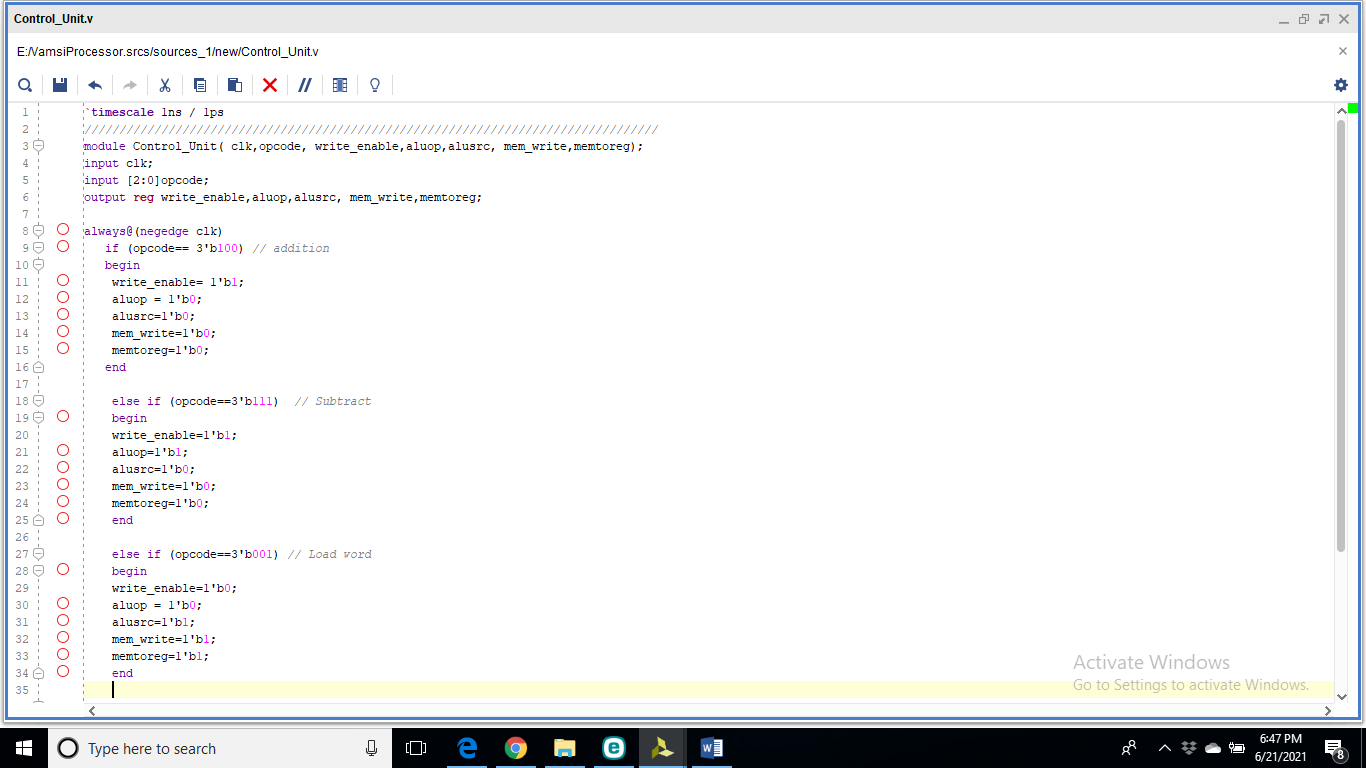


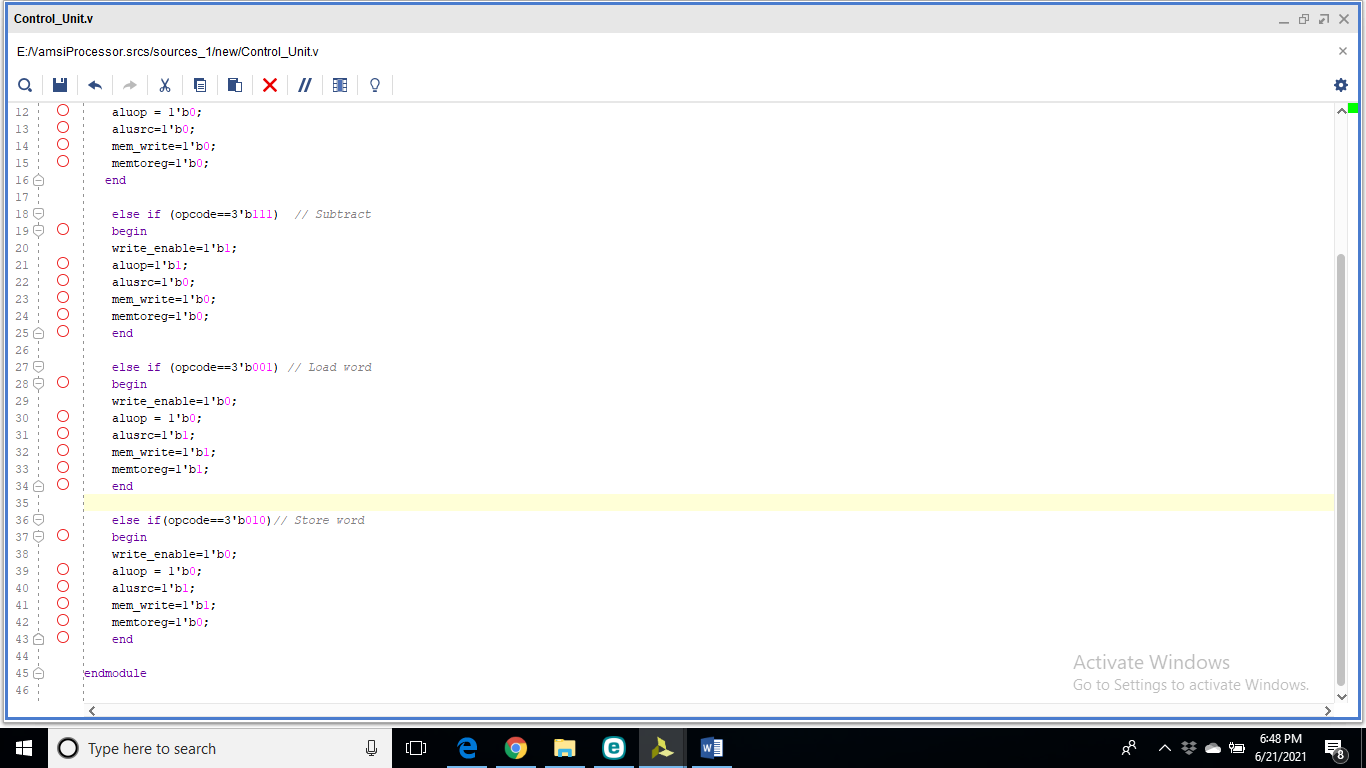
Codes:

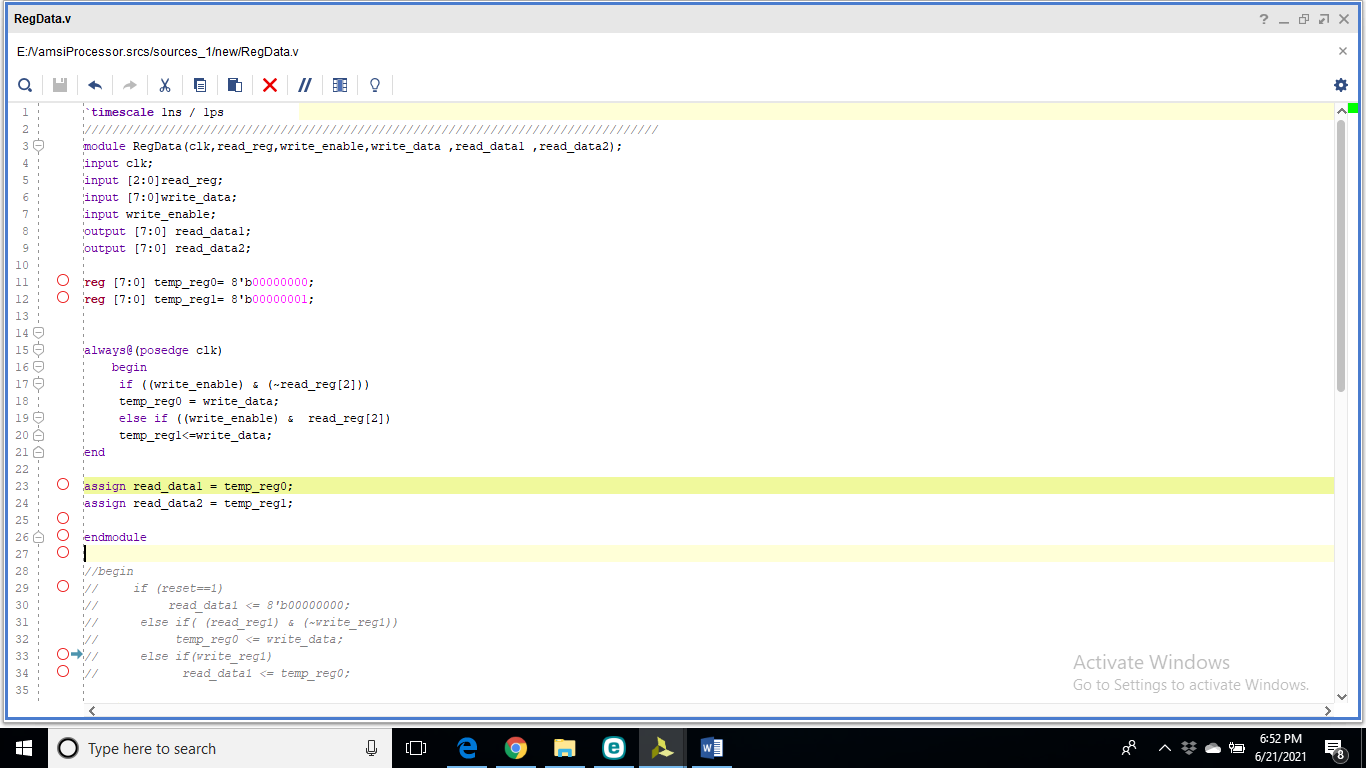


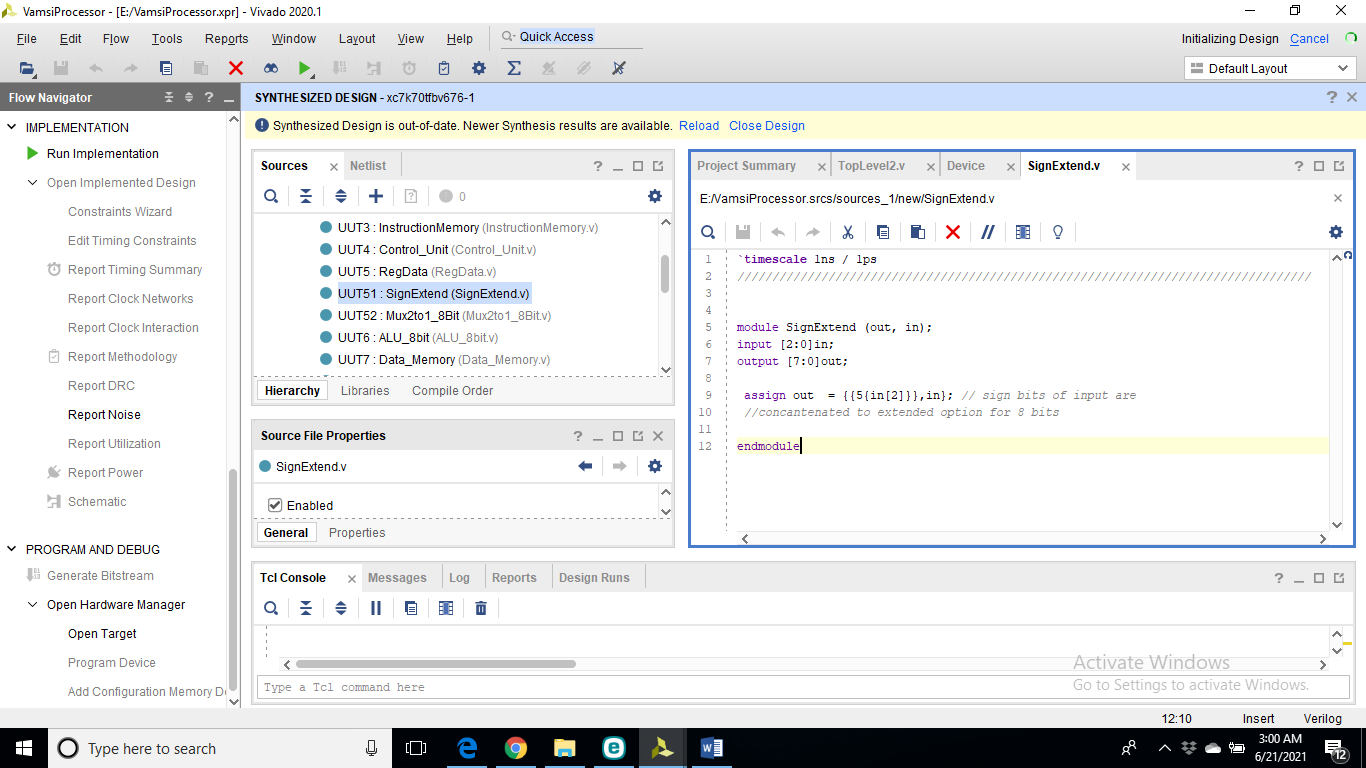


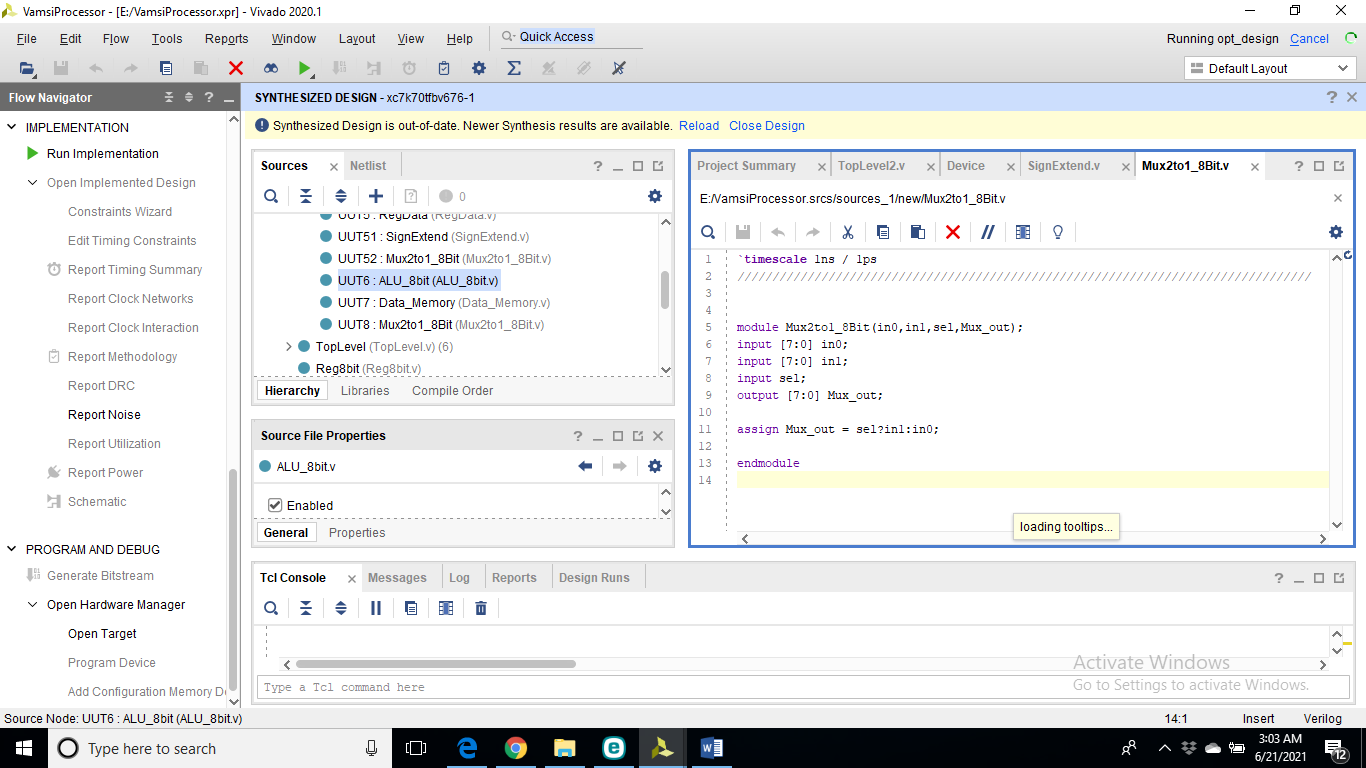


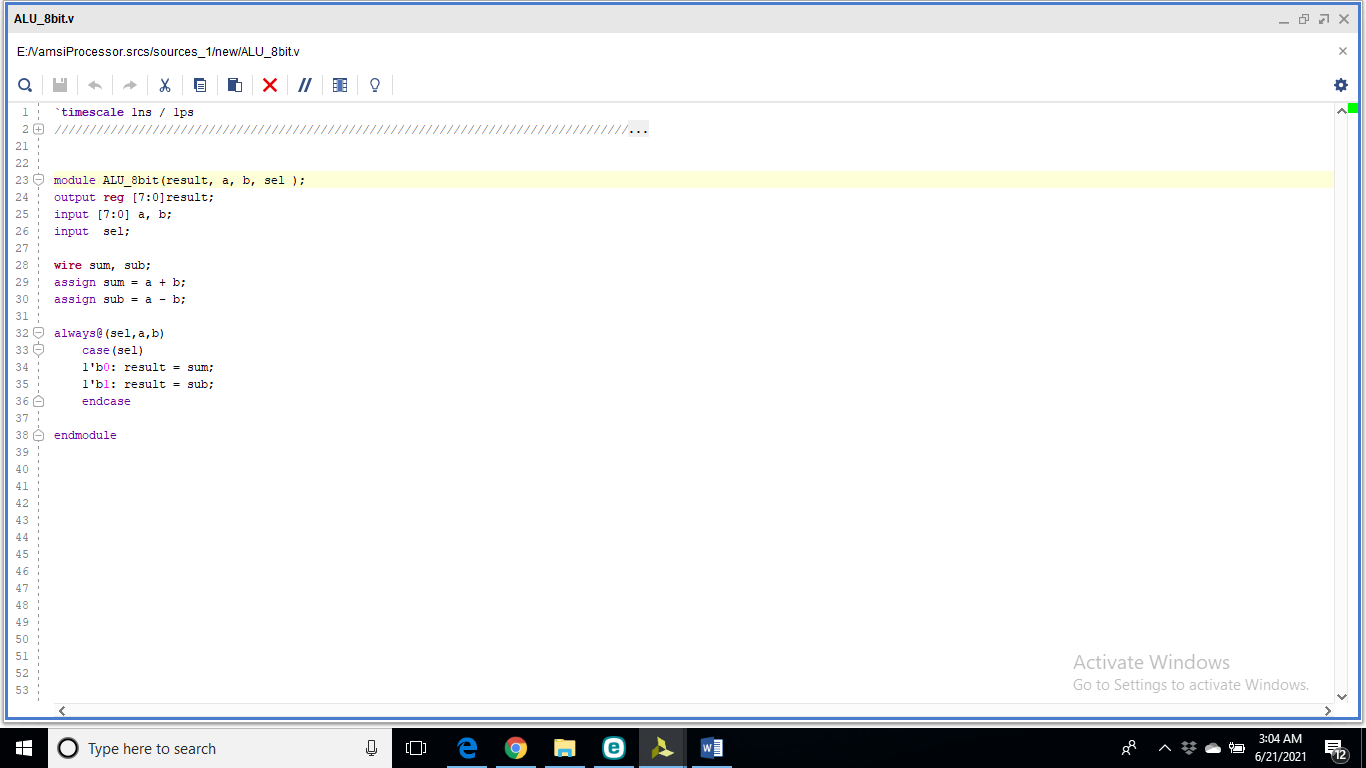


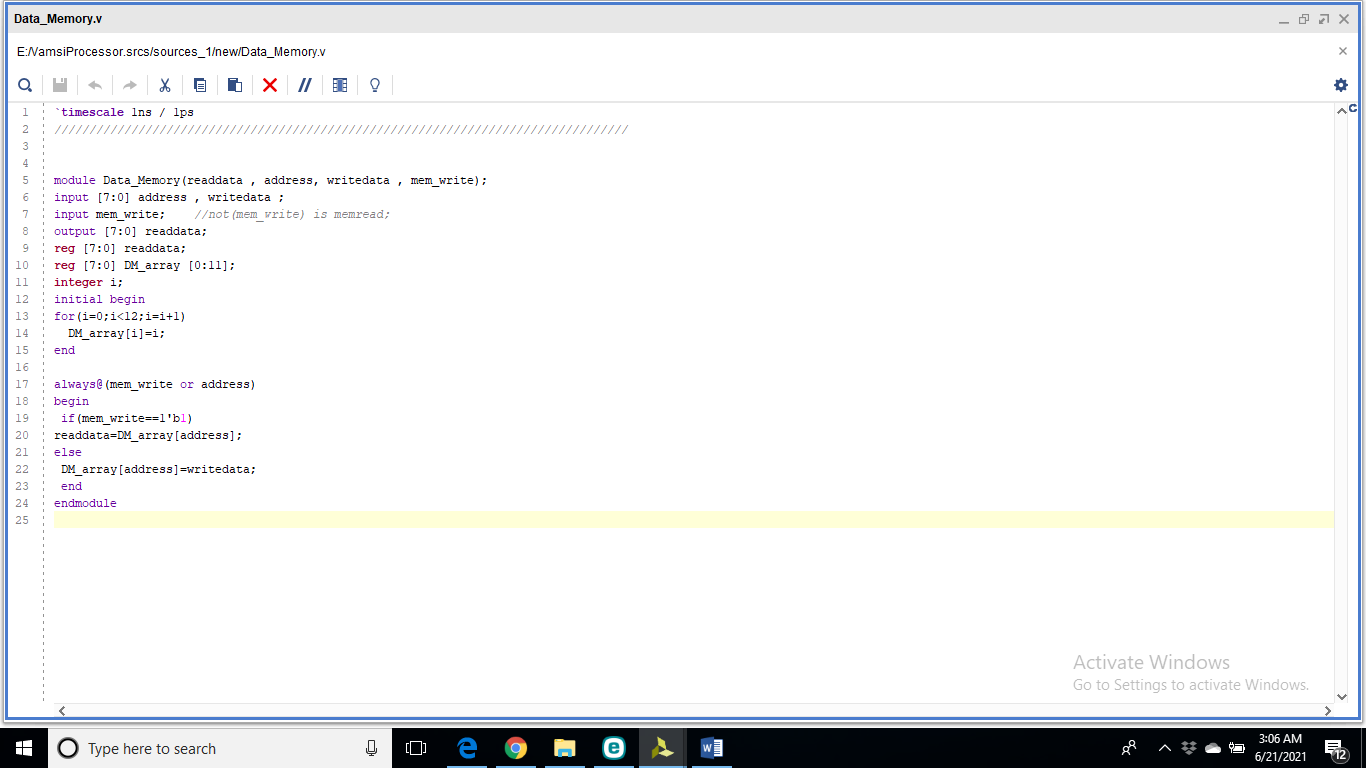




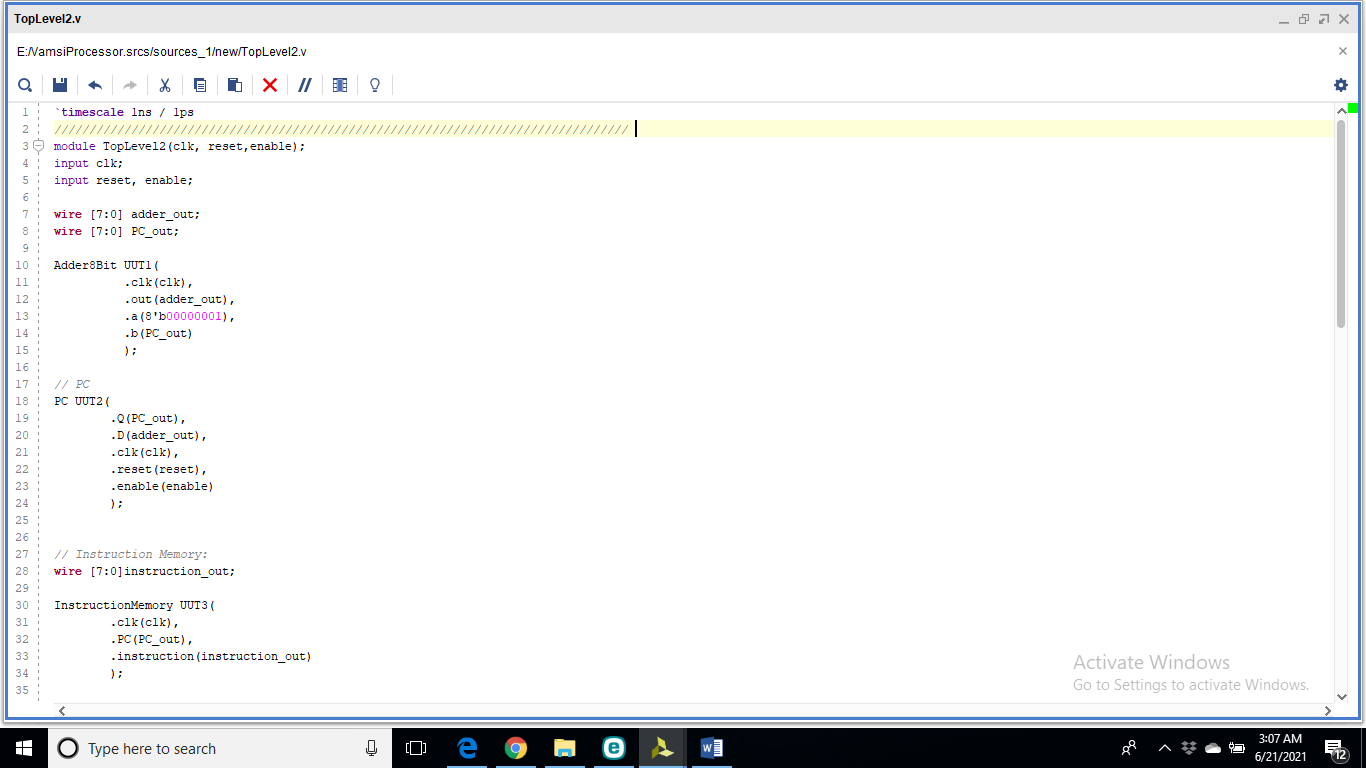


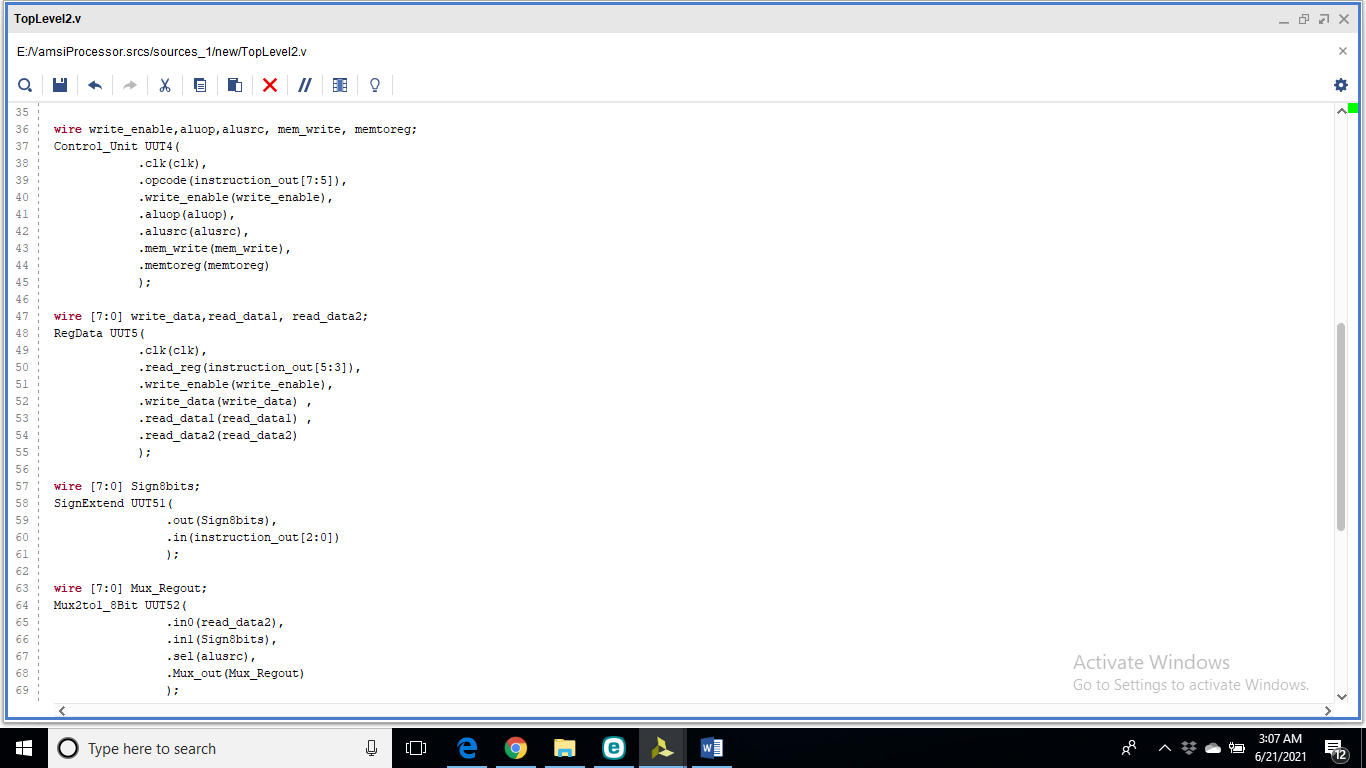


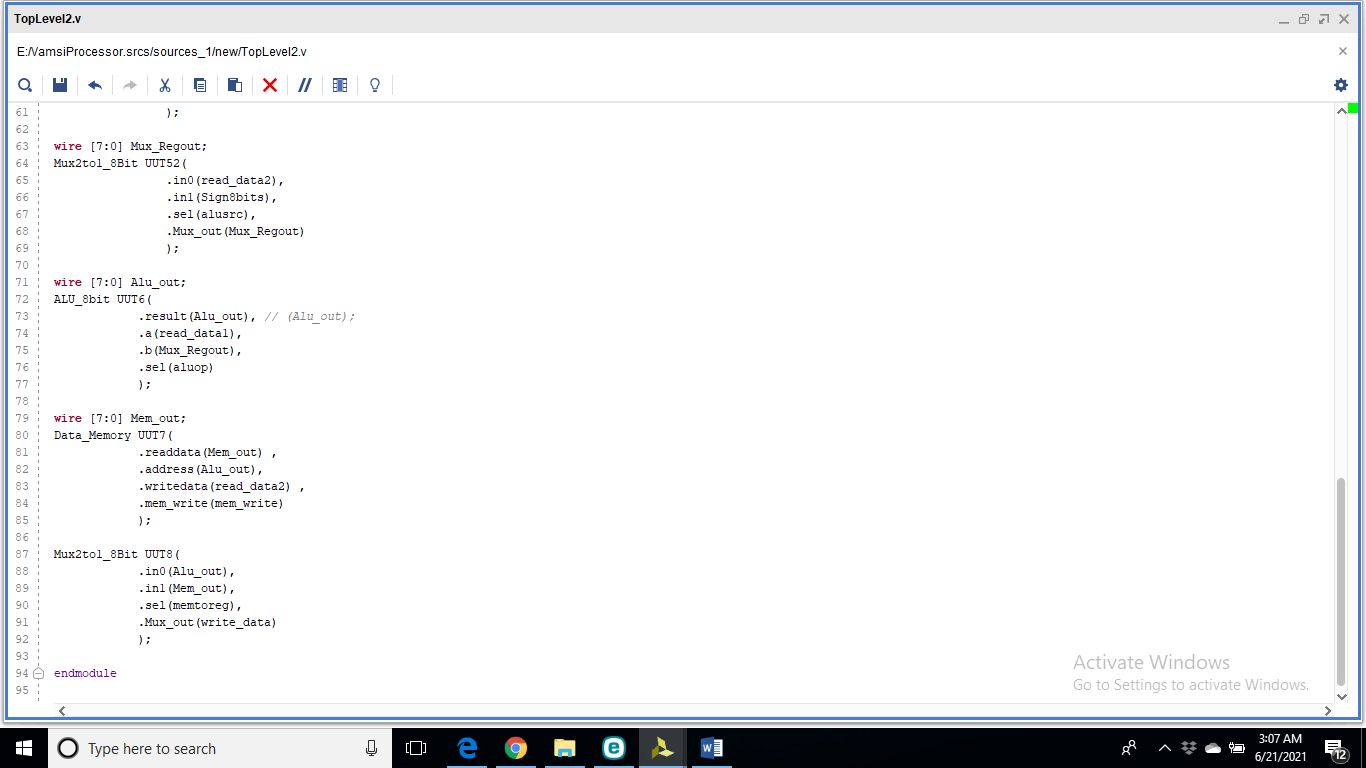




TopLevel2:







Schematics:

