

**ESE 345 - Computer Architecture**

**Prof. Mikhail Dorojevets**

***Pipelined Multimedia Unit***

**By**

**Wilmer Suarez**

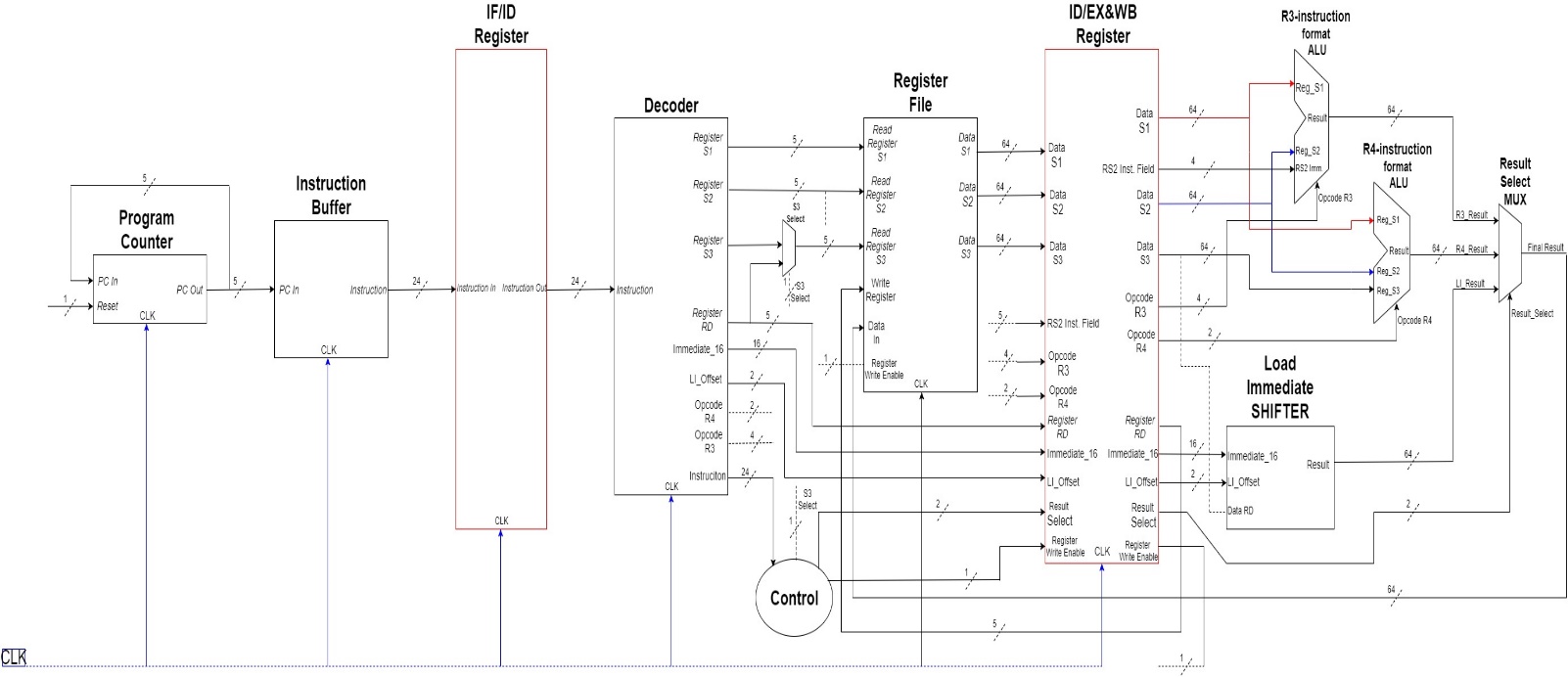
**&**

**Himanshu Goel**

**Goals**

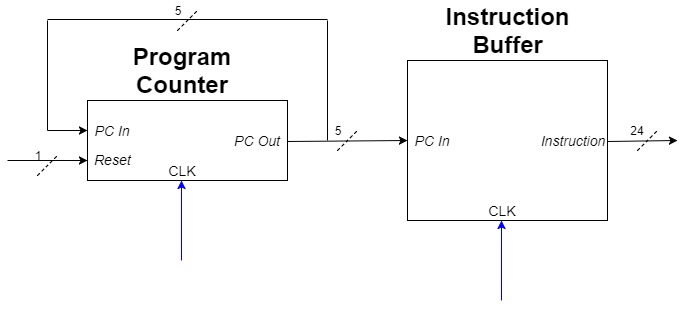
The primary goal of this project was to demonstrate an understanding of the material covered in this class through the implementation of a 3-stage pipelined multimedia processing unit for Single-Instruction Multiple-Data operations.

**Multimedia Unit Block Diagram**

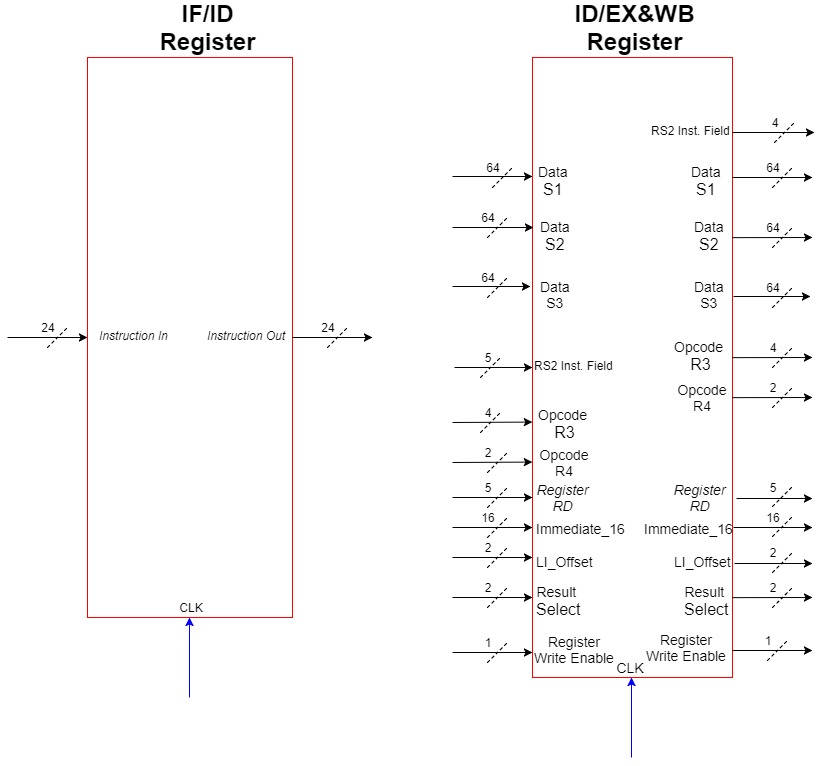


**Design Process**

**Program Counter and Instruction Buffer**

­­­­

**Pipeline Registers**

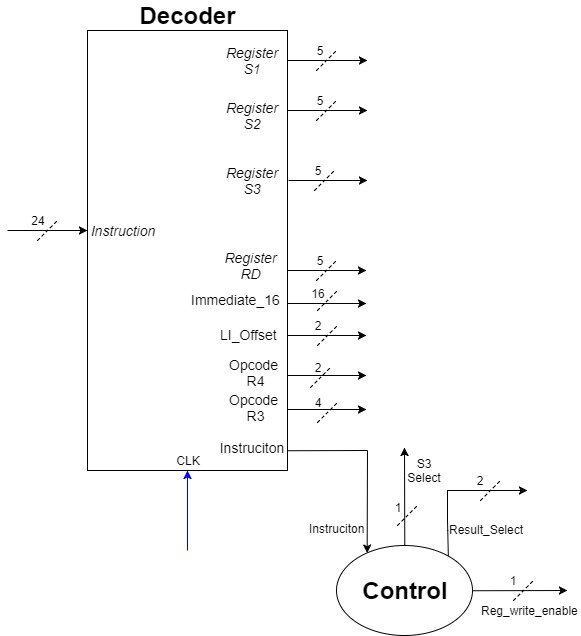


The pipeline registers, IF/ID and ID/ED&WB, are used to separate the 3-stages of the datapath. The registers are clock edge-sensitive (positive edge) to allow one instruction to be processed at every stage, and one instruction to be completed every cycle (after the initial 3 cycles).

This 3-stage design allows us to ignore the need to fix **Data Hazards** (instructions needing data before it is ready). By the time an Instruction is decoded (in the second stage), the data that it might need is already calculated by the third stage. **Structural Hazards** (attempting to use the same resource simultaneously by two instructions) only occur with the Register file, when an instruction is trying to read a register while another is trying to write to the same register. This hazard is resolved by writing to the Register File on the rising edge of the clock first and then reading from it at the falling edge of the clock; allowing the more recent instruction to receive the most up-to-date data. Lastly, **Control Hazards** (attempting to make decisions before a condition is evaluated) do not occur in this design because there are no branching instructions.

The first register receives instructions from the instruction buffer in the *Instruction Fetch (IF) Stage* and outputs an instruction after every positive edge of the clock. The instruction output is sent to the Decoder in the *Instruction Decode (ID) Stage*. The second register receives input from the Decoder, the Control Unit, and the Register File. From the Register File, it receives three 64-bit signals of the data from the three selected registers. From the Decoder, it receives the instruction opcodes and the associated signals needed for those instructions. From the Control Unit, it receives the 2-bit Result Select signal needed to select which output from the *EX* stage will be written back to the Register File, and the Register Write Enable.

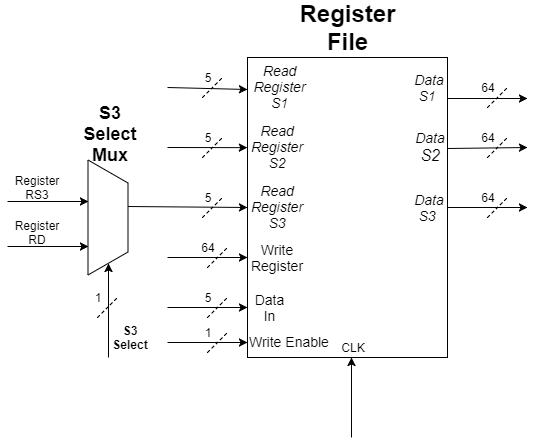
**Instruction Decoder and Control Unit**



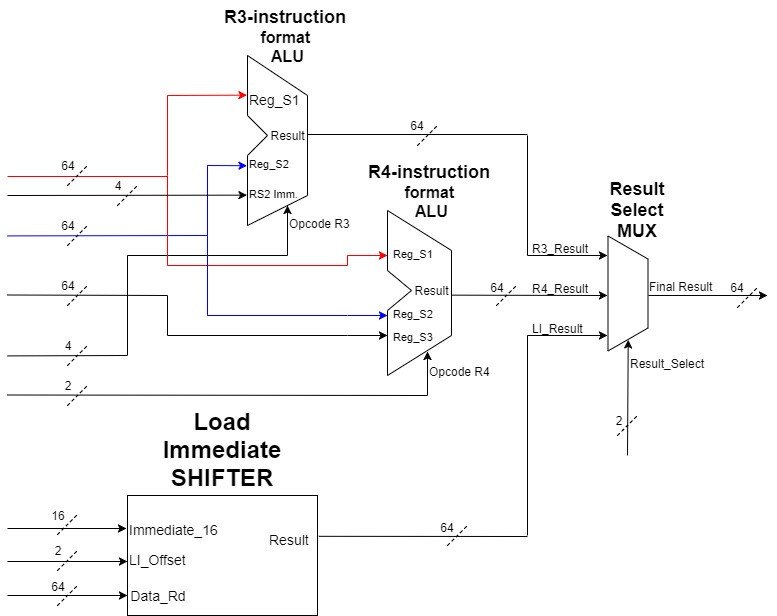
The Decoder (in the *Instruction Decode (ID) Stage*), receives an instruction from the *ID/IF* pipeline register. The Decoder generates subsets of the instruction received to use as input for the Register File, Control, and IF/EX&WB Register.

The Control Unit receives the Instruction (unchanged) from the Decoder and outputs 3 signals. The first signal, *S3\_Select*, is used when the instruction is a Load Immediate. This signal is sent to a multiplexer that is between the Decoder and the Register File that selects which Register File address input RS3 will receive. This is used to send the data of the writeback register to the Load Immediate Shifter in the *EX&WB* stage. The second signal, *Result\_Select* is used to select witch result from the *EX&WB* stage will be written back to the Register File. The last signal, *Reg\_Write\_Enable*, is used to allow data to be written to the Register File or not. The signal only prevents data to be written to the Register File during a NOP instruction.

**Register File**



**ALU 1, 2 and Load Immediate Shifter**



The *EX&WB* stage is the last stage of the pipelined design. This stage computes the values that will be written to the register file. The stage is composed of 2 ALUs and a separate module for the Load Immediate Instruction. The signals coming into this stage all come from the *ID/EX&WB Register*.

The first ALU is used to compute the R3-format instructions. This ALU only uses the first 2 of the three register data outputs from the pipeline register. The two other inputs to this module is the RS2 Immediate signal (used by the *Shift Left Halfword Immediate* instruction), and the opcode. The second ALU is used to compute the R4-format Instructions. These instructions are all SIMD instructions that receive all 3 register data outputs from the pipeline register. The only other signal is the opcode. The third module is the Load Immediate Shifter that computes the one R-1 format instruction (Load Immediate). The module receives the data of the register that will be written (used for preservation) and the 16-bit immediate with its offset. The output of all these modules are sent to a multiplexer that selects, using the *result select* signal from the Control Unit, which result will be written back to the Register File.

**Test Conditions (The code used here is in the Appendix, starting on page: )**

**Instruction Buffer**

**Register File**

**Program Counter**

**Pipeline Registers**

**ALU 1**

**ALU 2**

The ALU 2 is used to computer R4 format instructions. These instructions are Single-Instruction Multiple-Data instructions. They are also signed instructions with saturation. The instructions were first tested separately to ensure proper functionality:

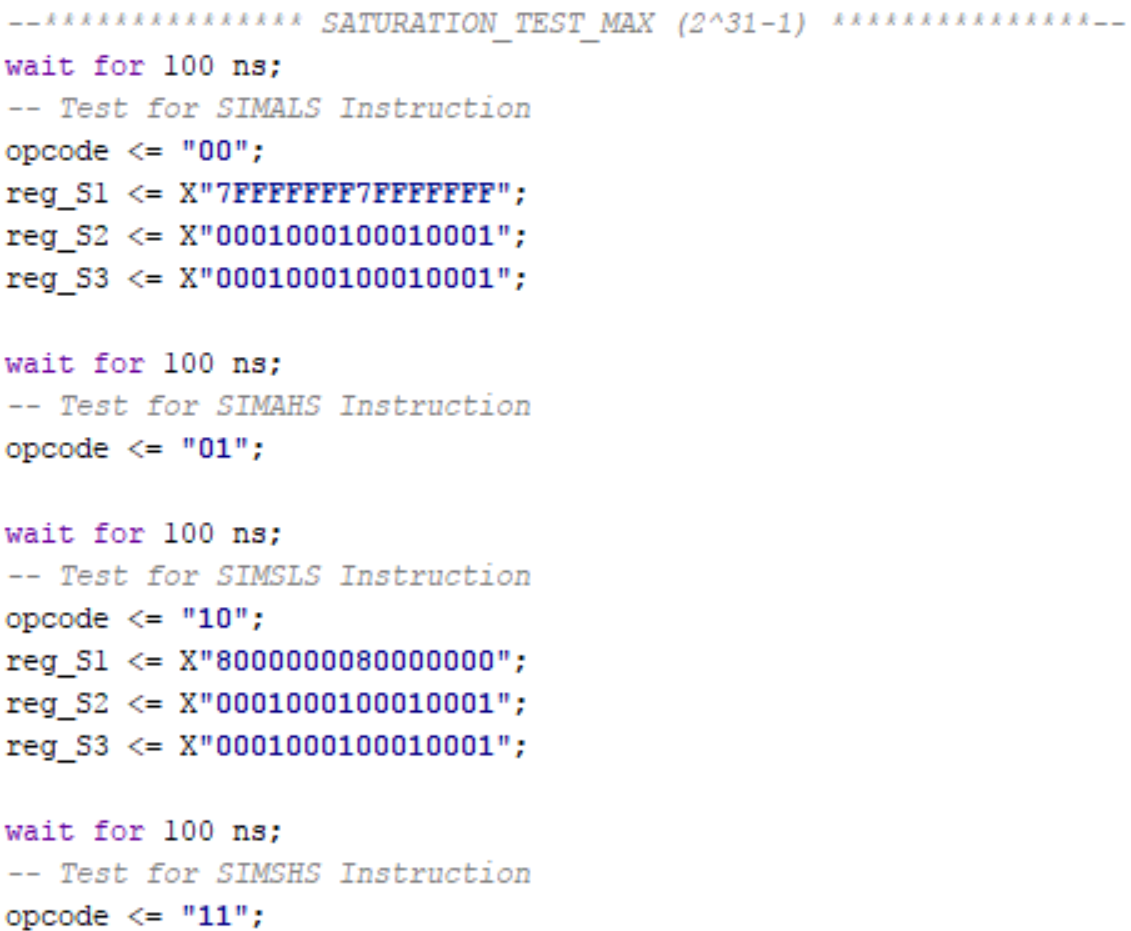


For example, the first test, tests the SIMALS instruction. This instruction multiplies the low 16-bits of rs2 and rs3 and adds the 32-bit product to the low 32-bits of rs1. In the test case, the low 16-bits multiplied are BEEF\*5CDF and the signed product is E865 3631. This product is added with DEAD BEEF and the result being placed into the low 32-bits of rs1 is: C712 F520.

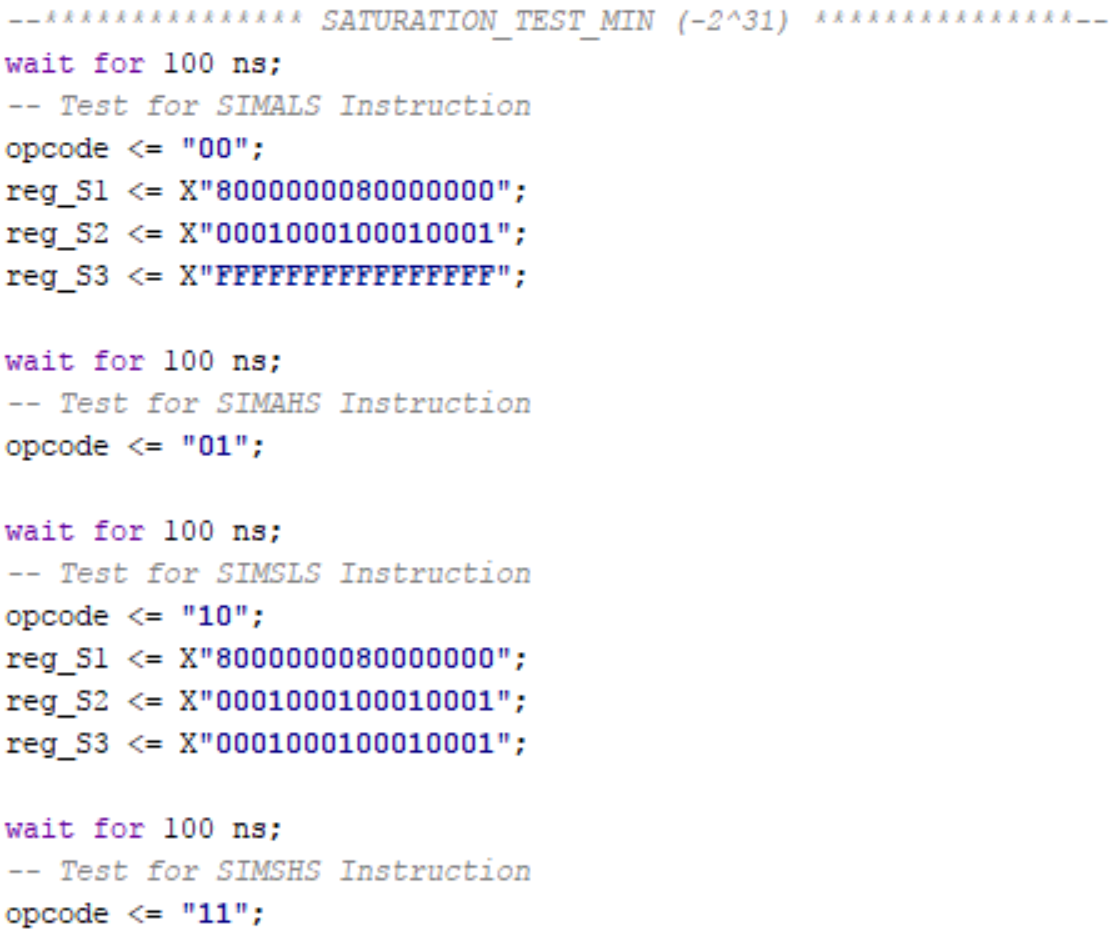
This is proved by the simulation result:



Next, the saturation rounding is tested by inputting instructions that will cause an overflow and an underflow. To test for overflow for the multiply-add instructions, all four 16-bit fields of rs2 and rs3 were set to 1 and rs1 was set to the maximum value. The 32-bit product would be 1 and adding 1 to the maximum value would cause and overflow. To test overflow for the multiply-subtract instructions, all four 16-bit fields of rs2 and rs3 were set to 1 and rs1 was set to the minimum value: 0x80000000. These values subtracted would cause an overflow.

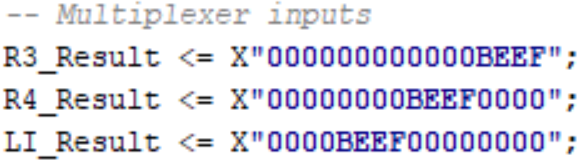


For the underflow checking, a similar test is done:

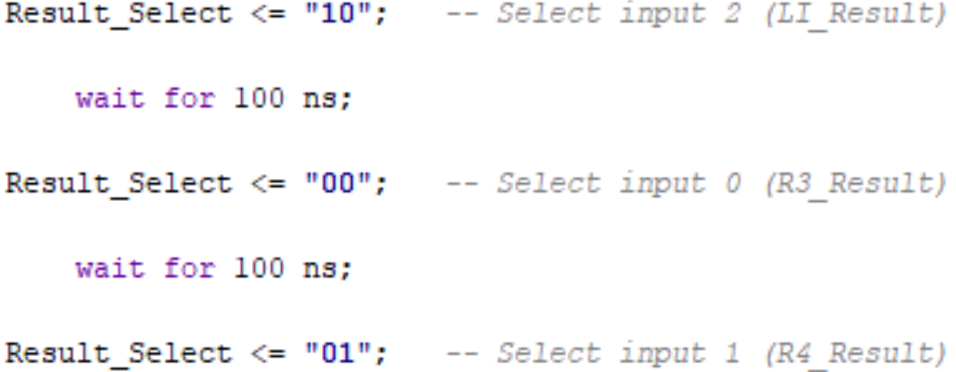


**Multiplexer**

The multiplexer required a very simple test. Three separate results were assigned to the inputs of the module:

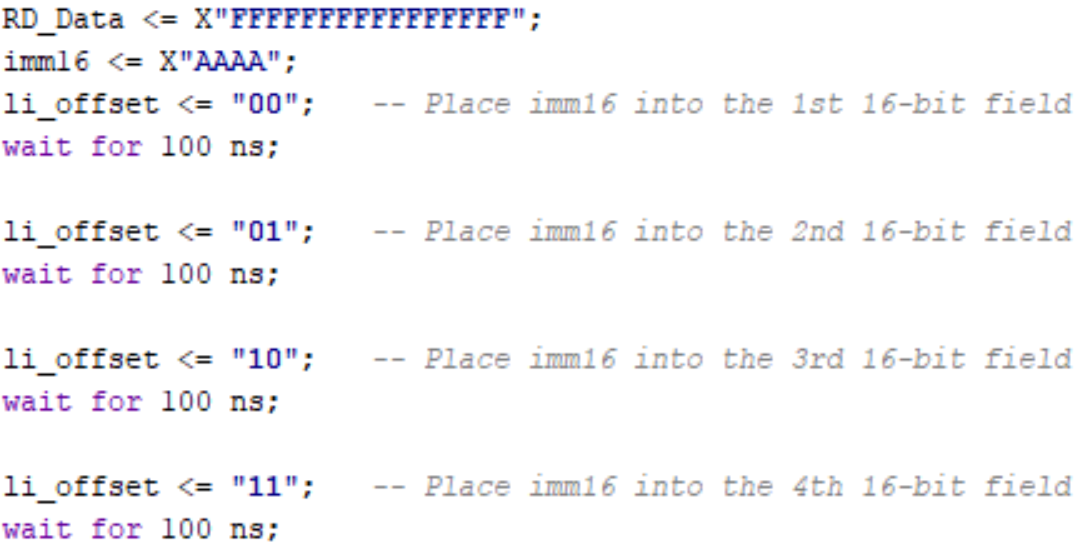


The results were then serially selected using the *Result Select* port:



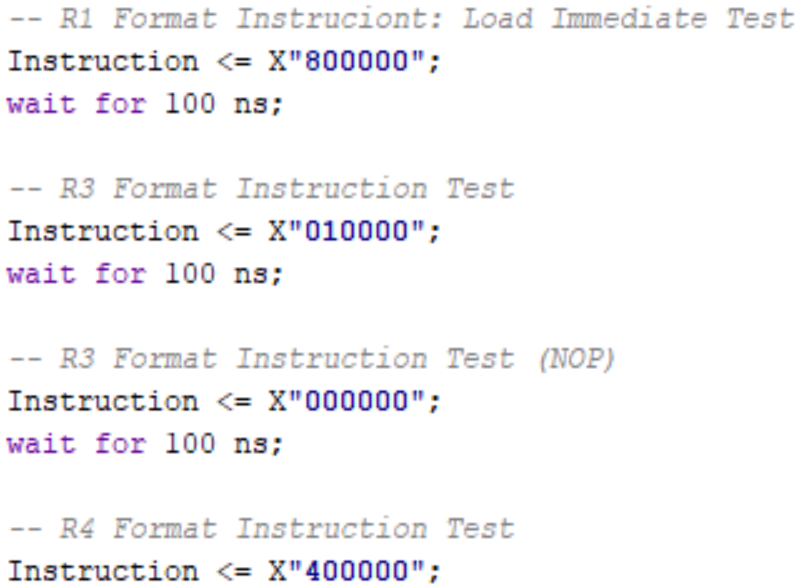
**Load Immediate Shifter**

The Load Immediate Shifter needs to preserve portion of the instruction it is not writing to. This was tested by writing an arbitrary instruction to the *RD\_Data* input of the module and an arbitrary 16-bit immediate to the *imm16* input. The 16-bit immediate was sequentially offset to the four different possible locations of the 64-bit result signal.



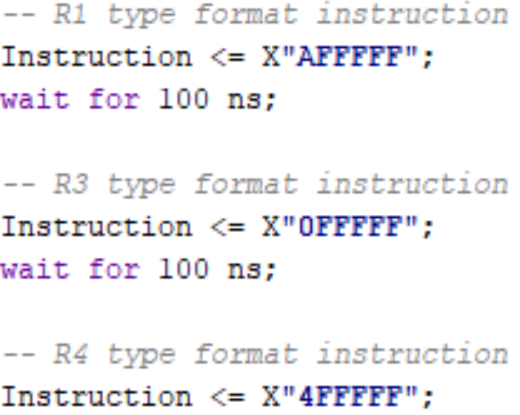
**Control Unit**

The Control Unit was tested by inputting the different types of instructions available and assuring the output control signals were correct. A Load Immediate *(R1)*, two *R3* format instructions (one NOP and a select other). Lastly, an *R4* format instruction.



**Instruction Decoder**

The decoder was tested using arbitrary instructions and ensuring that the generated subset signals were correct.



**Multimedia Unit**

**Conclusions and Discussion**

***Appendix***

**VHDL Code**

**Instruction Buffer**

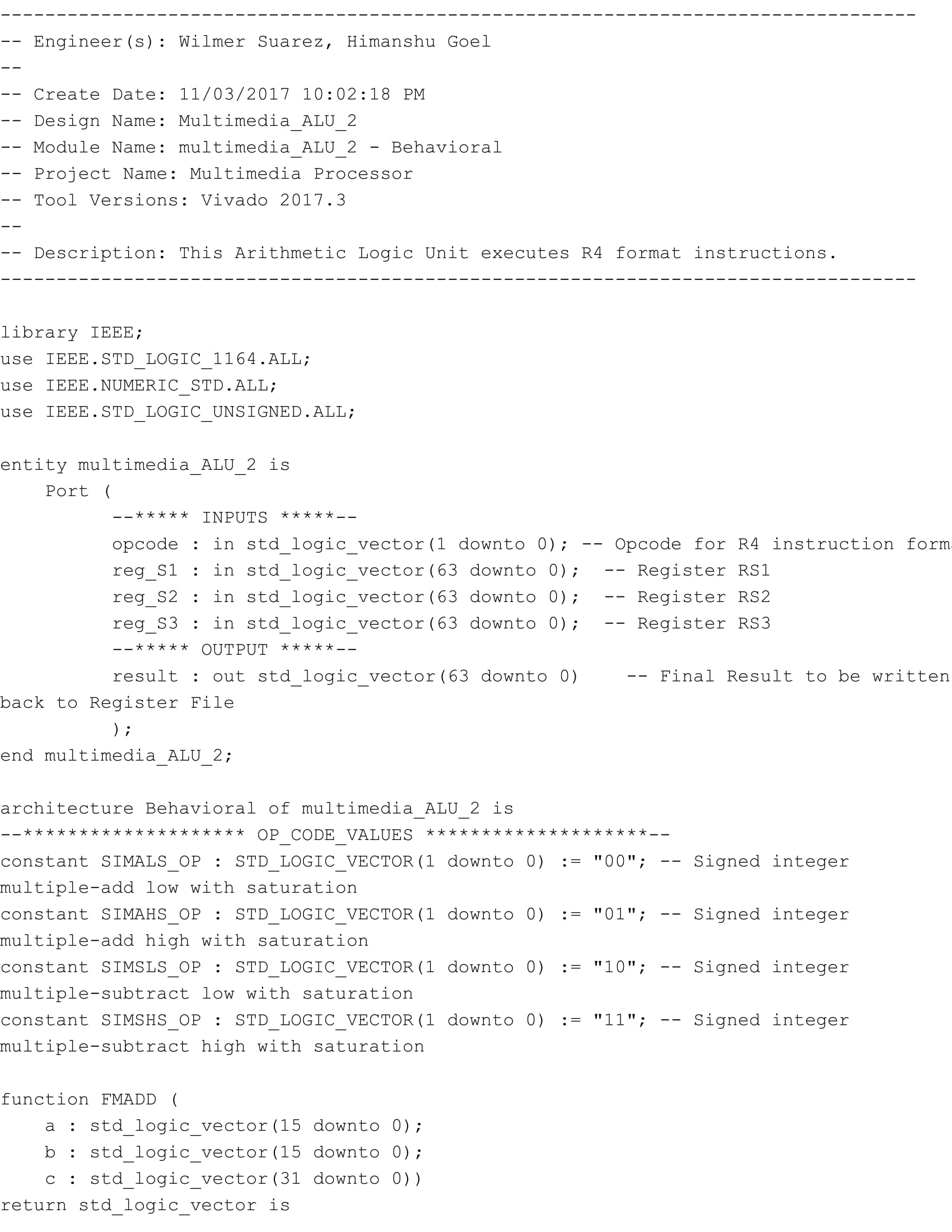
**Register File**

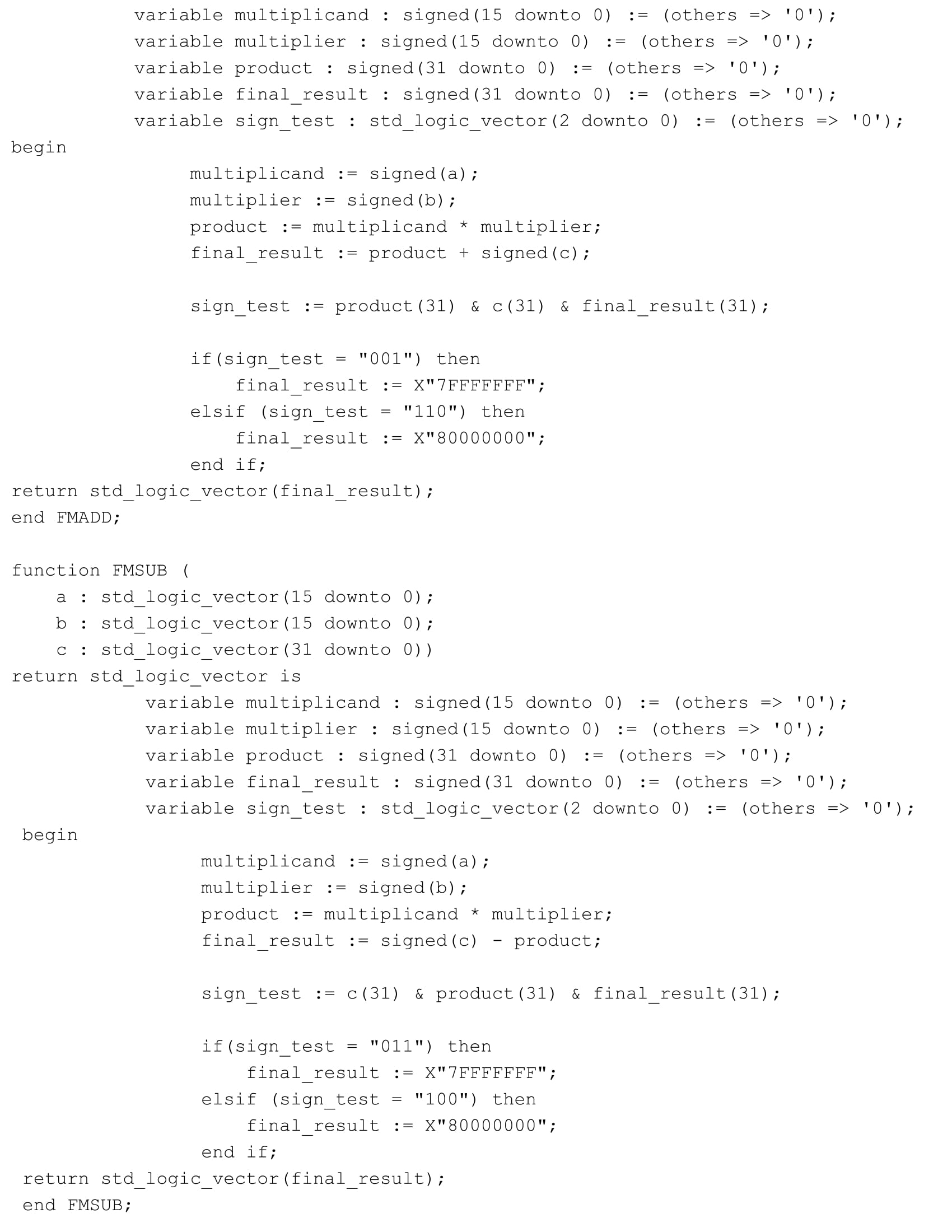
**Program Counter**

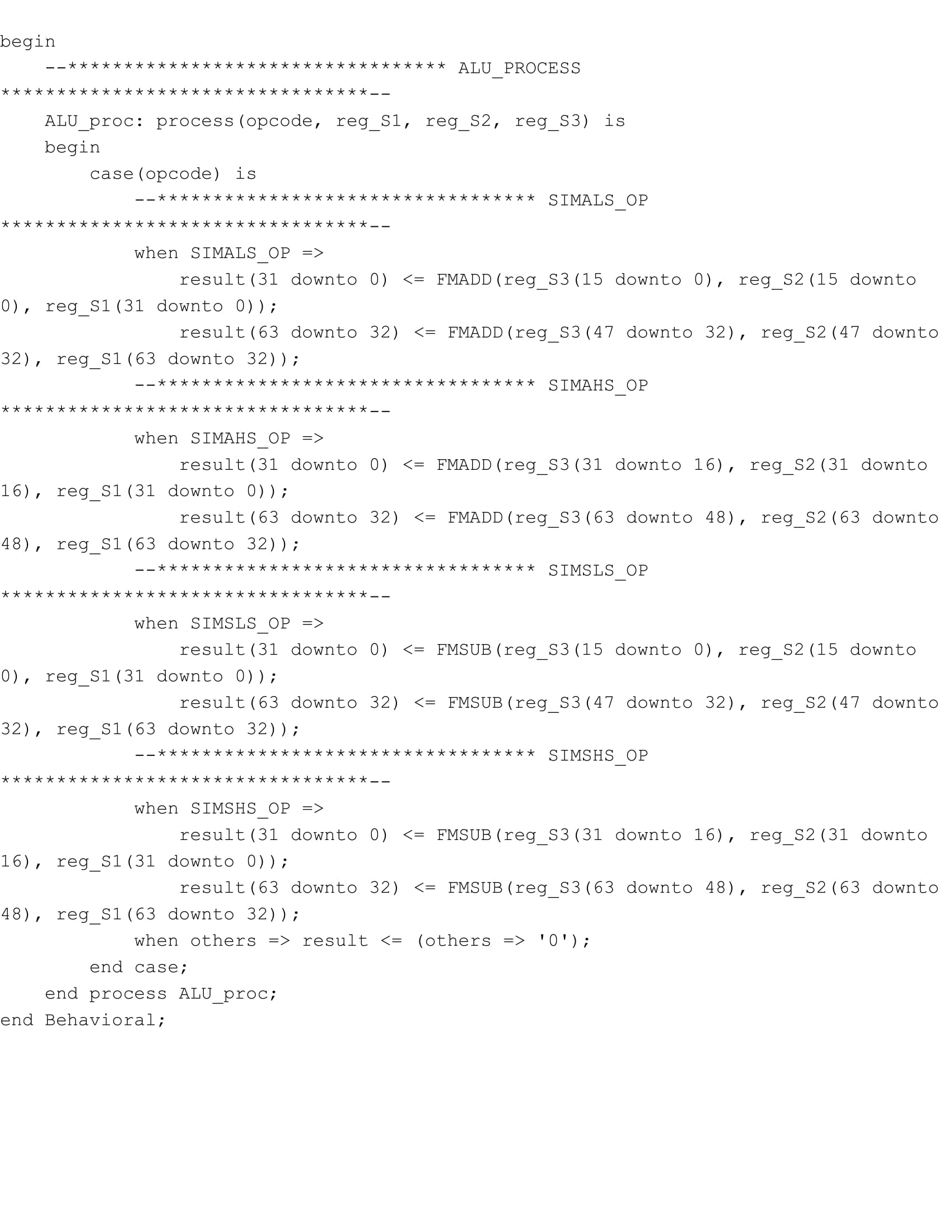
**Pipeline Registers**

**ALU 1**

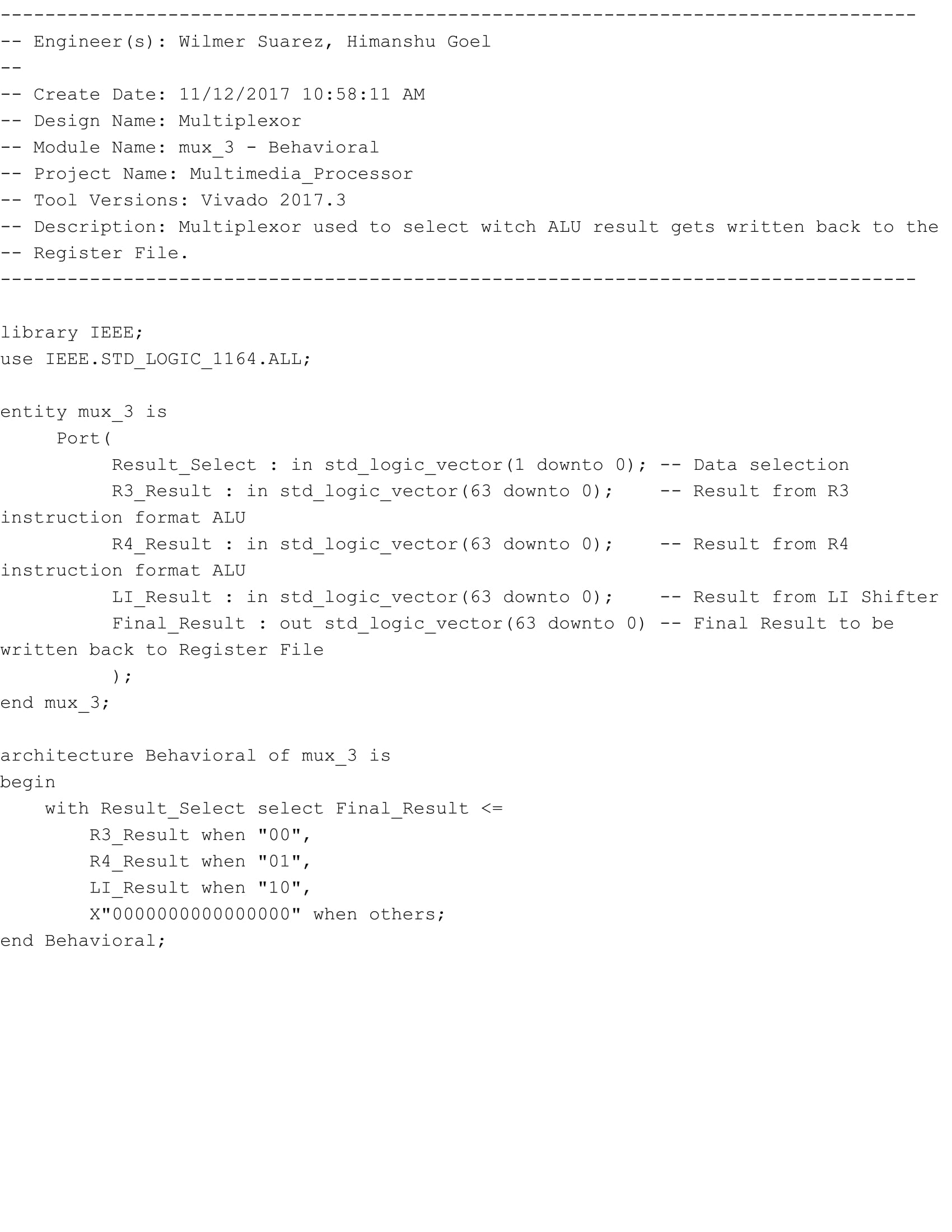
**ALU 2**

****

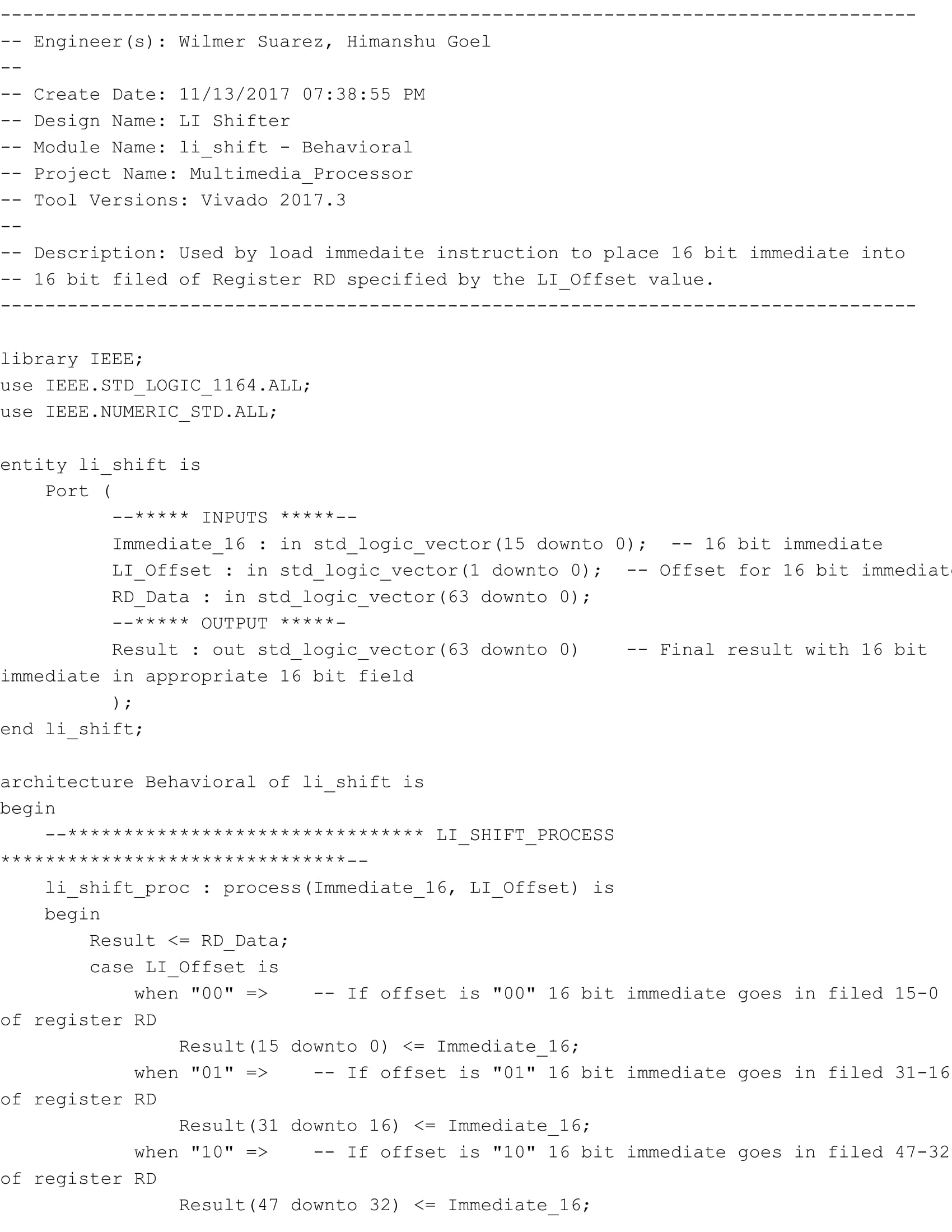
****

****

**Multiplexer**

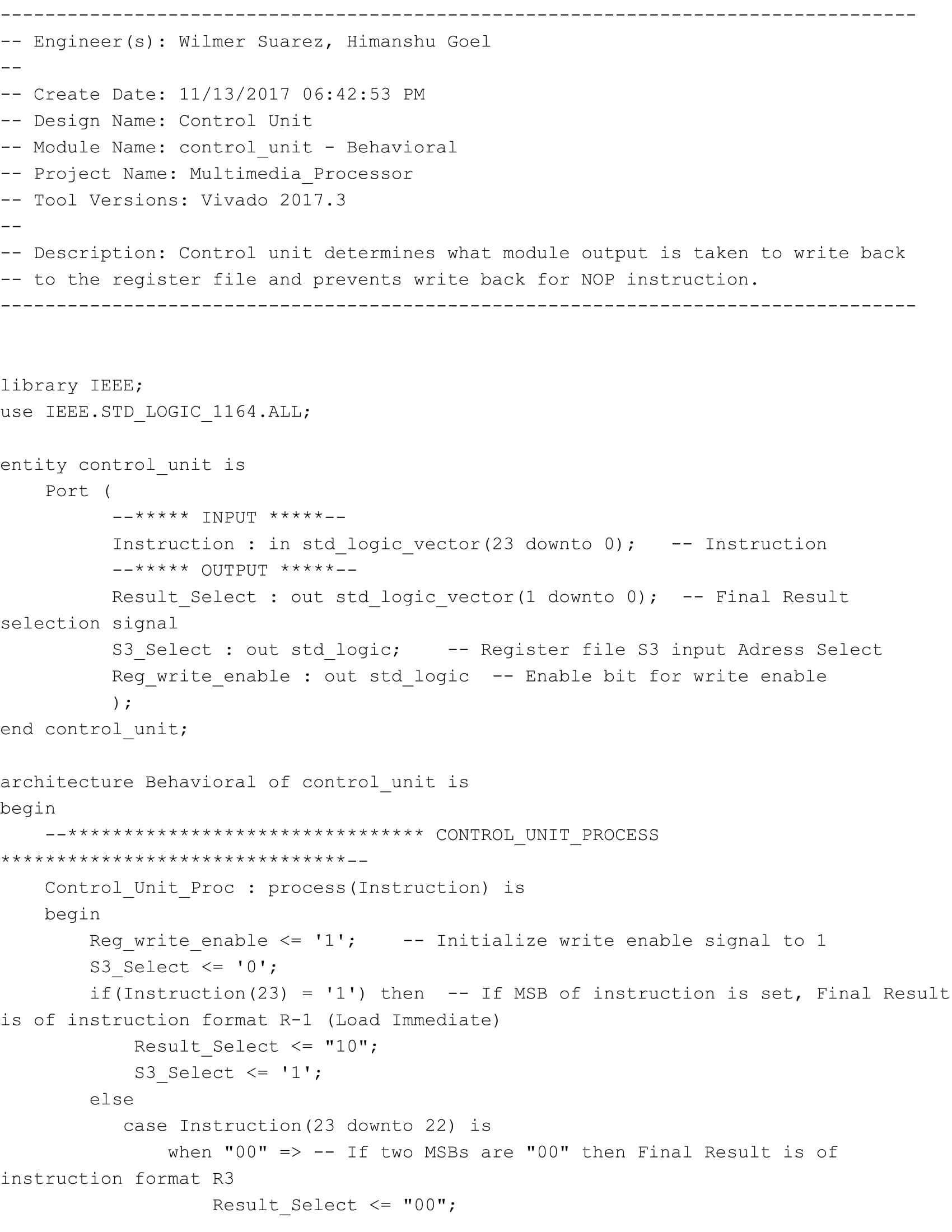


**Load Immediate Shifter**



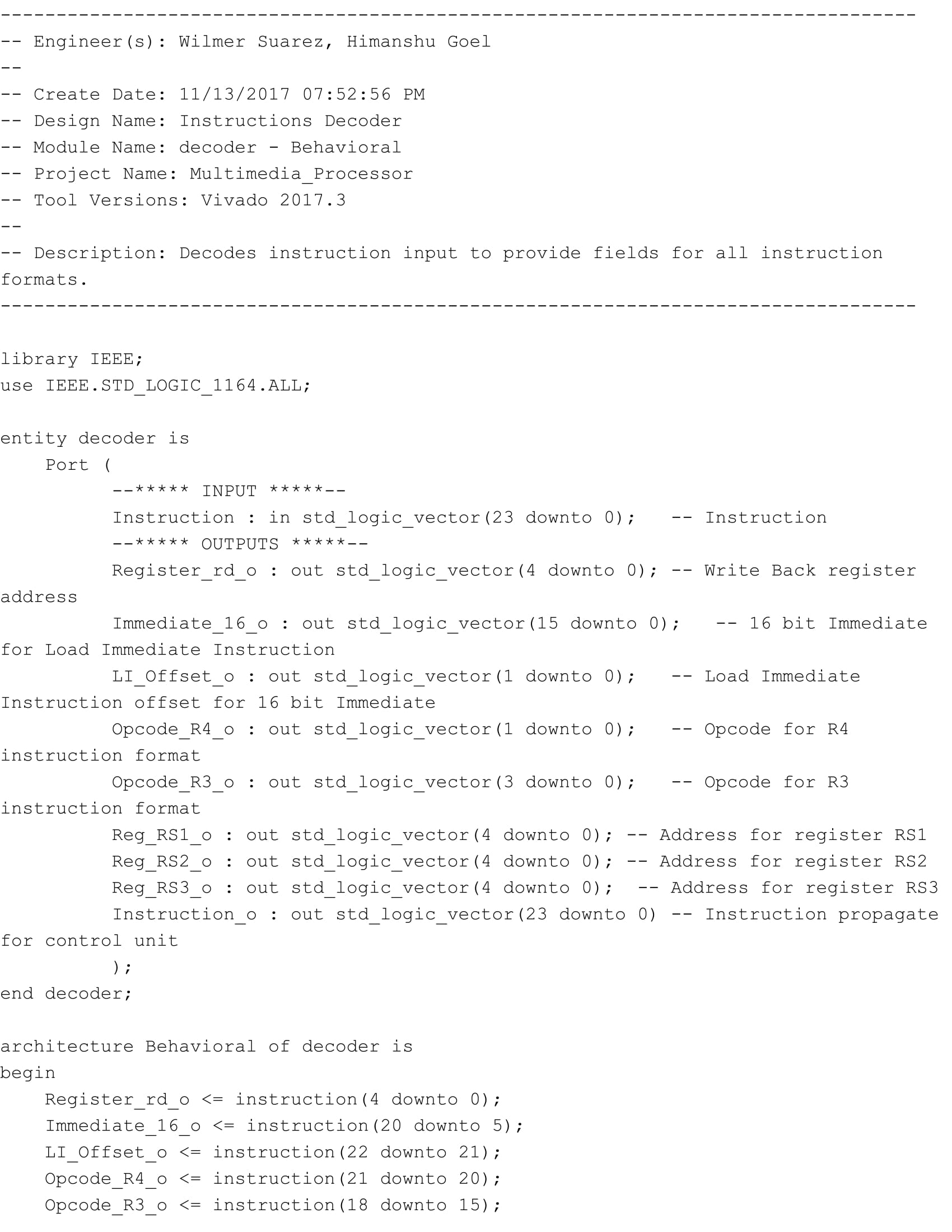


**Control Unit**

****

****

**Instruction Decoder**

****

****

**Multimedia Unit**

**VHDL Testbenches**

**Instruction Buffer**

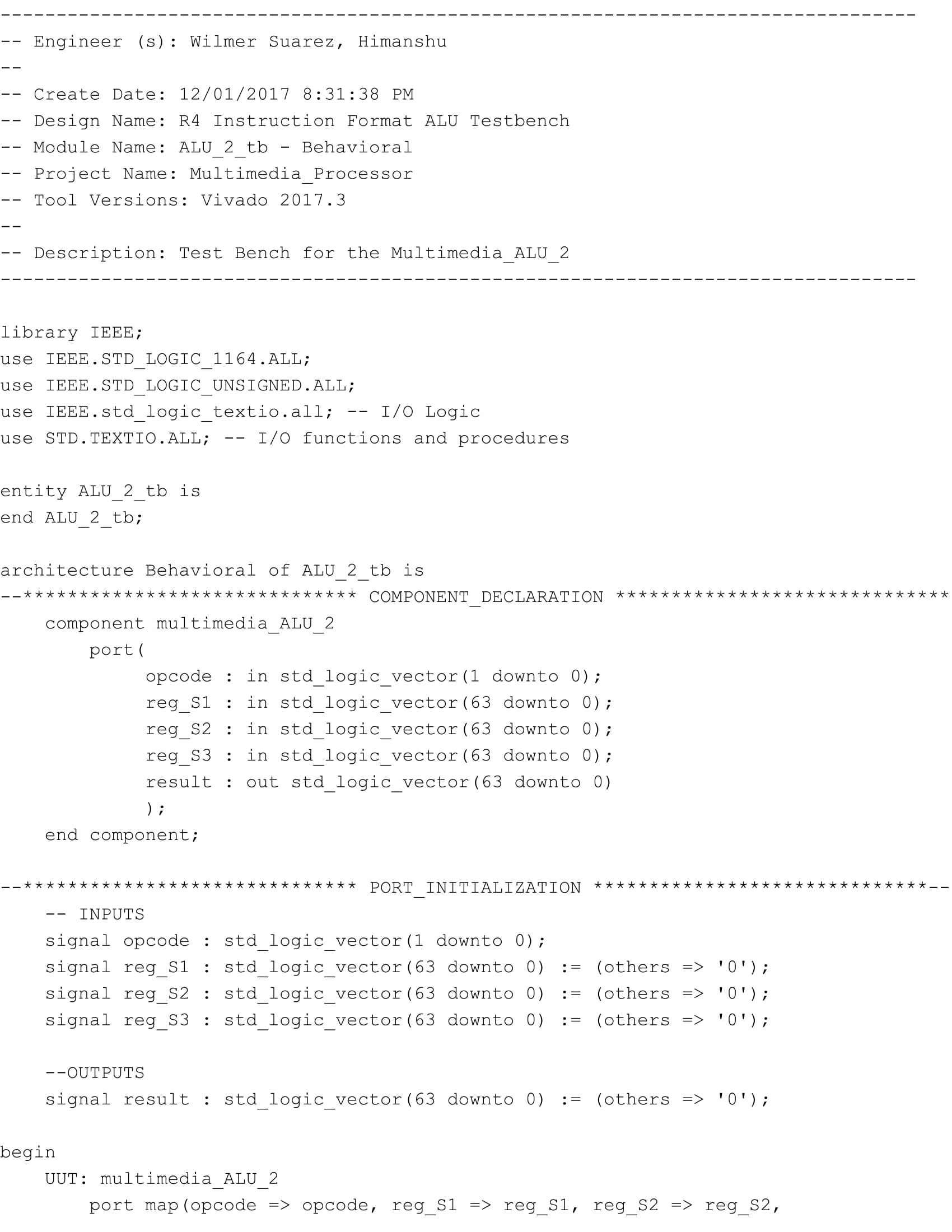
**Register File**

**Program Counter**

**Pipeline Registers**

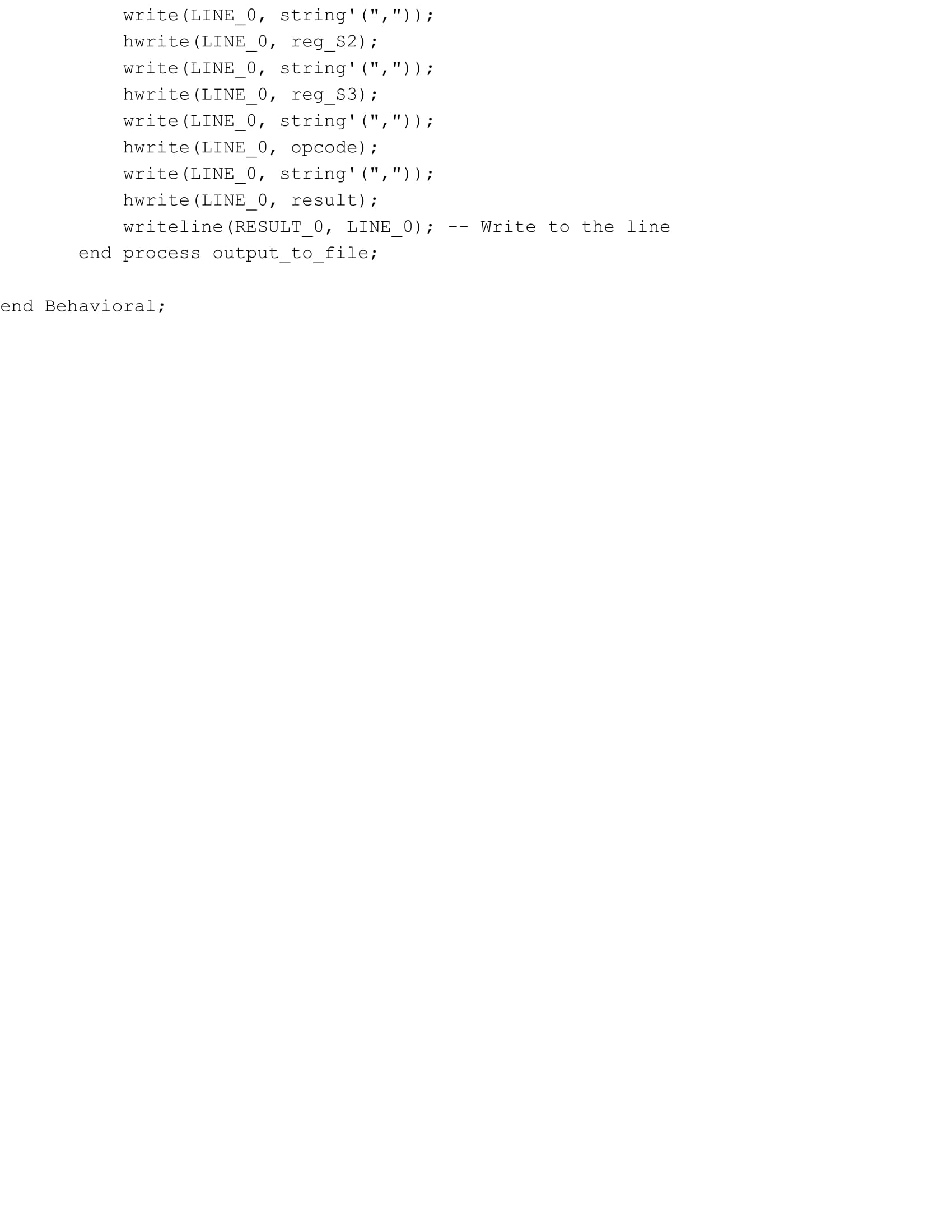
**ALU 1**

**ALU 2**

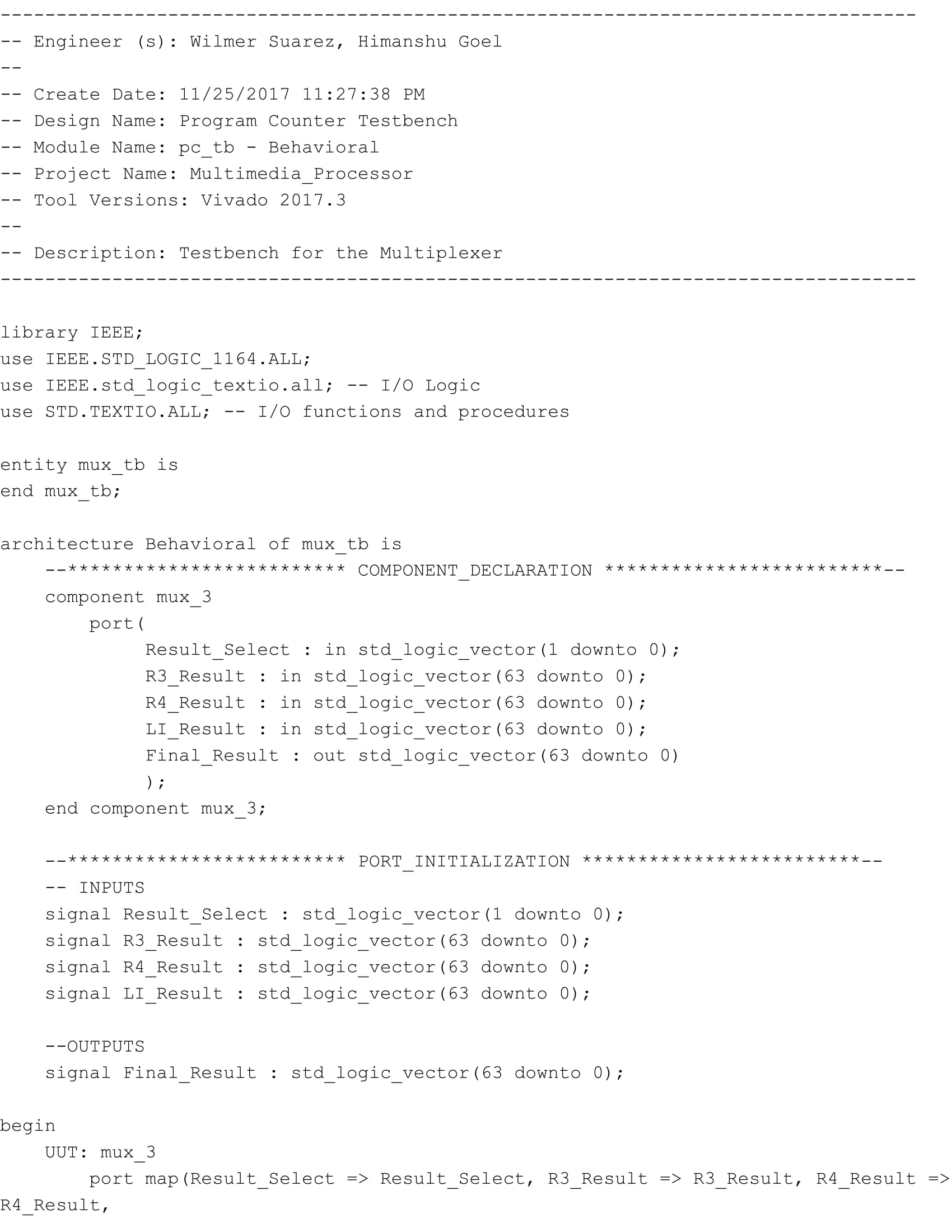
****

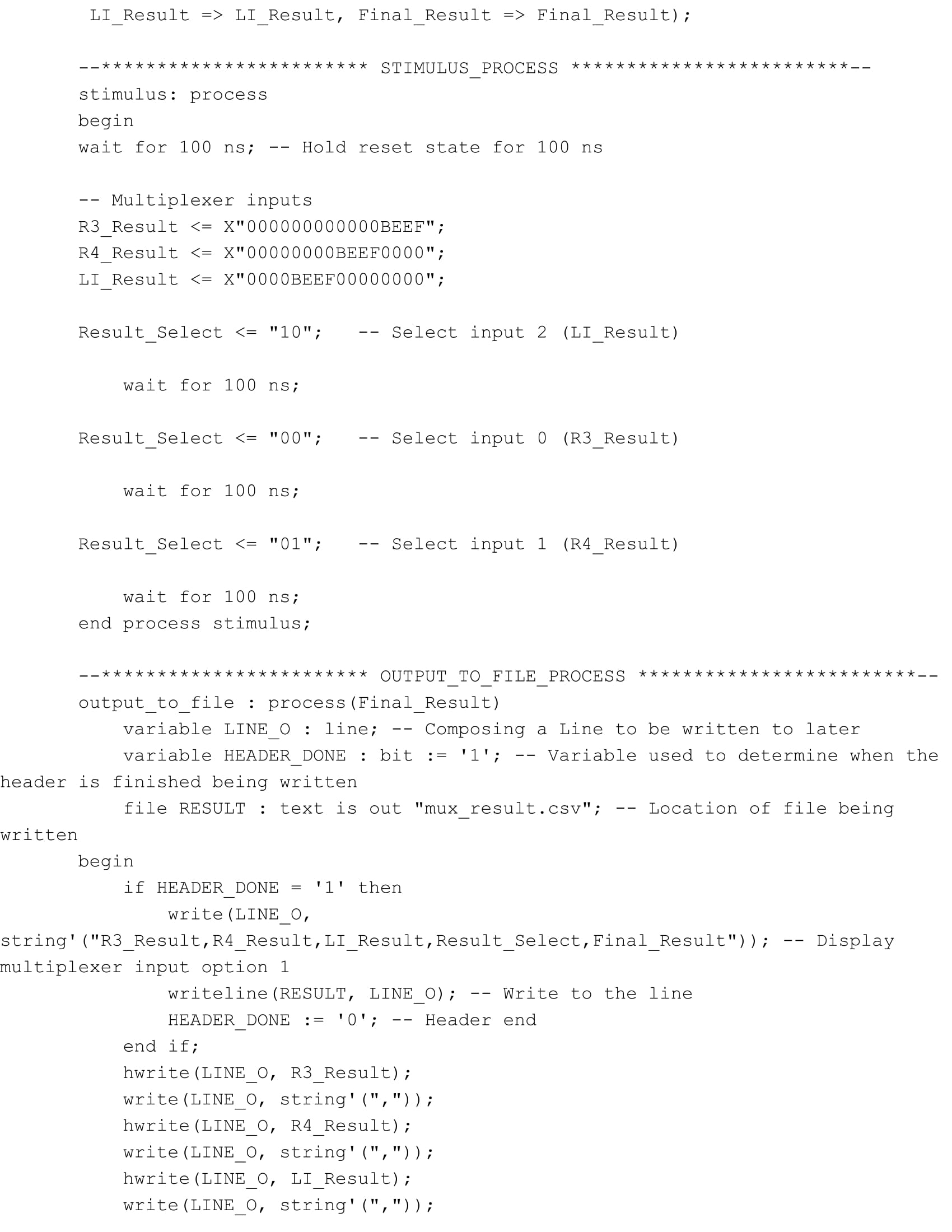
****

****

****

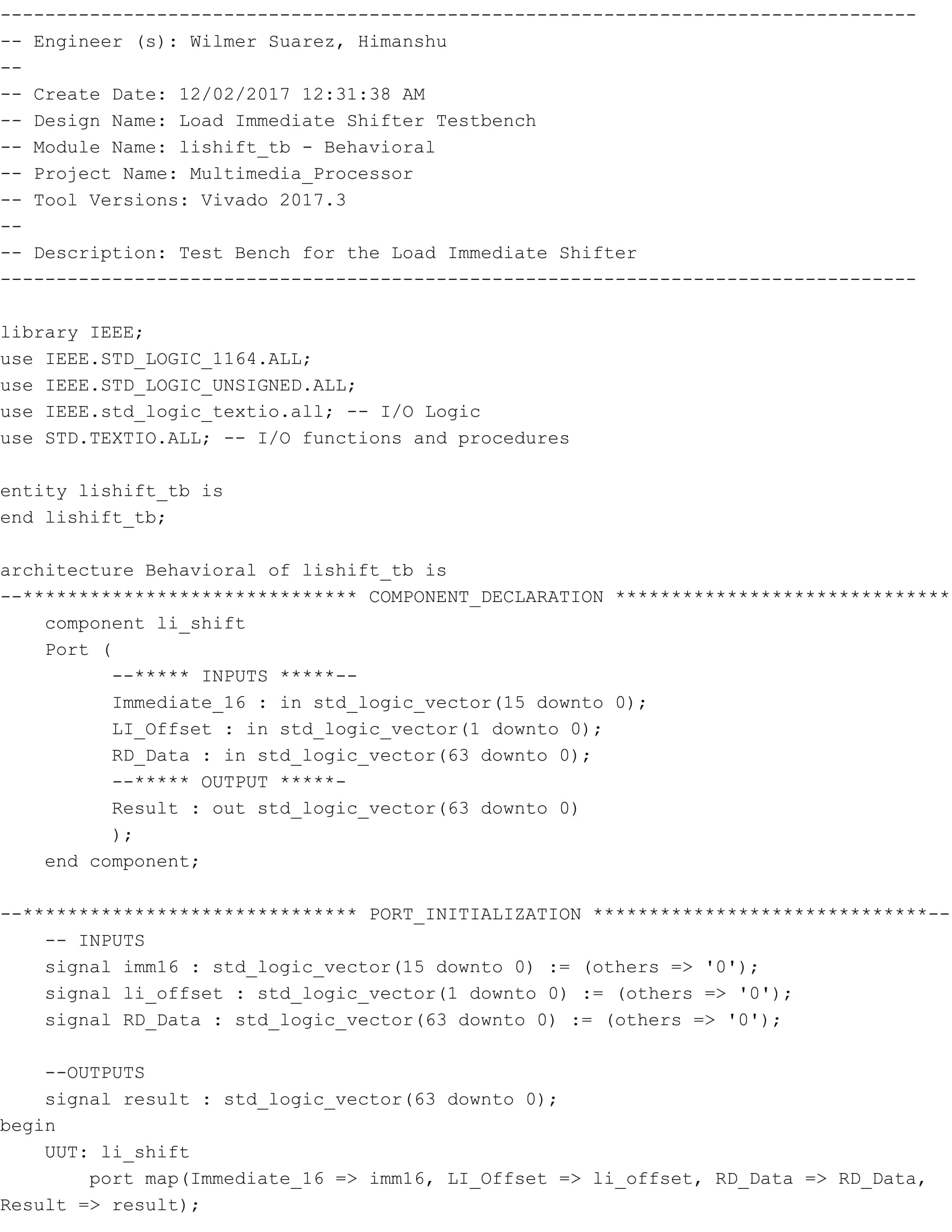
**Multiplexer**

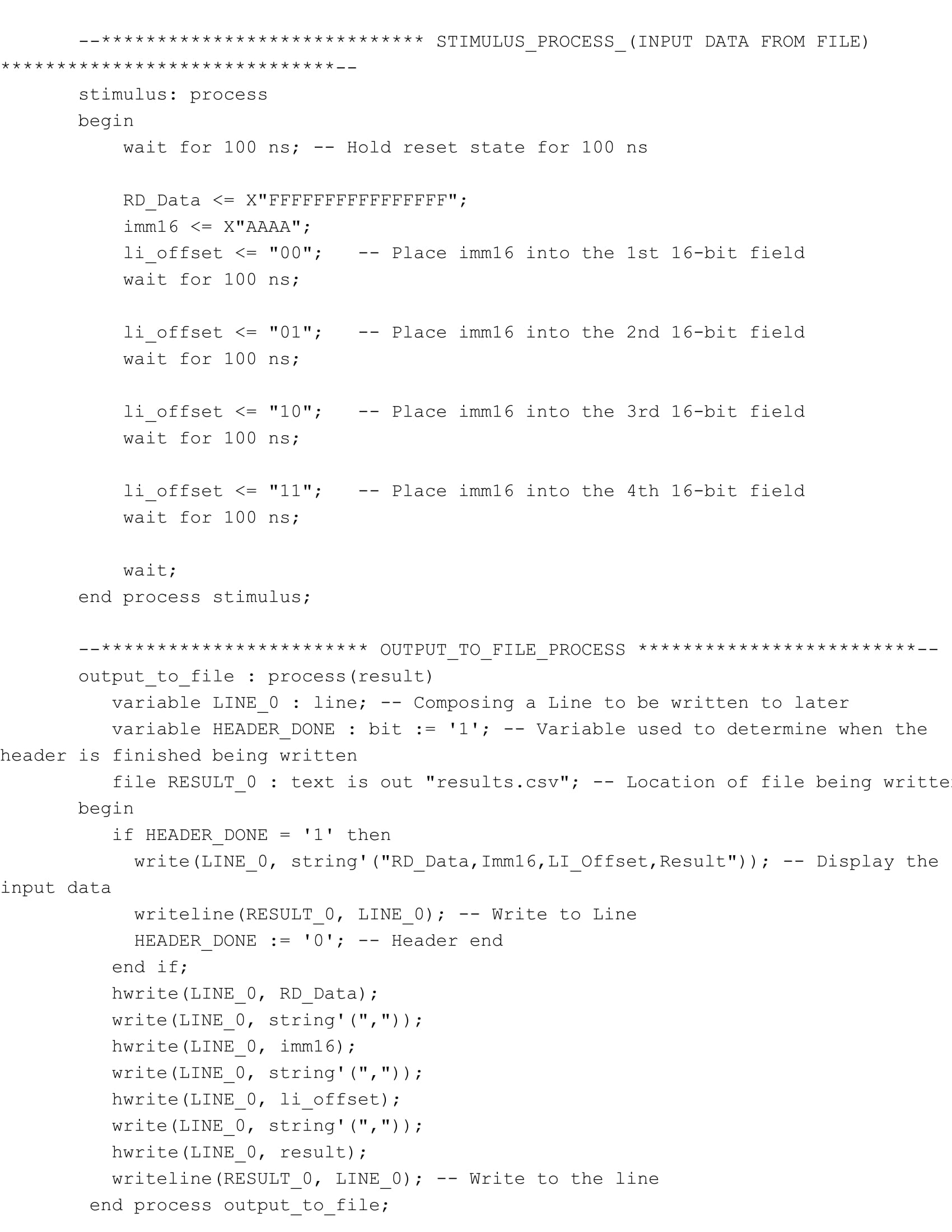
****

****

****

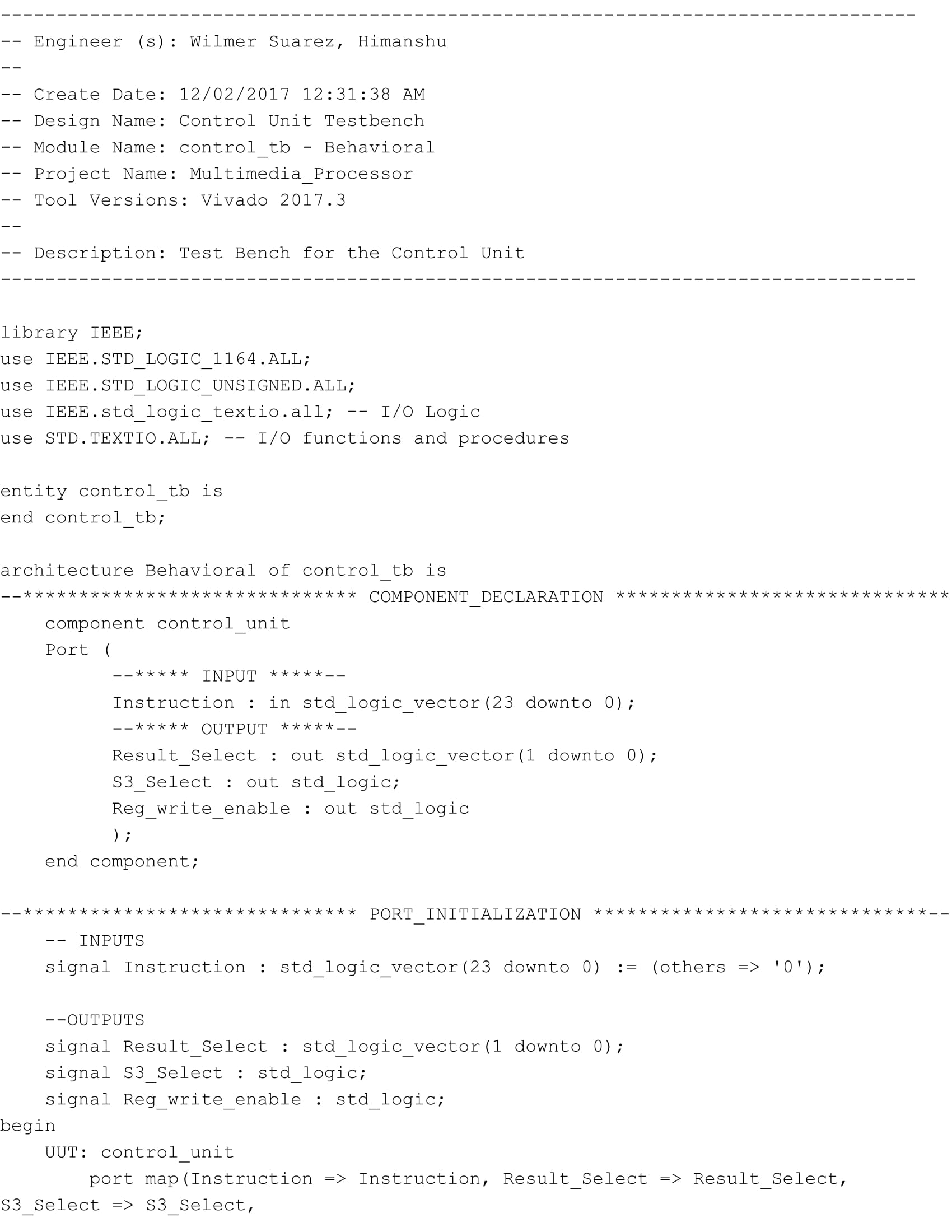
**Load Immediate Shifter**

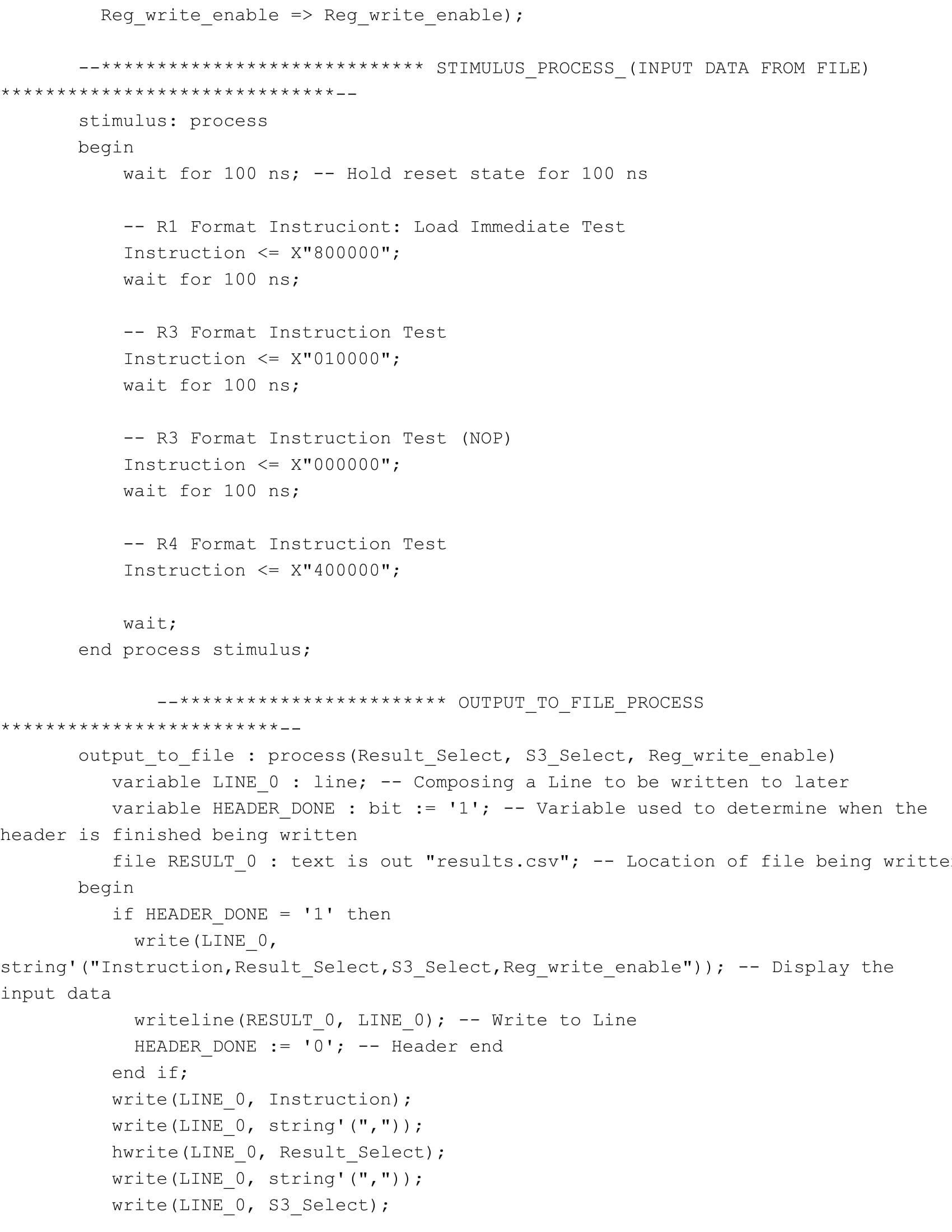
****

****

****

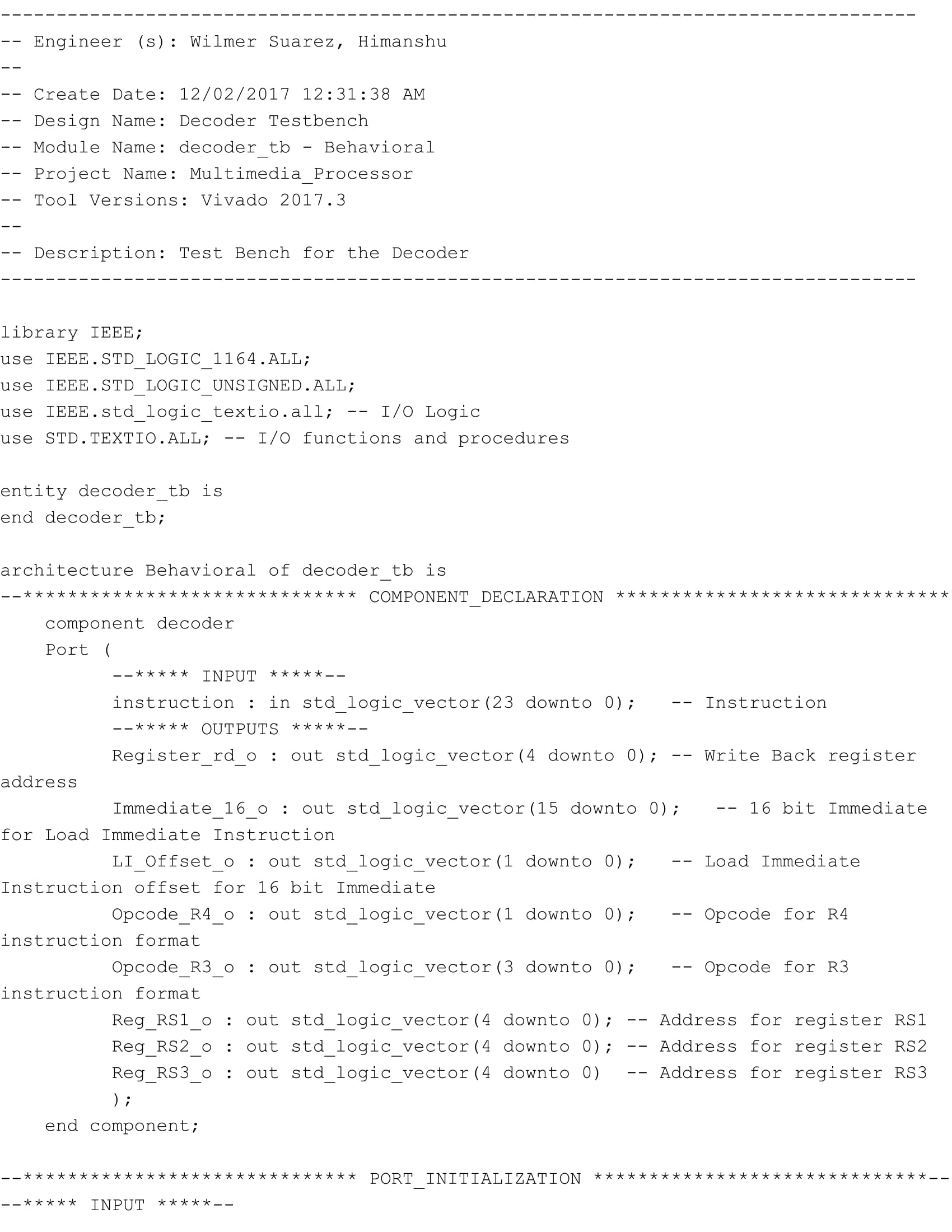
**Control Unit**

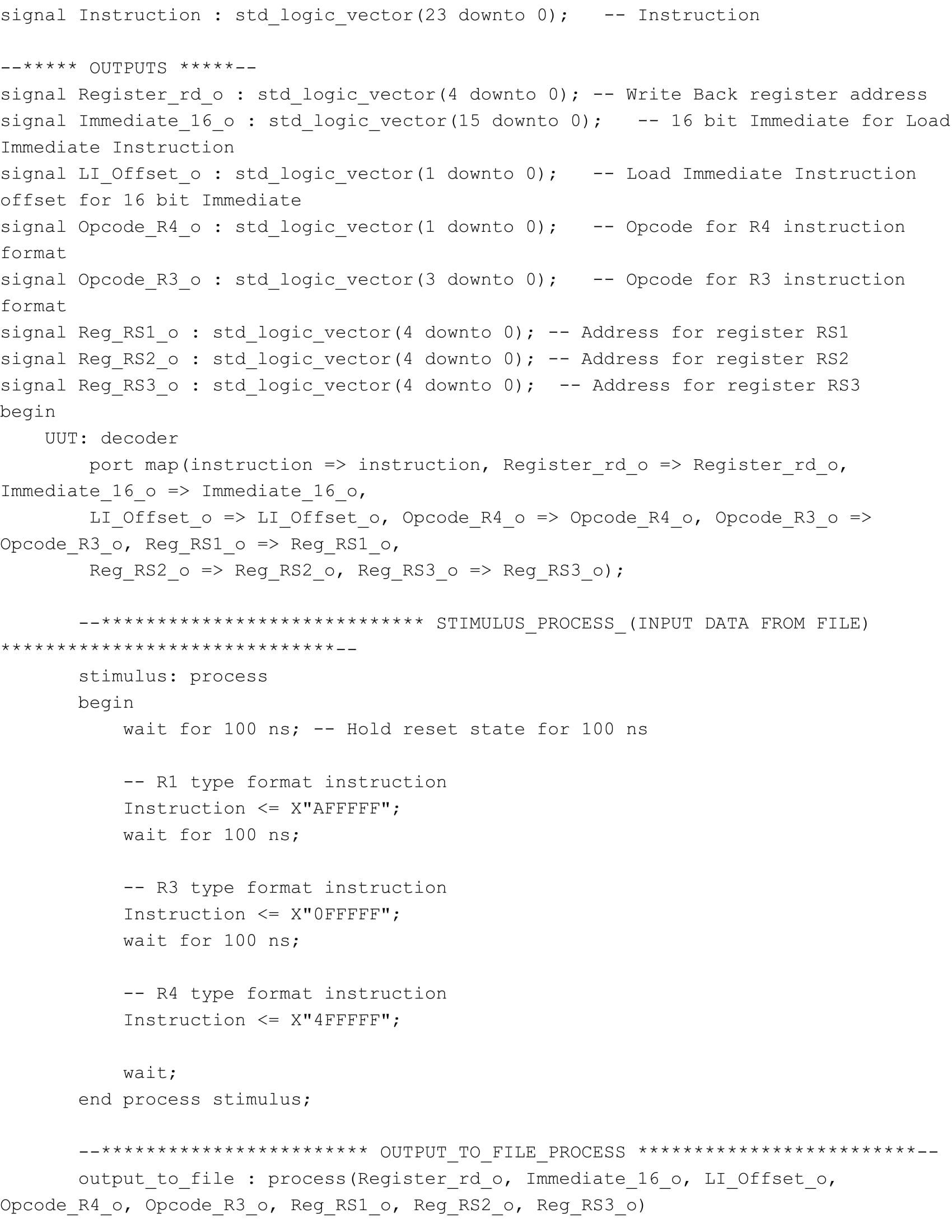
****

****

****

**Instruction Decoder**

****

****

****

**Multimedia Unit**

**Simulation Results**

**Instruction Buffer**

**Register File**

**Program Counter**

**Pipeline Registers**

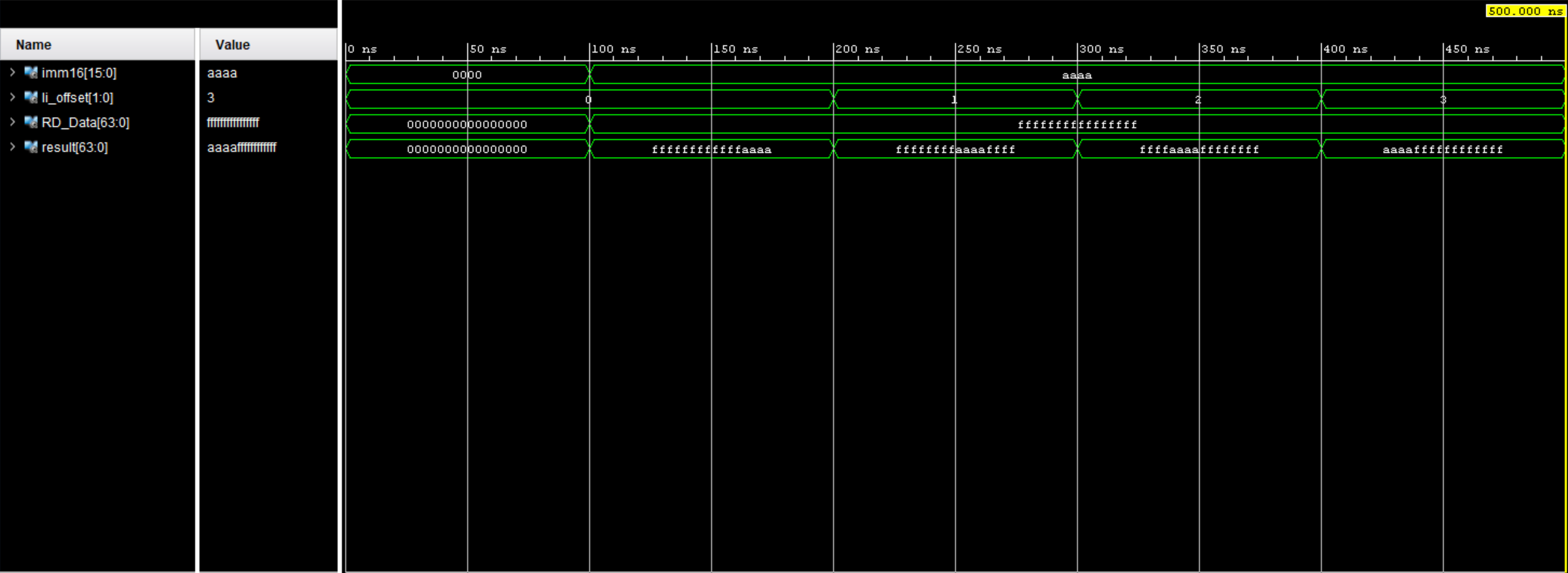
**ALU 1**

**ALU 2**

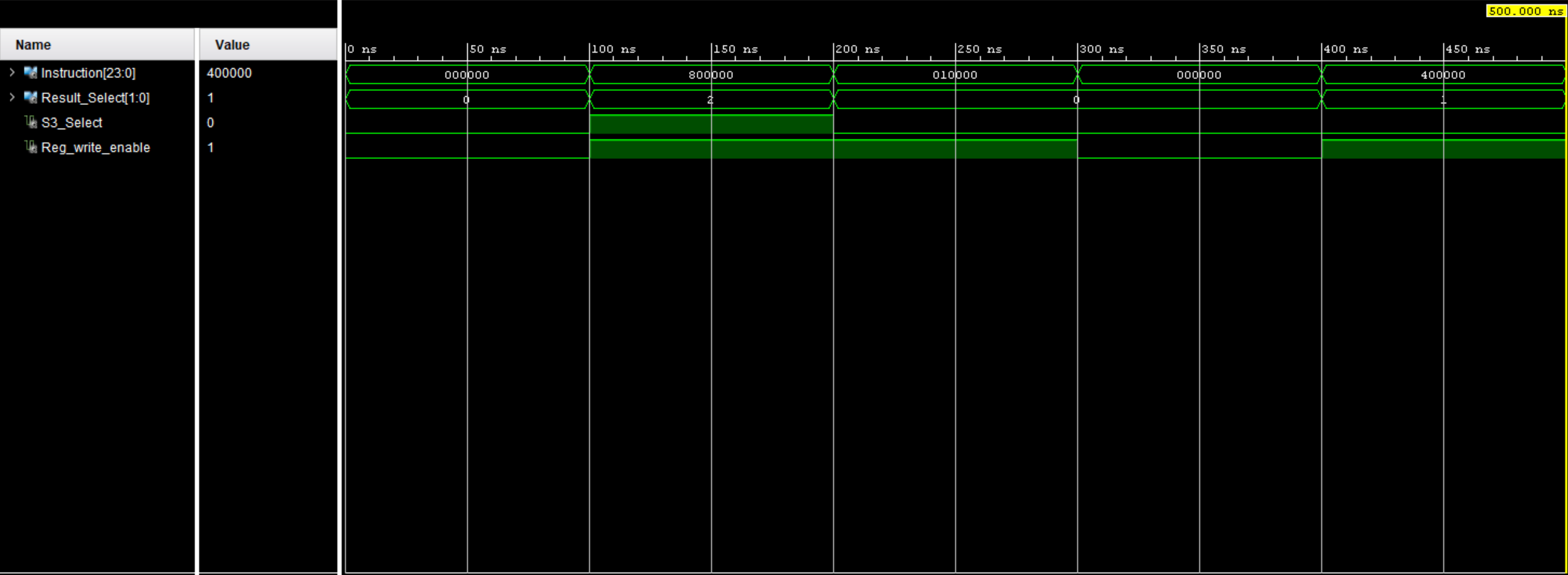
**Multiplexer**



**Load Immediate Shifter**



**Control Unit**



**Instruction Decoder**



**Multimedia Unit**

**RTL Design/Schematic of the Multimedia Processor Unit**

**Multimedia Unit Test Programs**

**2x2 Matrix Multiply**