[[1]](#footnote-1)

DaVinci 1.0: Simple computer system supporting CS147DV Instruction Set

Michelle Song   
Department of Computer Science  
San Jose State University  
E-mail: michelle.song@sjsu.edu

*Abstract*—dfdsfd

# introduction

# dfsdfsdfsd

# requirements

The following section states the software needed and minimal instructions for program execution and …

## Software Requirements

The digital simulation tool necessary for running the program is ModelSim. The installation process for the student edition will be specified, however, there are options for those who are not in academia. Additionally, the usage of ModelSim such as the creation and execution of the project will be briefly covered.

### Installation of ModelSim (Student Edition)

To install ModelSim, visit the link to the student edition: <http://www.mentor.com/company/higher_ed/modelsim-student-edition>. Click on the “Download Student Edition” button.

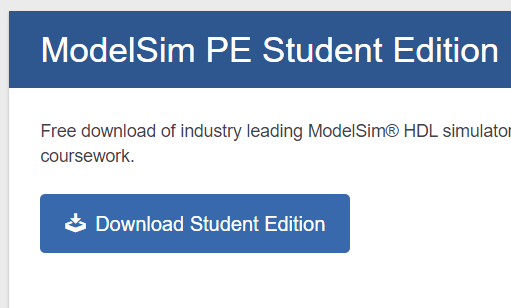


Fig. 1. Downloading ModelSim – Download button

After installation, fill out the form to obtain the student license, ensuring that the email is correct. Next, check the email received from ModelSim and download the license attached to the email. There are additional instructions on the email for where to save the license file. As stated in the email, it is mandatory to keep the license file untouched for the license to work properly.

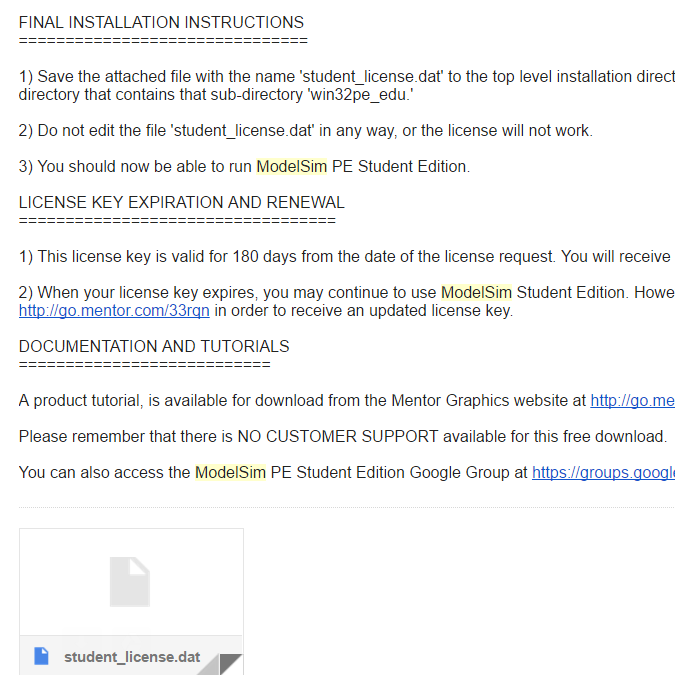


Fig. 2. Downloading ModelSim – License email

### Creating a simulation project

After successful installation of ModelSim, open the workbench such that the menu and archives are displayed. After confirming that the project files (.v files) are downloaded, go to File -> New -> Project. Enter a name for the project and navigate to the directory where the project files were downloaded. Afterwards, hit “OK”.

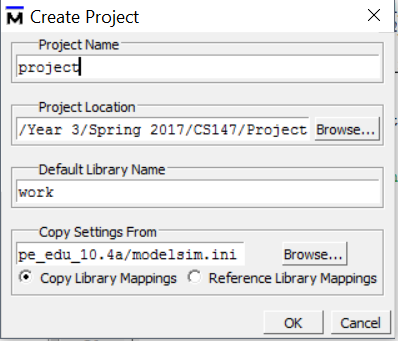


Fig. 3. Creating a project in ModelSim

### Creating a simulation

Once the project has been created, select all files on the workbench and right click and select Compile->Compile All. Next, select Add to Project -> Simulation Configuration. On the design tab, expand the options for “work” and select the modules as shown in Figure 4. Once done, hit save.

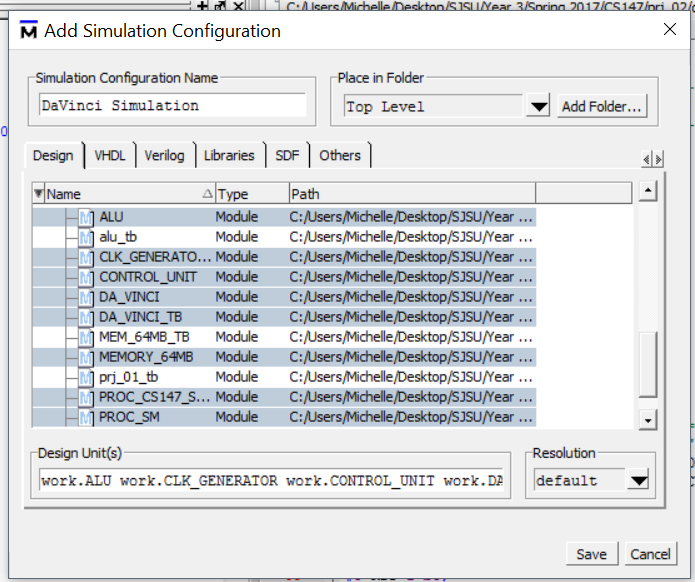


Fig. 4. Creating a simulation – Configuration window

### Running the simulation

To run the simulation, right click on the name of the simulation created and click on execute. Then, on the toolbar at the top, go to Simulate -> Run - > Run -All. The results shown on the transcript will display … (Select option in test bench) The memory data is dumped into the file “fibonacci\_mem\_dump” created in the current directory. For more information on test cases, see section IV – Testing of this report.

### Observing waveforms

To observe waveforms, go to the sim tab and double click “prj\_01\_tb”. In the objects window that appears afterwards, select all objects and click on “Add wave”.

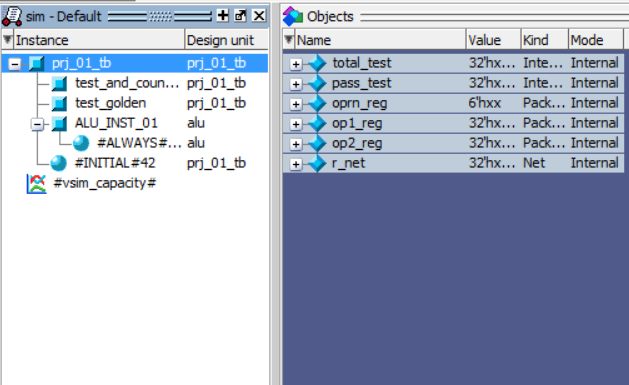


Fig. 5. Observing waveforms – Adding a wave

Next, run the simulation by clicking on “Simulate” on the top toolbar and selecting Run->Run –All. This enables navigation of the change in values occurring at specific time intervals in picoseconds.

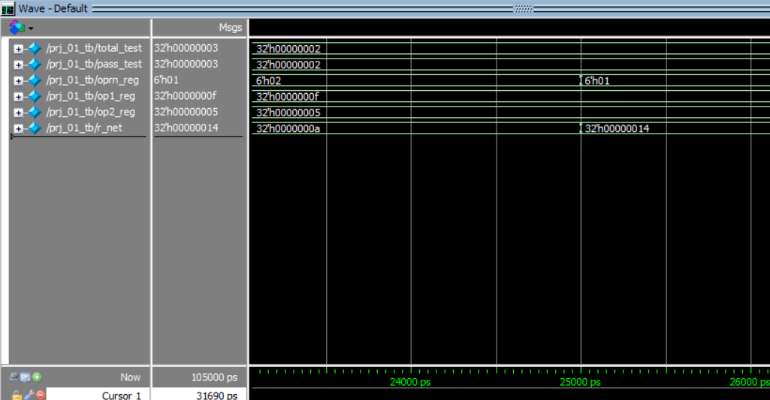


Fig. 6. Observing waveforms – Viewing the wave

## Requirement for DaVinci v1.0

Describe DaVinci here.

### ALU

The arithmetic and logic unit (ALU) is responsible for the mathematical and logical operations happening in a computer, providing the foundation for the functionality of a computer whose tasks are broken down into many arithmetic operations.

The structure of the ALU comprises of two operand ports, one operation port, and a port for the output of the computation. The number bits for every port depends the operation width of the computer. For example, in Figure 4, the operation width of the computer is 32-bit and thus the number of bits for op1, op2, and the result is 32.

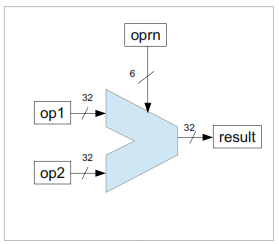


Fig.7. Schematic diagram representation of ALU

The ALU is responsible for handling basic arithmetic operations such as addition, subtraction, multiplication, and division as well as logical operations such as AND, OR, NOT, XOR. The correct operation is selected by the operation code passed from the control unit and applied to the two operands. The functionality of the ALU can be represented in a switch-case statement in the C code shown in Figure 5. Depending on the operation code given, an operation is selected to be used on the two operands.

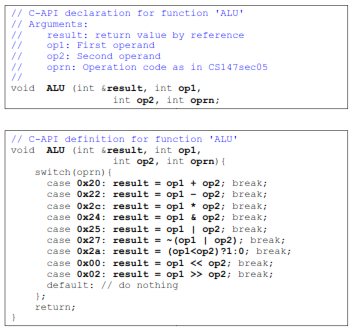


Fig.8. Corresponding C code for ALU

### Memory

### Register File

### Control Unit

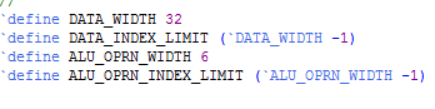
### Processor

### Clock

# implementation

### prj\_definition.v

The project definition file “prj\_definition.v” defines the number of bits for the operands, operation, and output ports. The variables DATA\_WIDTH and ALU\_OPRN\_WIDTH correspond to the operands and operation code inputs, respectively. The ALU implemented utilizes these definitions to follow the requirement of the ALU using the statement “include prj\_definition.v” located at the top of the “alu.v” file.

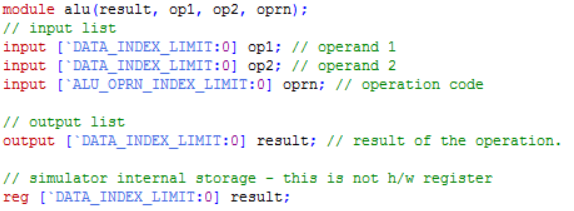


### alu.v

The “alu.v” file creates a module or a design of the ALU providing a way of communication between ports. The module is declared with the keyword “module” followed by the name of the module, “alu”, and the name of the ports passed in as parameters: result, op1, op2, and oprn.

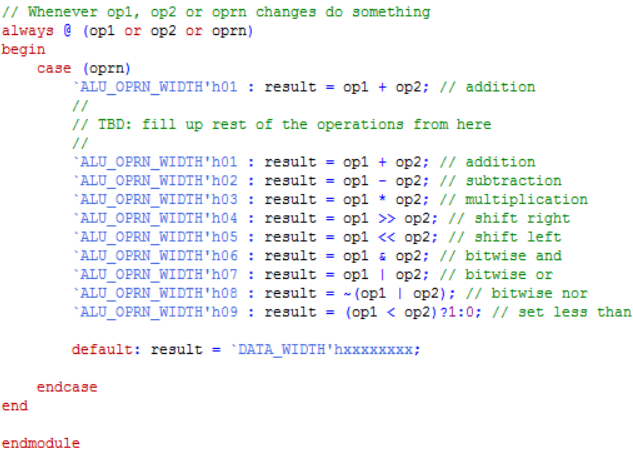
1. *Initializing the ports*

In the lines of code proceeding the declaration, whether the port is input or output is specified along with the port width. In this case, the ports op1, op2, and are wire. Since the default type is wire, only the result port needs to be specified as reg.



1. *Statements for basic and logic operations*

The “always” block ensures that the ALU is always functional and will perform as long as op1, op2, or oprn changes. Inside the always block, there is a case statement similar to the case-switch function as used in a higher level language like C or Java. Depending on the operation, the result is computed in a different way. For a simple example, if given the operands op1 = 5, op2 = 3, and operation code = 1 (addition), the result obtained will be 5 + 3 = 8. The 9 supported operations for the declared ALU are: addition, subtraction, multiplication, shift right, shift left, bitwise and, bitwise or, and set less than. Each operation has a corresponding opcode that will allow a different computation on the operands.



### Memory.v

### Register\_file.v

### Control\_unit.v

### Processor

### Clk\_gen.v

# testing

After installation of ModelSim and ensuring that the project is properly loaded, start the simulation and run all. The results should ..

### Testing procedure

Separate test benches. Test whole thing by picking file and comparing file to golden file.

1. *Timing control*

The first line of code in “prj\_01\_tb.v” specifies the time unit for delays that occur during simulation. The statement “`timescale 1ns/10ps” indicates that the timing delays are multiplied by 1ns. The compiler rounds the resulting delay by the closest integer multiple of 10ps.

1. *Initializing ports and instantiation of system*

# Conclusion

In CS47/147, we learned the concept of the computer system without a chance to implement it. With a hands-on approach as done in this project, the concept of the computer system became more of a reality. I learned how the system could be implemented using a high-level language such as Verilog. During the duration of the project, I learned how to interpret Verilog code through the comments provided in the original project files and the research needed for the project. Overall, this project contributed to my understanding of the hardware design process, brought the concept of the computer into reality, and provided me an opportunity to learn the Verilog programming language.

1. [↑](#footnote-ref-1)