**Module: R5: RV-fpga**

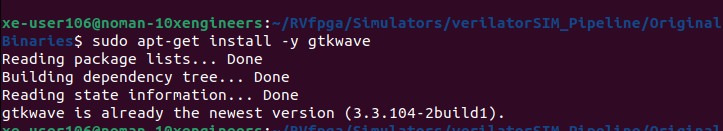
**Section:** Installations **Task:** Tools

**Task 1.4**

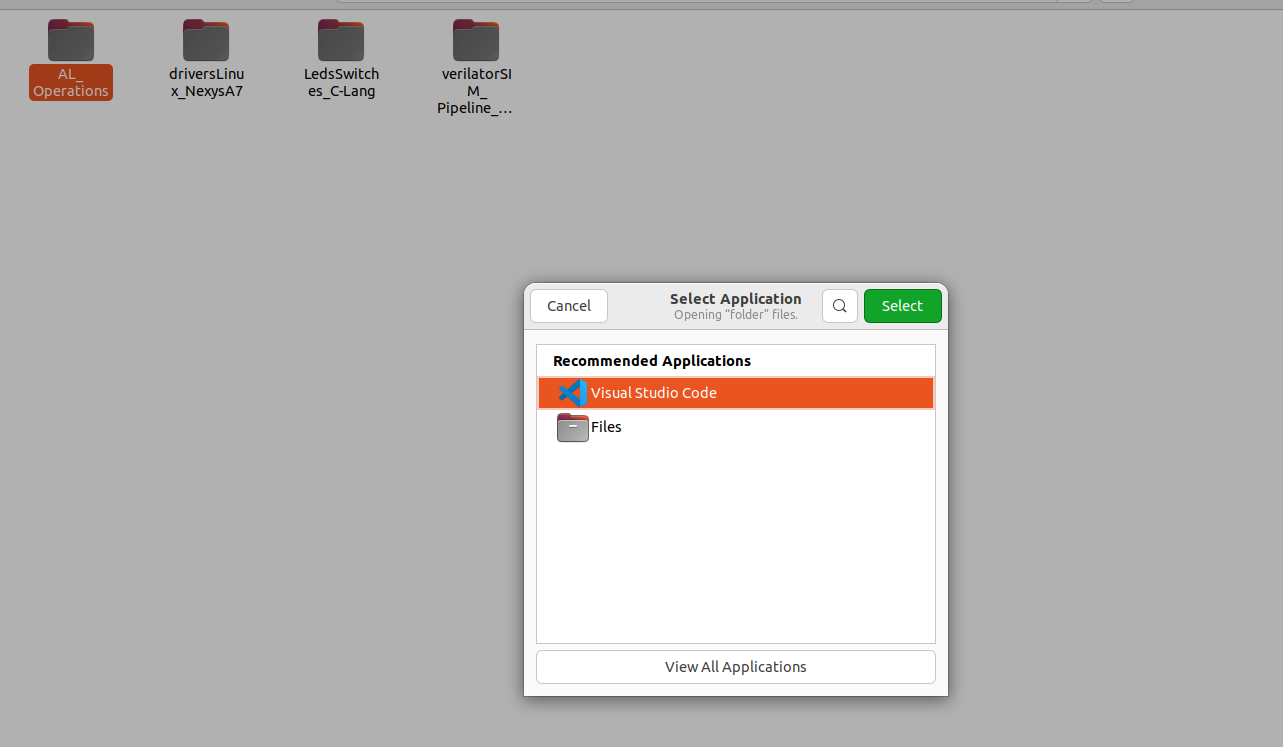
**RVfpga-Trace**

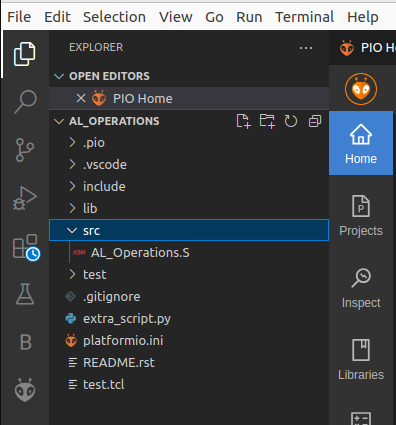
* **Testing:**
  + **RVfpga-Trace**
    1. Install GTKwave by running the following command:

sudo apt-get install -y gtkwave

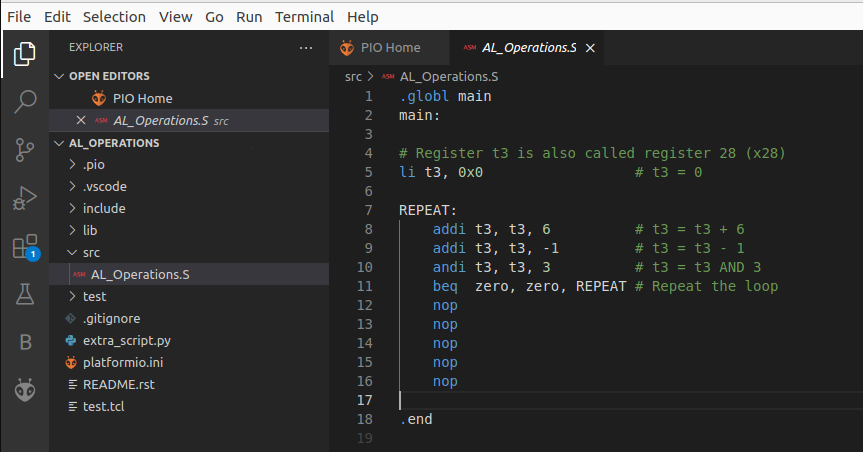


* + 1. Open the specified folder of example program in VS Code:



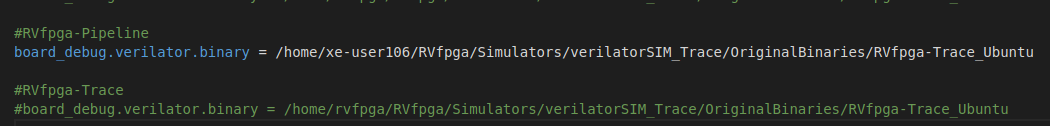


* + 1. PlatformIO will now open this program, which includes three assembly arithmetic-logic instructions (addition, subtraction, and logical and) on the same register, t3 (also called x28), within an infinite loop. We can view the program by expanding the src folder and double-clicking on **AL\_Operations.S**.



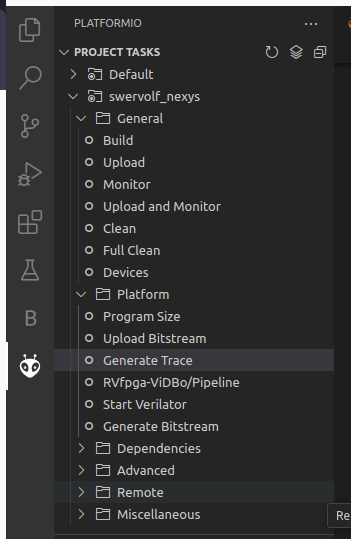
* + 1. Open file **platformio.ini**. Then, establish the path to the provided RVfpga-Trace simulator binary by editing the following line (replaced [Path-To-RVfpga] with the appropriate path in my system:

board\_debug.verilator.binary = /home/xe-user106/RVfpga/Simulators/verilatorSIM\_Trace/OriginalBinaries/RVfpga-Trace\_Ubuntu

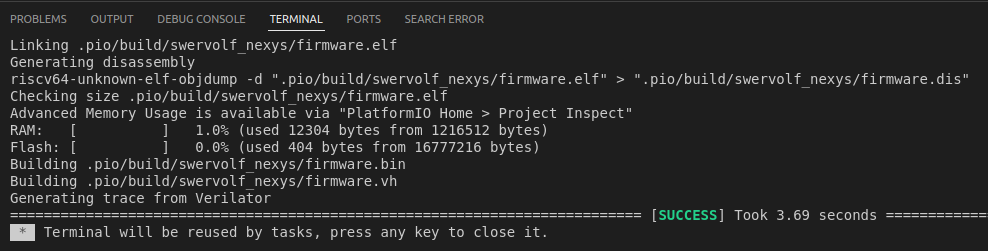


* + 1. Executed the RVfpga-Trace simulator from PlatformIO:
       - Click on the PlatformIO button on the left side.
       - Expand Project Tasks > env:swervolf\_nexys > Platform and clicked on **Generare Trace**.
       - We have to provide execution rights to the binary by running command

chmod +x RVfpga-Pipeline\_Ubuntu

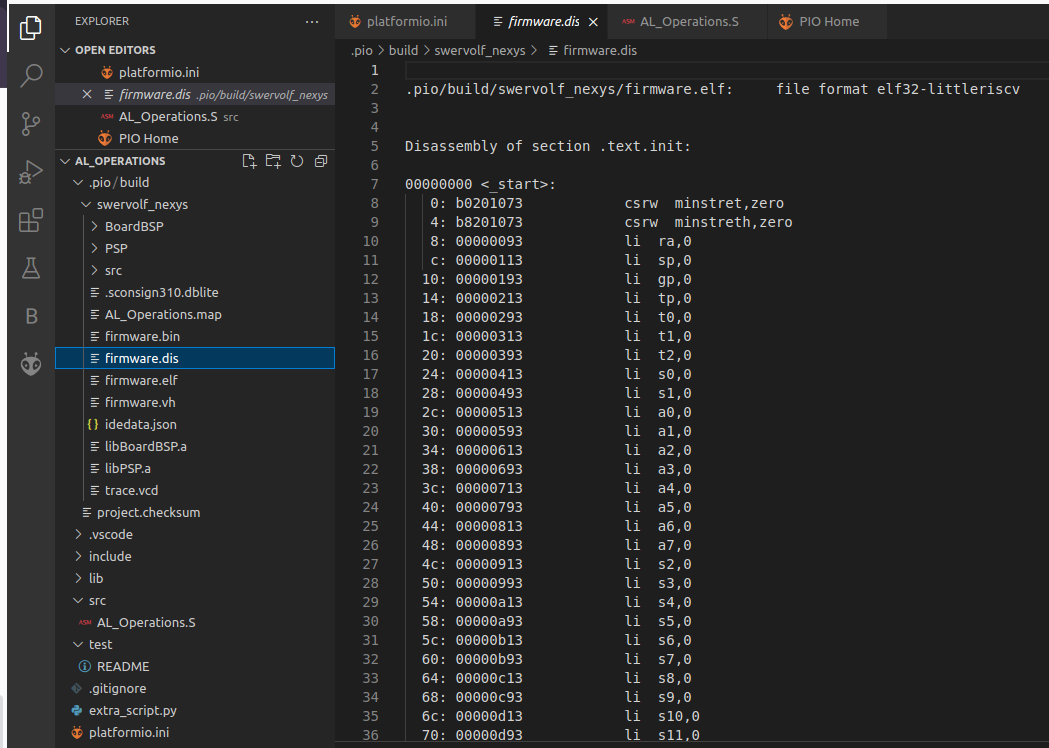


This first compiles the program and then launches the Verilator simulation of the RVfpga SoC running this program.



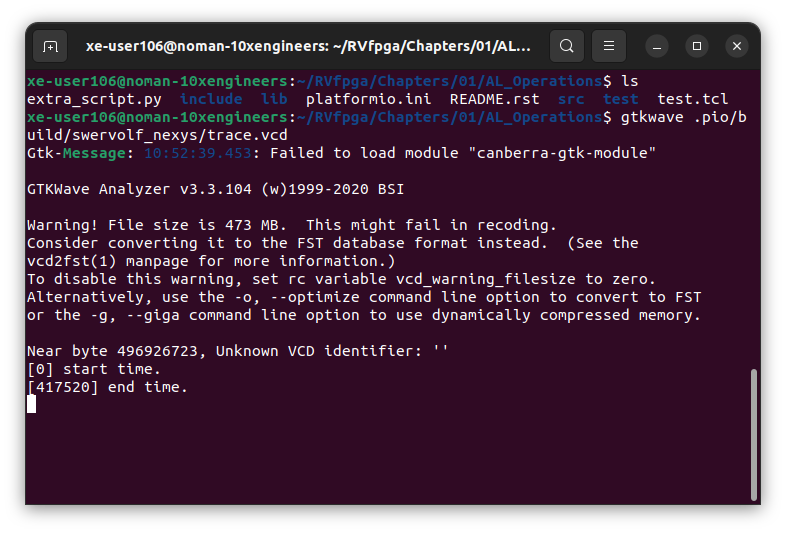
* + 1. A few seconds after the previous step, the **AL\_Operations** program is compiled and file **trace.vcd** is generated inside folder **RVfpga/Chapters/01/AL\_Operations/.pio/build/swervolf\_nexys**.

And a disassembly program has been generated at: **RVfpga/Chapters/01/AL\_Operations/.pio/build/swervolf\_nexys/firmware.dis**.



* + 1. Open the trace with GTKWave by executing the following command in a terminal:

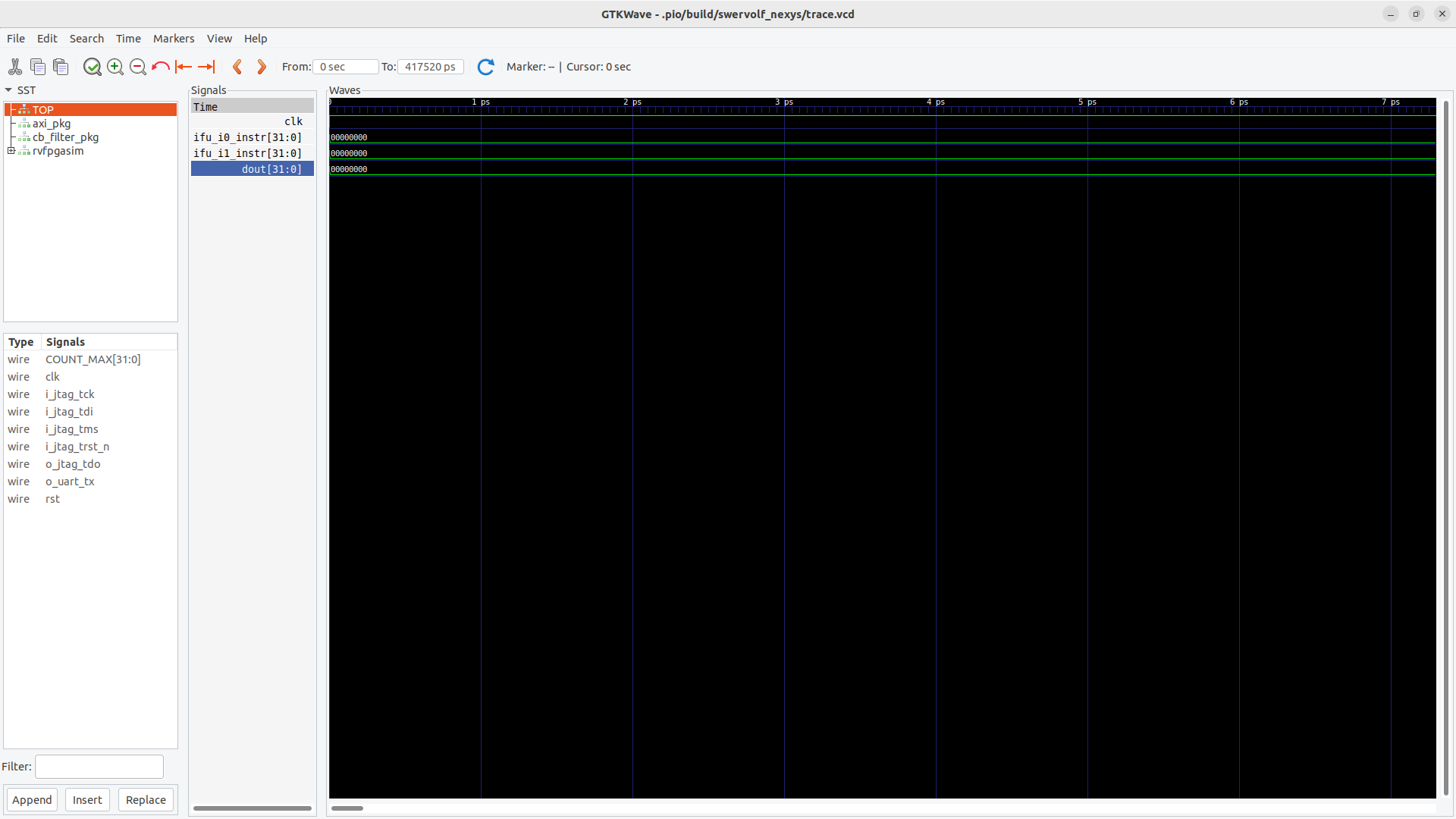
gtkwave .pio/build/swervolf\_nexys/trace.vcd



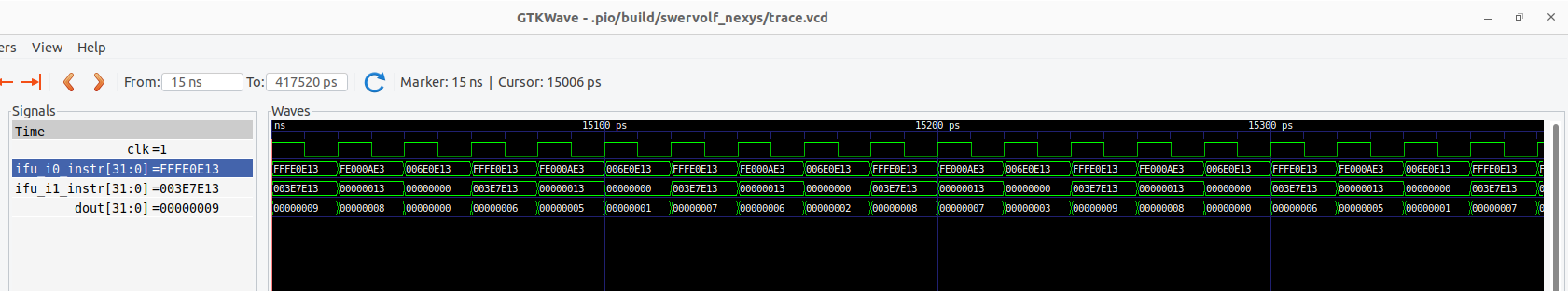
* + 1. Now we will visualize some core internal signals in GTKWave. For that purpose, we will use file **test.tcl**, which is provided at Path-to-RVfpga/RVfpga/Chapters/01/AL\_Operations.

For using this **.tcl** file on GTKWave, we can simply click on File > Read Tcl Script File and select the Path-to-RVfpga/RVfpga/Chapters/01/AL\_Operations/test.tcl file.

This will add the required signals to our GTKWave for visualization.



1. Once the signals are added in GTKWave. And skip the initial instructions in order to analyze the loop containing the three arithmetic-logic instructions; for example, we have selected a point in the simulation from 15ns on), in order to analyze the execution of the three arithmetic-logic instructions of a random iteration of the loop.



1. Note that every cycle two new instructions are fetched, one in each way of the superscalar RISC-V processor, as shown on signals **ifu\_i0\_instr[31:0]** and **ifu\_i1\_instr[31:0]**.

And the effects of the execution of the three arithmetic-logic instructions can be seen on signal **dout[31:0]**.

