

VLSI DESIGN PROJECTS/INTERNSHIP

# College Name

* **SRM Institute of Science and Technology.**

# Company Name

* **Vyorius.**

# Domain

* **VLSI design.**

### Project topics

1. 16-Bit RISC Processor
2. Digital Design

**Software used**

**Hardware,**

# Mimas A7 Mini FPGA Development Board

Mimas A7 Mini is an easy-to-use FPGA Development board featuring Artix 7 FPGA (XC7A35T – FTG256C package) with FTDI’s FT2232H Dual-Channel USB device. It is an Artix-7 based replacement and upgrades of Mimas Spartan 6 FPGA Board. It is specially designed for the development and integration of FPGA based accelerated features to other designs. The USB 2.0 host interface based on popular FT2232H offers high bandwidth data transfer and board programming without the need for any external programming adapters.

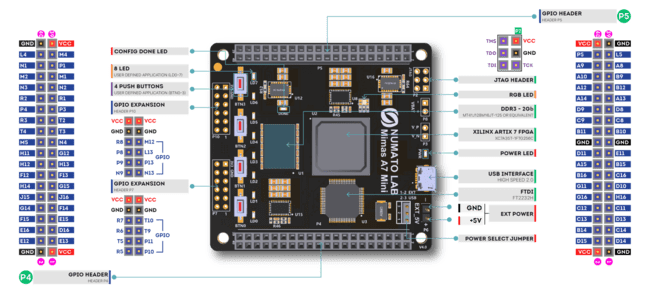
**Features,**

* Device: Xilinx Artix 7 FPGA (XC7A35T-1FTG256C)
* DDR3: 2Gb DDR3 (MT41J128M16JT-125 or equivalent)
* Built-in programming interface. No expensive JTAG adapters needed for programming the board
* Onboard 128Mb flash memory for FPGA configuration storage and custom user data storage
* High-Speed USB 2.0 interface for On-board flash programming. FT2232H Channel B is dedicated to JTAG Programming. Channel A can be used for custom applications
* 100MHz CMOS oscillator
* 8 LEDs, 1 RGB LED and 4 Push Buttons for user-defined purposes
* FPGA configuration via JTAG and USB
* Maximum IOs for user-defined purposes

FPGA – 70 IOs (35 professionally length matched Differential Pairs) and two 2×6 Expansion Headers

**Applications,**

* Product Prototype Development
* Accelerated computing integration
* Development and testing of custom embedded processors
* Communication devices development
* Educational tool for Schools and Universities



#### **Specifications,**

|  |  |
| --- | --- |
| Attribute | Value |
| **Dimensions** | 6 × 4 × 1 in |
| **FPGA** | [**XC7A35T – 1FTG256C**](https://numato.com/fpga/xc7a35t-1ftg256c/) |
| **Memory** | [**DDR3 – 2Gb**](https://numato.com/memory/ddr3-2gb/) |
| **Configuration Options** | [**JTAG**](https://numato.com/configuration-options/jtag/), [**USB**](https://numato.com/configuration-options/usb/) |
| **Host Interface** | [**USB 2.0**](https://numato.com/host-interface/usb-2-0/) |
| **Primary Clock Frequency** | [**100MHz**](https://numato.com/primary-clock-frequency/100mhz/) |
| **Number Of GPIOs (Max)** | [**70**](https://numato.com/number-of-gpios/70/) |
| **Number Of Diff Pairs** | [**35**](https://numato.com/number-of-diff-pairs/35/) |

**Software**



*Icarus Verilog* is a Verilog simulation and synthesis tool. It operates as a compiler, compiling source code written in Verilog (IEEE-1364) into some target format. For batch simulation, the compiler can generate an intermediate form called *vvp assembly*. This intermediate form is executed by the ``vvp'' command. For synthesis, the compiler generates netlists in the desired format.

The compiler proper is intended to parse and elaborate design descriptions written to the IEEE standard *IEEE Std 1364-2005.* This is a fairly large and complex standard, so it will take some time to fill all the dark alleys of the standard, but that's the goal.

*Icarus Verilog* is a work in progress, and since the language standard is not standing still either, it probably always will be. That is as it should be. However, I will make stable releases from time to time, and will endeavour to not retract any features that appear in these stable releases. The quick links above will show the current stable release.

The main porting target is Linux, although it works well on many similar operating systems. Various people have contributed precompiled binaries of stable releases for a variety of targets. These releases are ported by volunteers, so what binaries are available depends on who takes the time to do the packaging. *Icarus Verilog* has been ported to That Other Operating System, as a command line tool, and there are installers for users without compilers. You can compile it entirely with free tools, too, although there are precompiled binaries of stable releases.



**Vivado Design Suite** is a software suite produced by [Xilinx](https://en.wikipedia.org/wiki/Xilinx) for synthesis and analysis of [HDL](https://en.wikipedia.org/wiki/Hardware_description_language) designs, superseding [Xilinx ISE](https://en.wikipedia.org/wiki/Xilinx_ISE) with additional features for [system on a chip](https://en.wikipedia.org/wiki/System_on_a_chip) development and [high-level synthesis](https://en.wikipedia.org/wiki/High-level_synthesis).[[1]](https://en.wikipedia.org/wiki/Xilinx_Vivado#cite_note-Xilinx-Inc-Apr-2012-8-K-1)[[5]](https://en.wikipedia.org/wiki/Xilinx_Vivado#cite_note-5)[[6]](https://en.wikipedia.org/wiki/Xilinx_Vivado#cite_note-hls1-6)[[7]](https://en.wikipedia.org/wiki/Xilinx_Vivado#cite_note-7) Vivado represents a ground-up rewrite and re-thinking of the entire design flow (compared to ISE).[[8]](https://en.wikipedia.org/wiki/Xilinx_Vivado#cite_note-8)[[9]](https://en.wikipedia.org/wiki/Xilinx_Vivado#cite_note-9)[[10]](https://en.wikipedia.org/wiki/Xilinx_Vivado#cite_note-10)

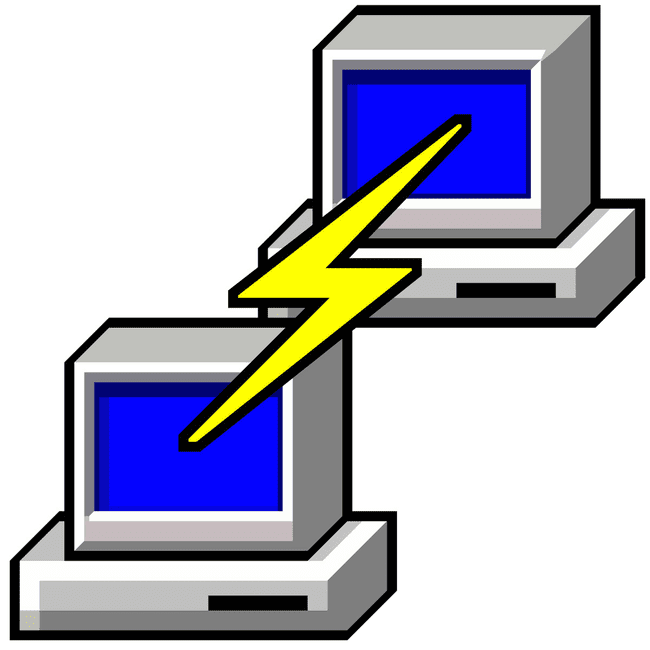
Like the later versions of [ISE](https://en.wikipedia.org/wiki/Xilinx_ISE), Vivado includes the in-built logic simulator [ISIM](https://www.xilinx.com/products/design-tools/isim.html).[[11]](https://en.wikipedia.org/wiki/Xilinx_Vivado#cite_note-11) Vivado also introduces high-level synthesis, with a toolchain that converts C code into programmable logic.[[6]](https://en.wikipedia.org/wiki/Xilinx_Vivado#cite_note-hls1-6)

Replacing the 15-year-old ISE with Vivado Design Suite took 1000 [person-years](https://en.wikipedia.org/wiki/Person-years) and cost US$200 million.



Tenagra is an FPGA System management tool for configuring and communicating with Numato Lab’s supported FPGA modules and development platforms. This software is designed to be a single interface for managing the devices and exercising some of the available features. Currently, Tenagra supports configuring the FPGA module/board (programming) and Memory Exerciser that can transfer data between various memories on the device. This includes both external DDR Memory and Block RAM available within the FPGA device. With Tenagra, you can create multiple configuration setups with different bitstreams and settings for each device model so that switching between multiple bitstreams is a breeze. This is especially helpful during development where the device may need to be reprogrammed with various bitstreams repeatedly.

**Driver for Mimas A7 Mini Module**



PuTTY is a free implementation of SSH **(and telnet) for PCs running Microsoft Windows** (it also includes an xterm terminal emulator). You will find PuTTY useful if you want to access an account on a Unix or other multi-user system from a PC (for example your own or one in an internet cafe).

**Things to know**

# What Is RISC?

A Reduced Instruction Set Computer is a type of microprocessor architecture that utilizes a small, highly-optimized set of instructions rather than the highly-specialized set of instructions typically found in other architectures. RISC is an alternative to the Complex Instruction Set Computing (CISC) architecture and is often considered the most efficient CPU architecture technology available today.

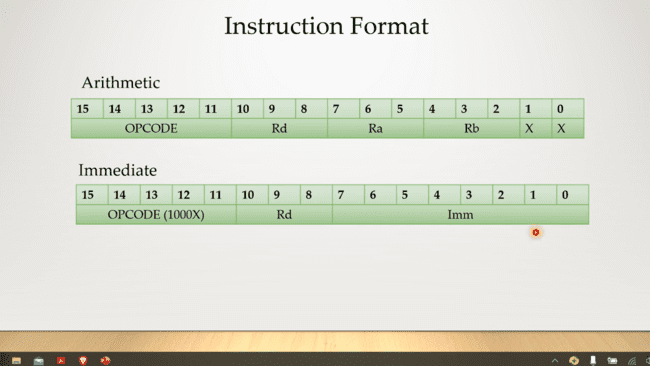
With RISC, a central processing unit (CPU) implements the processor design principle of simplified instructions that can do less but can execute more rapidly. The result is improved performance. A key RISC feature is that it allows developers to increase the register set and increase internal parallelism by increasing the number of parallel threads executed by the CPU and increasing the speed of the CPU's executing instructions. ARM, or “Advanced RISC Machine” is a specific family of instruction set architecture that’s based on reduced instruction set architecture developed by Arm Ltd. Processors based on this architecture are common in smartphones, tablets, laptops, gaming consoles and desktops, as well as a growing number of other intelligent devices.

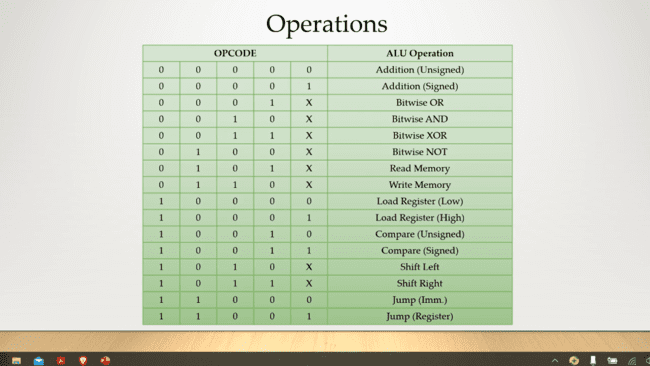
## Why Is RISC Important?

RISC provides high performance per watt for battery operated devices where energy efficiency is key. A RISC processor executes one action per instruction. By taking just one cycle to complete, operation execution time is optimized.

Because the architecture uses a fixed length of instruction, it’s easier to pipeline. And because it lacks complex instruction decoding logic, it supports more registers and spends less time on loading and storing values to memory.

For chip designers, RISC processors simplify the design and deployment process and provide a lower per-chip cost due to the smaller components required. Because of the reduced instruction set and simple decoding logic, less chip space is used, fewer transistors are required, and more general-purpose registers can fit into the central processing unit.





**Design Modules**

1. **Instruction Decoder**
2. **Control Unit**
3. **ALU**
4. **Registers handler**
5. **Program counter**
6. **RAM**

**Testing Modules**

1. **Top test bench**
2. **Instruction Decoder**
3. **Control Unit**
4. **ALU**
5. **Registers handler**
6. **Program counter**
7. **RAM**

**1. 16-Bit RISC Processor**

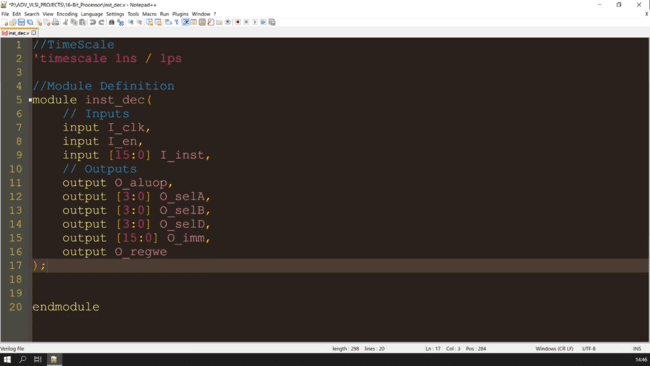
**Software to download**

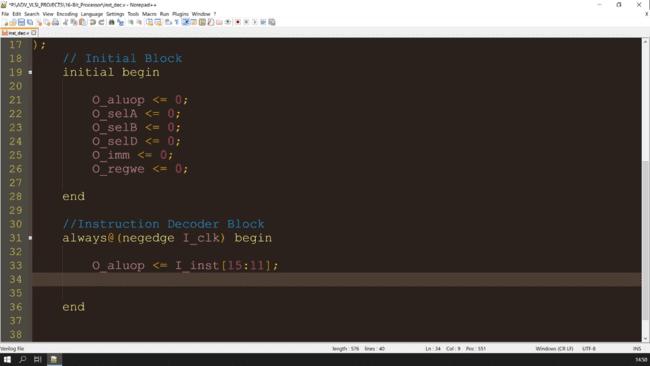
* Notepad++

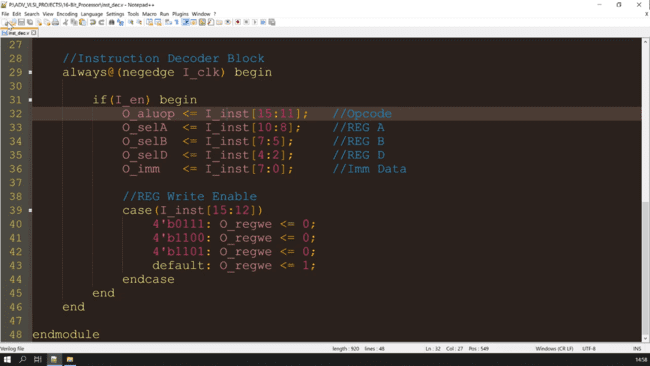
Step 1- Write the verilog program in notepad ++ .

**DESIGN CODES**

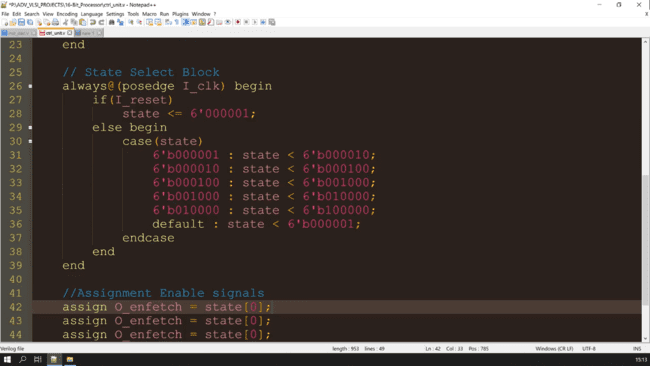
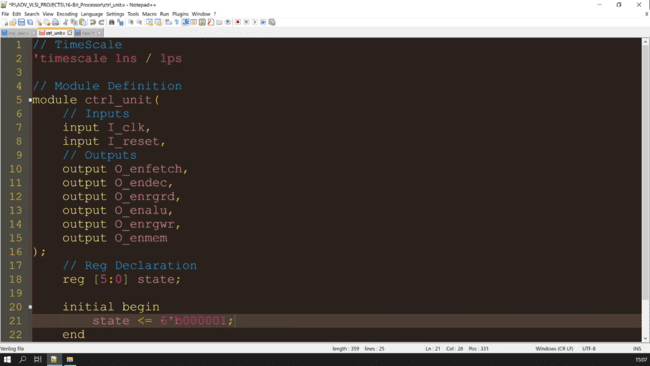
**Module inst\_dec**

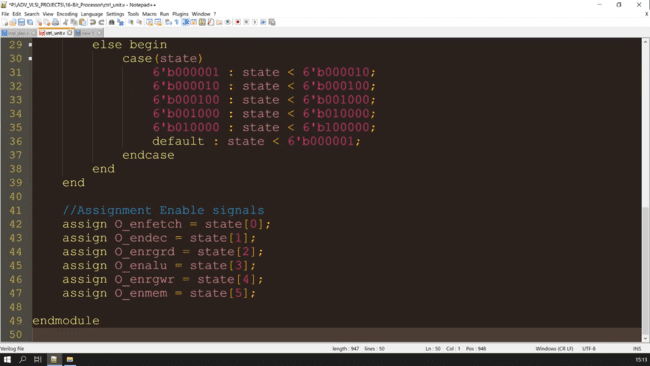




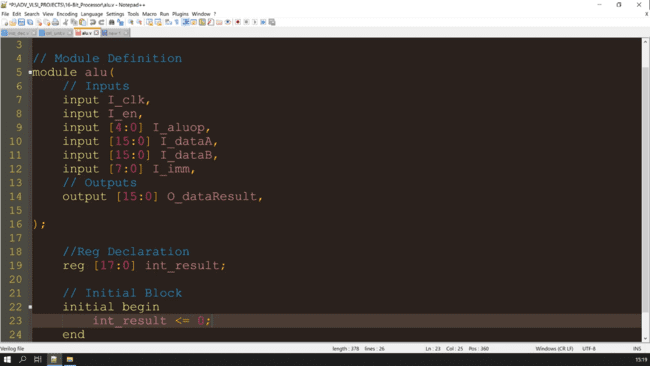


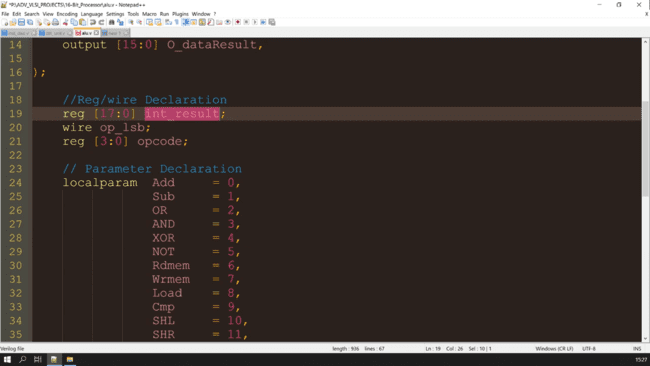
**Module ctrl\_unit**

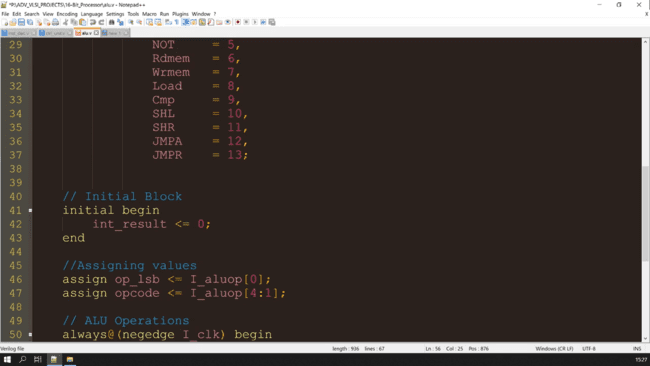


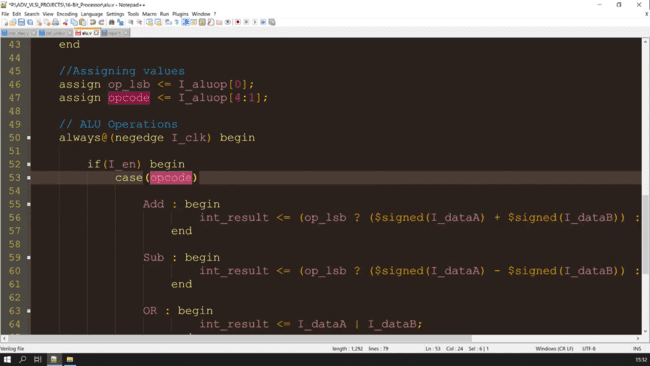


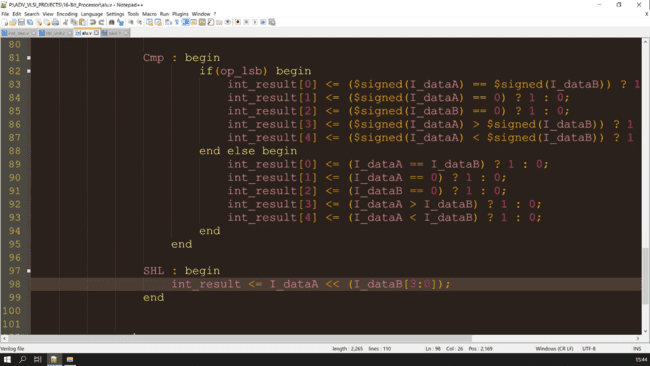
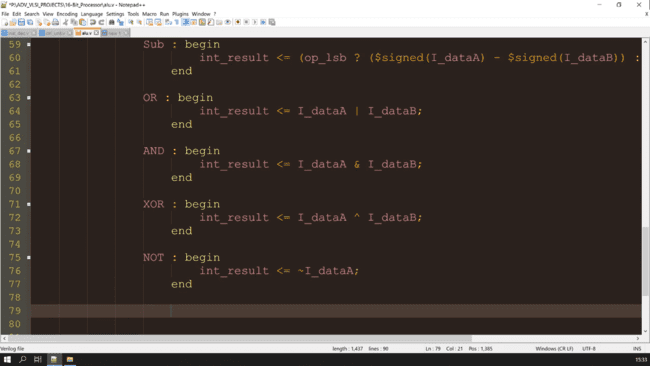
**Module alu**

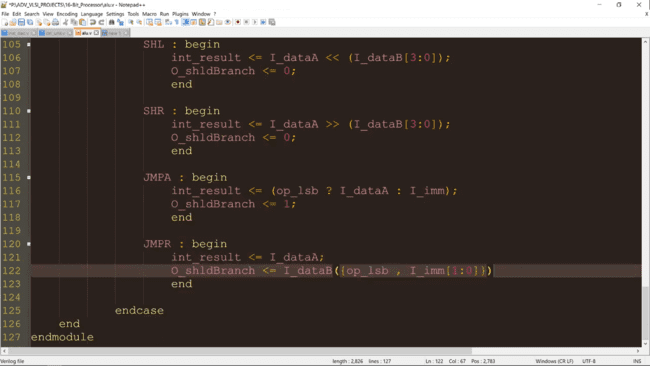
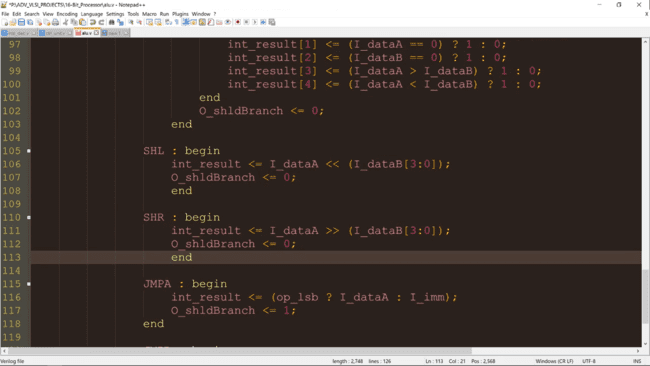






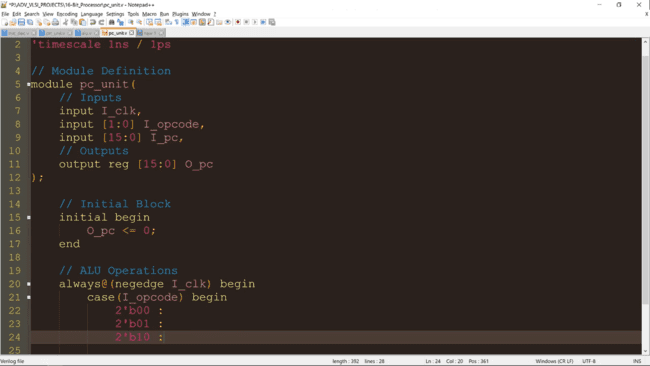


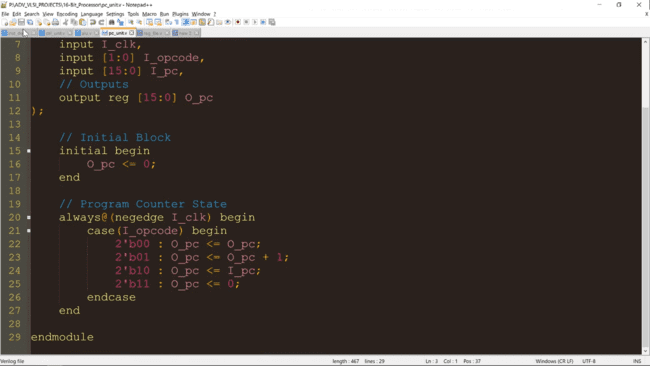


PS- O\_shldBranch defined at the input and also in the ALU operations for the input one as

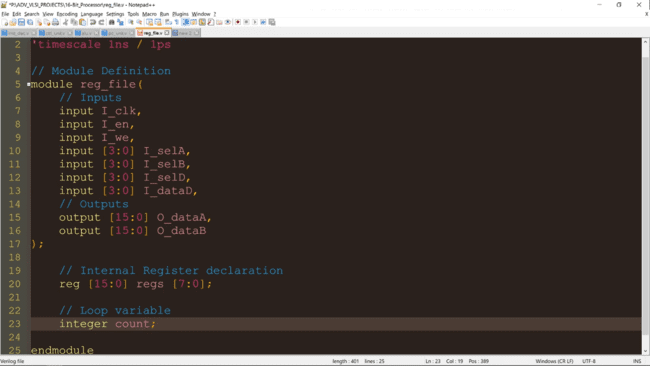
O\_shldBranch <= 0;

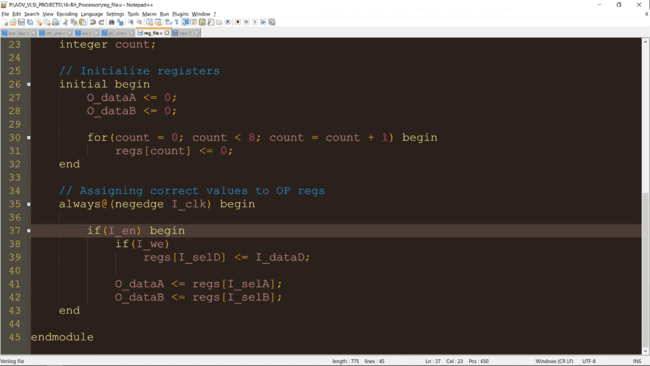
**Module pc\_unit**



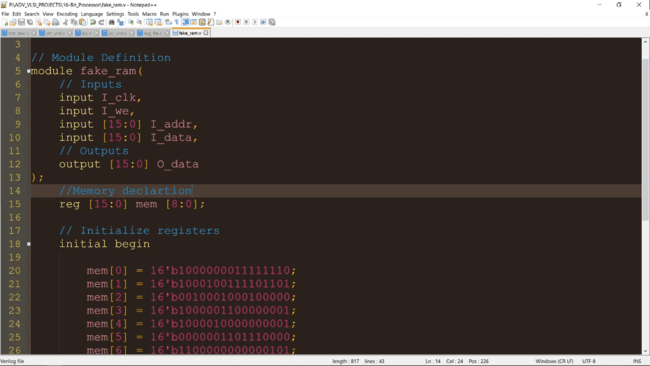


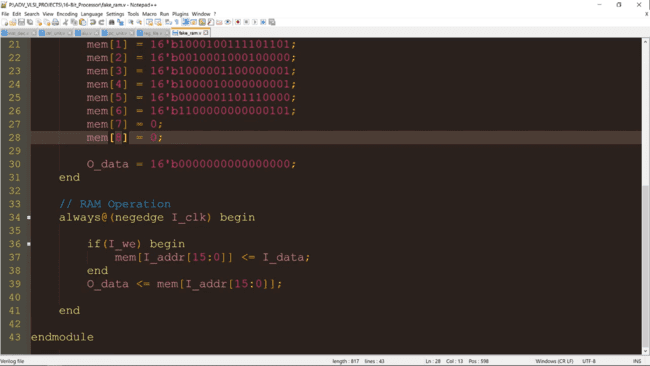
**Module reg\_file**





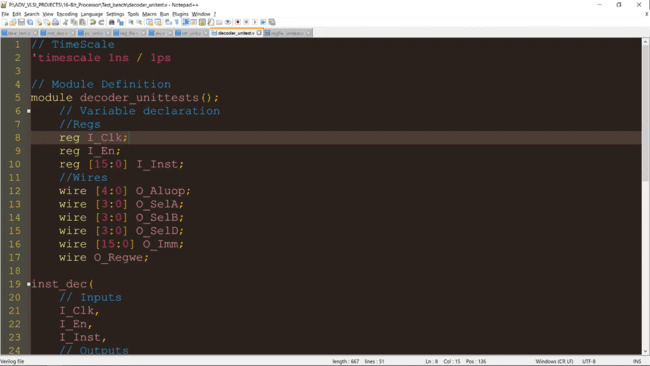
**Module fake\_ram**



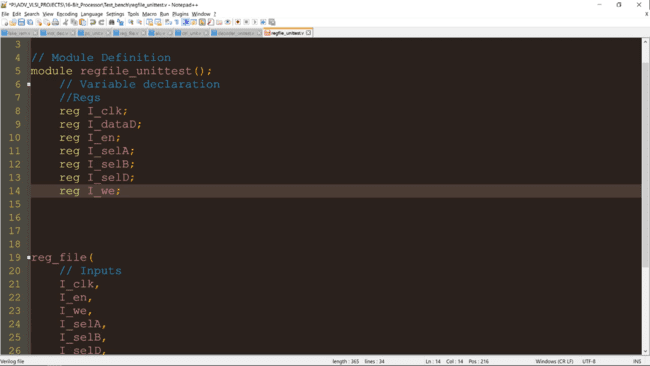


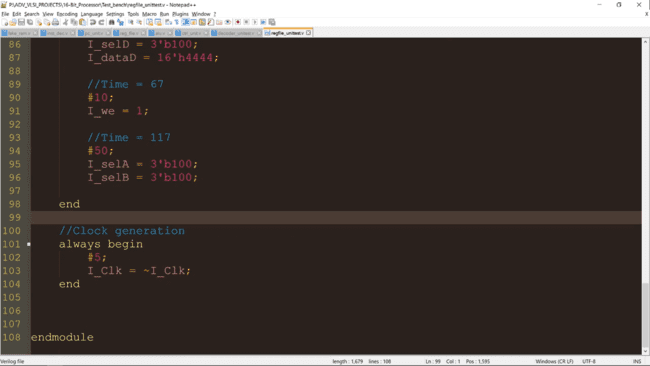
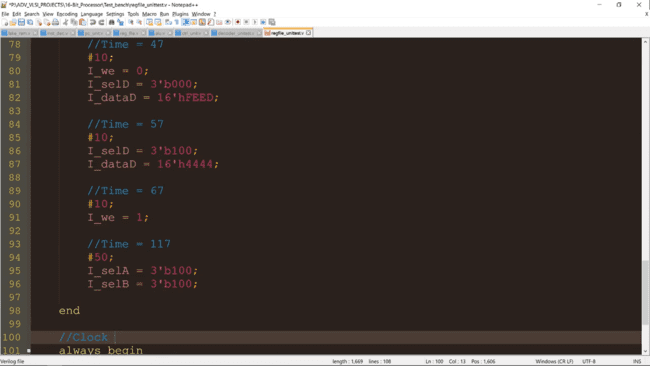
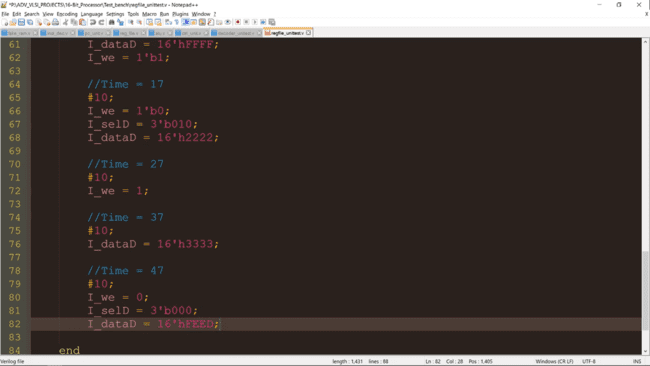
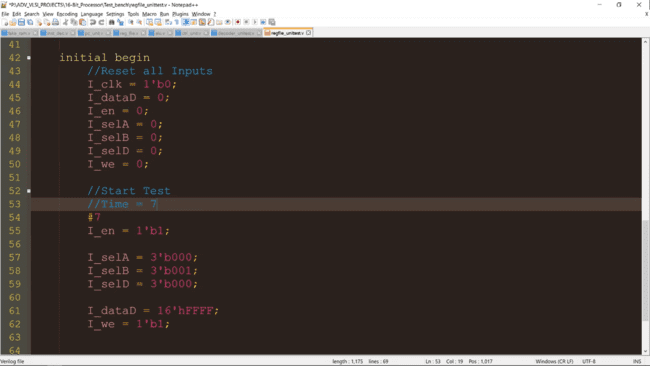
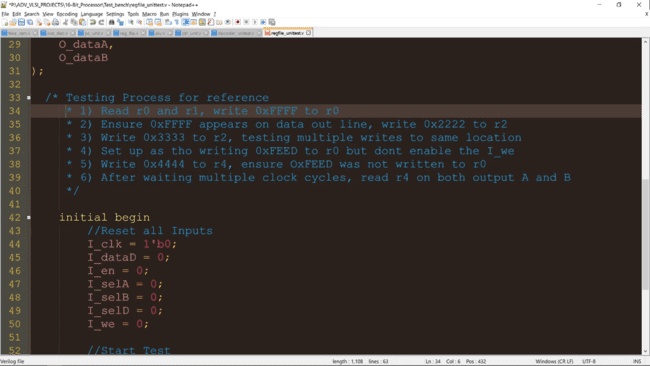
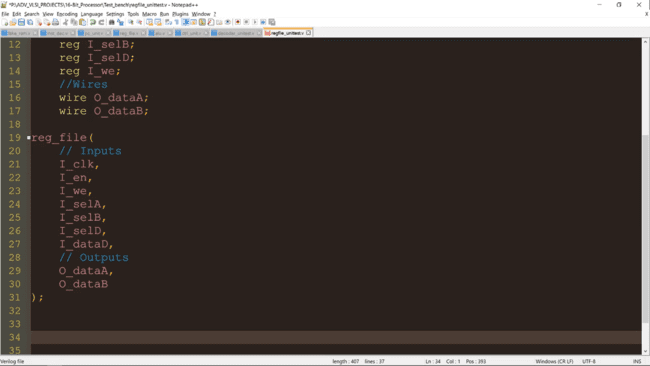
**TESTING CODES**

**Module decoder unit\_test**

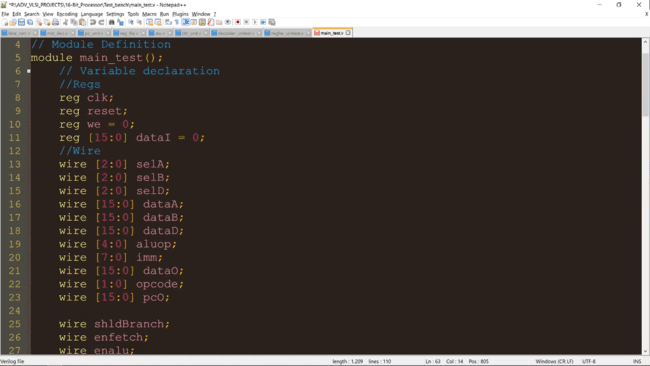


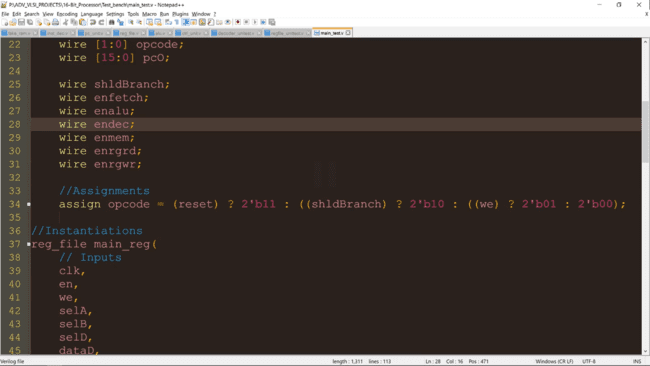
**Module regfile unit\_test**

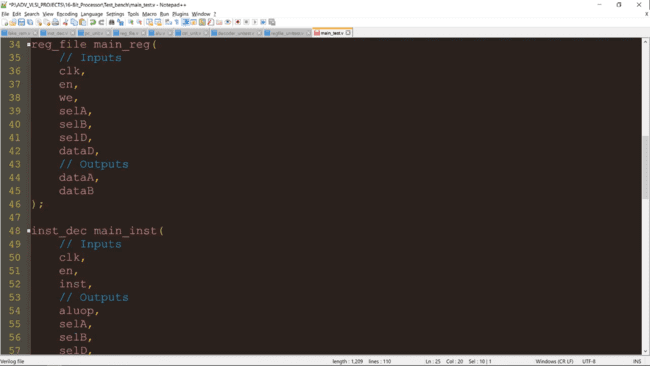
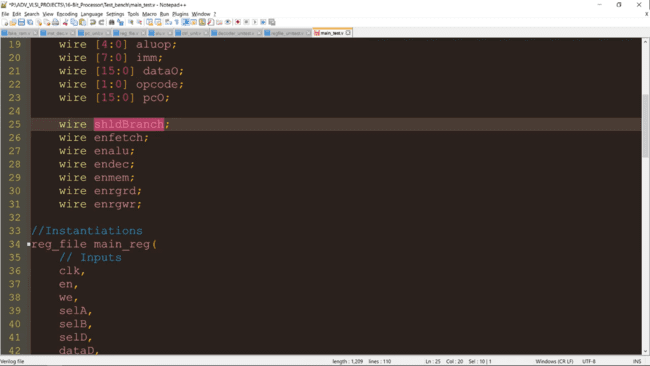


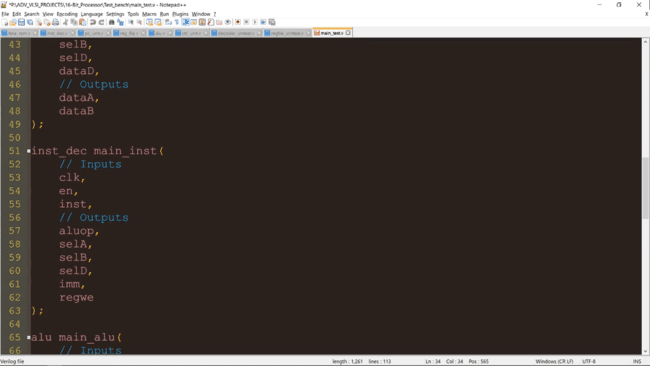


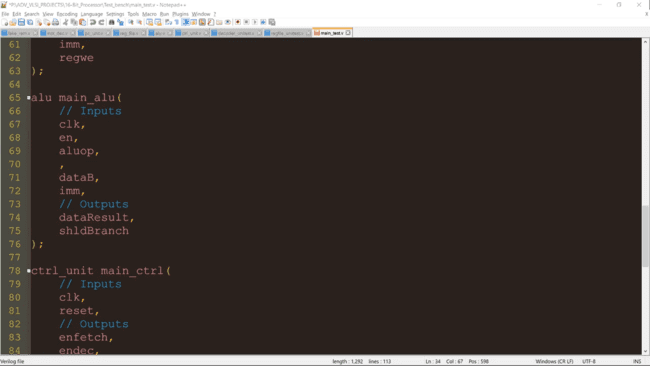
**Module regfile main\_test**

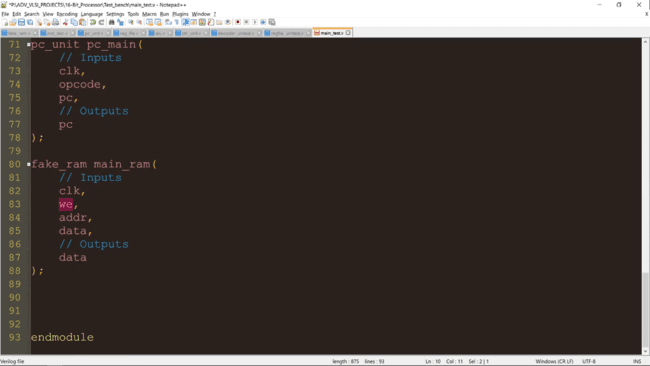
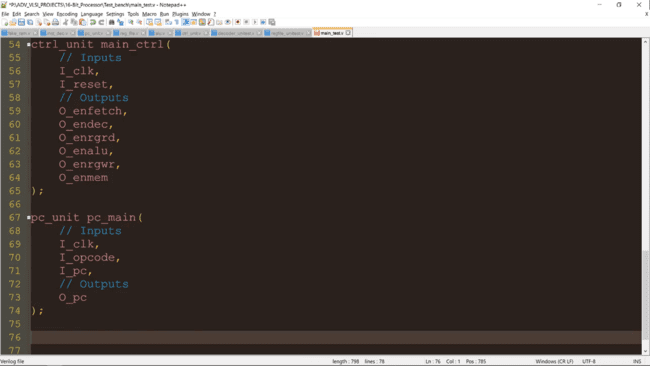


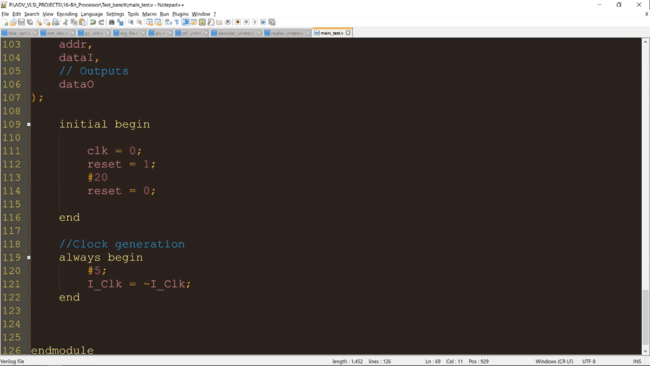








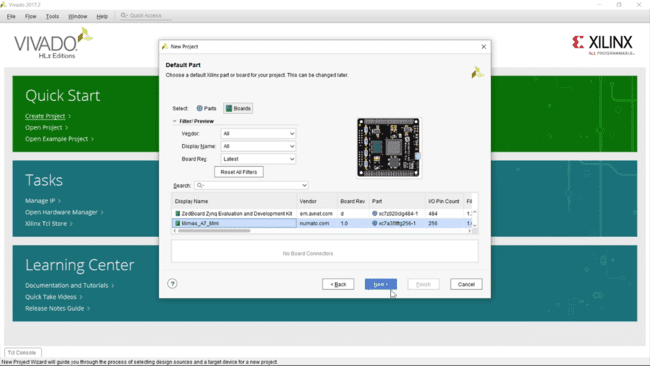


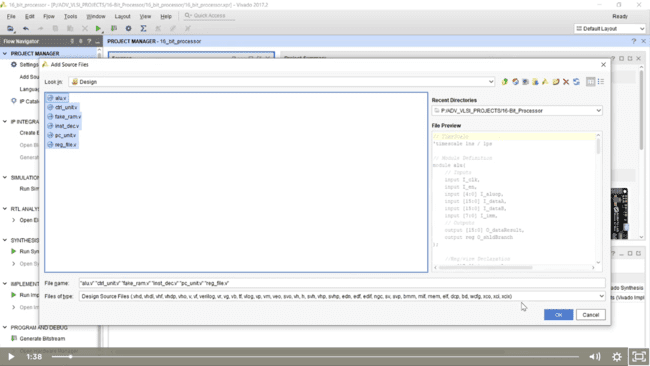


**Time to simulate!**

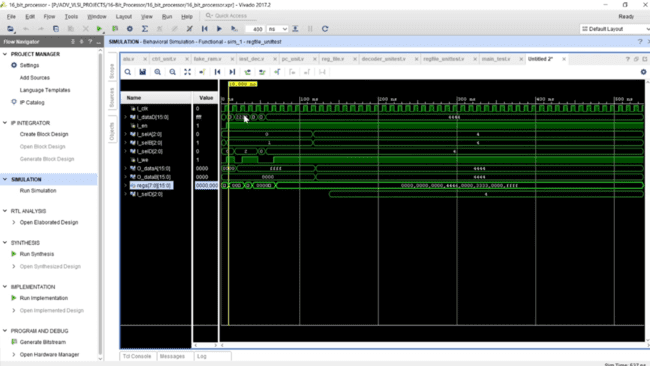
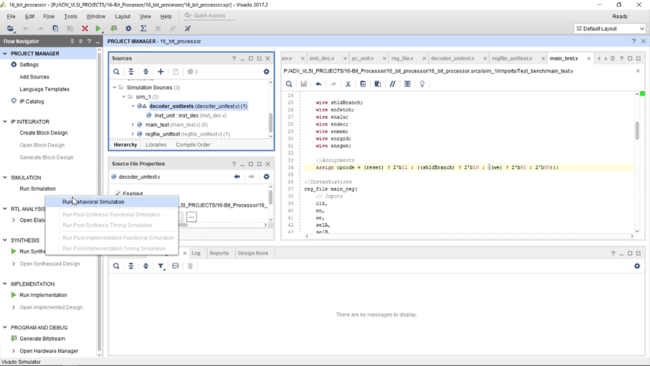
Create a project with a name.

Chose the board, Mimas\_A7\_Mini

Import all the verilog files



After successfully compiling the errors, run Behavorial simulation model of our instruction decoder.

Run the main test bench simulation

