This project was created to teach RISC computer architecture and VHDL to graduate students.

The HDL code implements a simplified subset of the early MIPS instruction set, without considering exception handling. A special instruction "probe" is implemented to route one of the selected register to the external output. Only 8 bits of the 32bit register is exposed, due to the limited number of LEDs onboard. The HDL project is created in Xilinx ISE 14.7, and the target board is the ZedBoard with a ZYNQ-7000 SoC onboard. The original project directory is provided to keep all the project settings, including the user constraint file (.ucf), and all the Xilinx IPs used. The data and instruction memory are implemented using Xilinx IP, and instruction memory is initialized with the content in a .coe file. The generation of the .coe file will be discussed later.

The project can be implemented to generate the bitstream, and the generated bitstream can be used to program the ZYNQ-7000 SoC. When applied to the right target (the ZedBoard), all the 8 onboard LEDs will toggle based on the application code and one of the press buttons can be used to halt/resume the CPU operation. The HDL project contains a lot of Xilinx generated files. If you just want to study the code, look at every file with .vhd extension. With the -1 speed grade of ZYNQ, the project can be implemented at least 100MHz (verified).

Since this is a customized CPU design, the instruction memory needs to be initialized to the right application before synthesis (unlike the MicroBlaze core, who’s application binary and FPGA binary can be separately generated). That means the user of this project will have to manually generate MIPS machine code (‘1’s and ‘0’s) and put them into a .coe file for the Xilinx ROM IP to initialize its content. The assembler project is created to make this simpler. The assembler project is written in C. Using macro definitions, the user can easily generate machine code (in ‘0’ and ‘1’ text) by writing “assembly” code. The main.c file already contains an example to demonstrate how to program in “assembly”. I created the assembler project using Visual Studio, so some of the #include header might be missing if the same code is to be compiled and run on Linux.