ratload
Installation and Use

Jacob Hladky
California Polytechnic State University
San Luis Obispo, CA
jhladky@calpoly.edu

January 31, 2015

1 Introduction

The ratload system consists of two separate components: a computer program and a set of VHDL modules. This guide will detail how to install, configure and run both. It is necesssary to follow the instructions in the *Integration* section, and then either of the *Installation* sections, depending on the target operating system.

1.1 Requirements

If you have a Nexys 2 board — You will need a serial cable. If your computer does not have a serial port, then you will need a USB-to-serial adapter. Here's an example: `http://amzn.com/B0007T27H8`. Any adapter will do as long as you have the right drivers for it.
If you have a Nexys 3 board — No serial cable is required, as an onboard FTDI chip provides serial emulation.

Do not integrate ratload into your RAT CPU until all the major components of the CPU are in place, and you have a good understanding of how they fit together. Understanding how the RAT architecture handles I/O and interrupts will make integration much easier.

1.2 Bug Reporting

All programs have bugs. `ratload` is beta software and is no exception. If you encounter a bug in either the ratload or the winRATLoad program, please contact the author. It is also possible (albeit significantly less likely) that you find a bug in the project VHDL modules. If you do please verify the bug via simulation and report it immediately so it can be fixed.

PLEASE KEEP IN MIND — Since you are writing the mechanics of the CPU itself, it is possible for very odd bugs to happen, and for those bugs to interact with `ratload` in bizarre ways. Although it may seem like there is a problem with `ratload`, it has been thoroughly tested and is almost certainly free of any major bugs.

1.3 Contact

If you have any questions about the project itself, or suggestions for improvement for this guide, please contact the author, at jhladky@calpoly.edu. This project licensed under the MIT license and the complete source code – including the LaTeX source for this guide – is available at `http://www.github.com/jhladky/ratload`.

2 Installation

2.1 Integration on RAT CPU

1. In the Xilinx ISE Environment, go to **Project >Add Copy of Source**. Navigate to the directory "nexys_vhd_files". Select all files in the directory and click **Open**. Another dialog box will pop up confirming you want to add these files. Click **OK**.

   > ⚠ **WARNING** This is a destructive action and will overwrite any files already in your project directory that have a name identical to the files being added e.g. any existing prog_rom.vhd file will be overwritten by the `ratload` prog_rom.vhd file.

2. In the architecture section of the rat_cpu module, edit the componet declaration for the prog_rom module. Add the following line:

   ```
   TRISTATE_IN : in STD_LOGIC_VECTOR(7 downto 0)
   ```

3. In the same file, edit the port map decaration for the prog_rom module. Map the signal for the RAT CPU's tristate bus into to the prog_rom module's tristate bus. The RAT CPU's "tristate bus" may be called the "multibus" in your CPU. (Regardless of its name, this is the bux that connects the ALU, the program counter, the scratch pad, and others.) That line will look similar to this:

   ```
   tristate_in => tristate_bus_sig(7 downto 0),
   ```

4. In the entity declaration for the rat_wrapper module, add the following two lines:

   ```
   rx :   in STD_LOGIC
   tx :  out STD_LOGIC
   ```

5. In the architecture section of the rat_wrapper module, add the following component declaration:

   ```
   component uart is
      Port ( txd      : out STD_LOGIC := '1';
             rxd      : in  STD_LOGIC := '1';
             clk, rst : in  STD_LOGIC;
             int      : out STD_LOGIC;
             data_out : out STD_LOGIC_VECTOR(7 downto 0);
             data_in  : in  STD_LOGIC_VECTOR(7 downto 0));
   end component;
   ```

6. You now need to add the port map for the UART module. Note that how you map this component may differ slightly from what is listed here because of naming conventions, architecture differences, etc. Add the following port map:

   ```
   uart1 : uart port map(
      txd => tx,
      rxd => rx,
      clk => clk,
      rst => rst,
      int => int_sig,
      data_in => uart_out_sig,
      data_out => uart_in_sig);
   ```

   The following is a line-by-line breakdown of the port map:

   - `txd`: Transmit port for the UART, goes out of the rat_wrapper and onto the physical UART on the Nexys board.

- `rxd`: Receive port for the UART. Similar to the transmit port.
- `clk`: Clock for the UART. Hook it up to yer clock. 🕘
- `rst`: Reset for the UART. Hook it up to the RAT's global reset
- `int`: The UART generates an interrupt when data has been received or sent. You need to create an interrupt signal, and **OR** that signal with any other interrupt you might be using e.g. a mouse or ethernet interrupt. The signal should then be sent to the rat_cpu module.
- `data_in`: Data going into the UART, and out of the rat_cpu module. Create an internal signal called uart_out_sig (or some other name consistent with your own naming convention), and integrate with whatever system you are using to manage your outputs to the CPU. For example, if you are have separate input and out modules, then send the uart_out_sig into that module. If you manage your I/O in the rat_wrapper module, then add an `if` or `elsif` block for the UART.
- `data_out`: Data going out of the UART, and into the rat_cpu module. As above, add this to whatever system you have in place to manage inputs.

You will need PORT_IDs for the data_in and data_out signals in the UART. For data going out of the UART, and into the CPU, the port ID is 0x0F. For data going into the UART and out of the CPU, the port ID is 0x0E.

7. In the rat_wrapper.ucf (your title may differ slightly) file, add the following NETs if you have a Nexys 2 board:

```
NET "tx" LOC = P9;
NET "rx" LOC = U6;
```

   If you have a Nexys 3 board, then add these lines instead:

```
NET "tx" LOC = XX;
NET "rx" LOC = XX;
```

8. Synthesize your newly integrated system.

> ⚠ **WARNING** You **WILL** receive the following warning about a latch:
>
> `Found 1-bit latch for signal <TBE>. Latches may be ...`
>
> This is a result of a flaw in the Digilent Romania provided UART module. It is safe to ignore. Any other latch, however, is not the result of this flaw and must be fixed!

9. Generate the bit file for the system, This finishes the integration steps.

2.2 Installation on GNU/Linux or OS X

The two operating systems (or families of operating systems) listed in the sub-section header are only three specific examples of OSes that should be able to run the `ratload` program. It should build and run on any POSIX-compliant OS.

To build the ratload program, `cd` into the 'ratload' directory and build the program with `make`. Obviously you must have some sort of build environment set up for this to work. Mac users should have XCode tools installed or get gcc or clang via homebrew/macports/fink, etc. GNU/Linux users should install gcc via their package manager. If you're using a Ubuntu based distribution, such as Linux Mint, or Ubuntu itself, then the following will get everything you need:

```
sudo apt-get install build-essential
```

After building the binary, there is no installation step. Simply copy it to some convenient location (such as a personal bin folder).

2.3 Installation on Windows

Windows users must use the winRATLoad program

Copy the “winRATLoad” folder to somewhere useful for you. The winRATLoad executeable must remain in the folder to function properly. The program is otherwise self contained and no other installation is necessary.

2.4 Serial Configuration

2.4.1 Nexys 2

If you have a serial port on your computer, congratulations, you are using a computer from the 20th century! Your OS almost certainly already has installed drivers to use this port, so you don’t need to do anything else.

If you don’t have a serial port, then you need to obtain a USB-to-serial adapter cable. If you’re on Windows, the manufacturer of the cable probably provided drivers for you to install along with the cable itself, make sure you install them. If you’re using GNU/Linux or OS X, google around to try to find the right driver.

Once you think you’ve installed the driver then you simply need to verify that the OS can see it. In Windows 7, go to the Start menu and right click Computer, and then click manage. Look for the serial device in the device management section of the console. In GNU/Linux or OS X, list the contents of the `/dev` folder. In GNU/Linux your serial adapter will be called something similar to “ttyS0”. In OS X, it will look like “tty.usb0”, or similar.

2.4.2 Nexys 3

If you have a serial port on your computer, it doesn’t matter, you can’t use it! The Nexys 3 doesn’t have a physical serial port, but instead emulates one with an FTDI chip. The USB cable you use to power and program the board also functions as a serial cable when it needs to. This means you need to install an FTDI driver. FTDI drivers for OS X and Windows 7 are included in the project folder. Install them and be glad you don’t have to deal with USB-to-serial cable drivers!

3 Use

Up until now, you've probably followed this pattern when developing your assembly language programs:

1. Run your assembly repeatedly in the ratsim program until it produces a prog_rom.vhd file and seems to be relatively bug free.
2. Add the new prog_rom.vhd file to your project, replacing the old one.
3. Resynthesize the entire project and reprogram the bit file onto your Nexys board.
4. Repeat ad infinitum.

The steps you have just followed to integrate the vhd files into your RAT CPU, and to install the ratload program onto your computer, will dramatically change this pattern. From now on you do not need to resynthesize your project when ratsim generates a new prog_rom.vhd file, nor is it necessary to copy that new file into your project. In fact, **making any more changes to the prog_rom.vhd file in your project directory will break `ratload`**.

The following is a general overview of the new pattern you need to follow in order to use `ratload` properly:

1. Program your Nexys board with the generated bit file.
2. Without disconnecting anything else, connect the Nexys 2 board to your computer via the serial cable. **If you have a Nexys 3 board, ignore this step.**
3. Start either the winRATLoad or the ratload program, depending on your OS. Select the proper serial device and prog_rom.vhd file to read.
4. The program will then communicate with the Nexys board and send the prog_rom.vhd to your RAT CPU via the serial connection. Once the program displays a success message, the serial cable (again, only if you have a Nexys 2) can be disconnected and the board used normally. You can treat the program running on the board like it was synthesized with the system using the previous pattern.
5. To send a new prog_rom.vhd file you must power cycle the Nexys board (If you programmed your bit file into volatile memory, then you'll need to reprogram it as well.) Then repeat this procedure.

3.1 Using winRATLoad (Windows)

3.2 Using ratload (GNU/Linux or OS X)

ratload takes exactly two arguments: "-d" to specify a serial sevice and "-f" to specify a prog_rom.vhd file to parse. You'll probably have to run it as root in order to access the serial device. The following is an exlpanation of all the errors messages ratload can produce:

- "Opening prog_rom failed": Ratload could not open your prog_rom.vhd file. Perhaps it couldn't find it, or it didn't have permission.
- "Opening serial device failed": Ratload could not open the serial device you specified. Perhaps you specified it incorrectly, or ratload does not have permission to access it.
- "Invalid prog_rom.vhd file, exiting.": Ratload was able to find and open the file you specified, but it couldn't parse it. Ratload expects the prog_rom.vhd to be structured in a very specific way. Because this file is auto-generated by the ratsim program, this is not a problem. Make sure you are specifying the exact prog_rom.vhd file that ratsim generates. If you continue to receive this error, try generating the prog_rom.vhd file again.
- "Serial Configuration Failed": Ratload was able to find and open the device you specified, but when it failed to configure it. It is possible but extremely unlikely that you have a serial device that does not support the proper settings. It is much more like that you specified a valid but incorrect device.

- “Error communicating with Nexys2 board.”: Ratload was able to open and parse the file you specified, and was able to open and configure the serial device you specified, but it received no or incorrect data from the Nexys board. Program the “reference_rat_wrapper.bit” file onto your Nexys board and try to send data to it wih ratload. If that works, then you have misconfigured your RAT CPU, and you need to return to the integration section and make sure you followed those steps correctly.
- “Too many [few] arguments.”: You specified the arguments to ratload incorrectly.
- “Option not supported. Only -f and -d supported.”: You specified the arguments to ratload incorrectly.