Bootloader

Jesse E.J. op den Brouw*

The Hague University of Applied Sciences

June 26, 2022

https://github.com/jesseopdenbrouw/riscv-minimal

For design `riscv-pipe3-csr-md-lic.bootloader`

Abstract

The RISC-V RV32IM three-staged pipelined processor has a spin-off that incorporates a hard-coded bootloader. The bootloader is started at reset and waits for 5 seconds to be contacted by an upload program. If contacted, an S-record file is uploaded to the ROM (or RAM, but programs can only started from the ROM). After uploading, the application is either started or the bootloader falls to a simple monitor program.

This processor is not intended as a replacement for commercial available processors. It is intended as a study object for Computer Science students.

This is work in progress. Things will certainly change in the future.

*J.E.J.opdenBrouw@hhs.nl

Contents

1 Bootloader

Design `riscv-pipe3-csr-md-lic.bootloader` incorporates a hard-coded bootloader with an upload and a simple monitor program. The bootloader is placed in a separate ROM starting at address 0x10000000 and has a maximum length of 4 KB (may be extended). The bootloader cannot be overwritten by an upload.

2 S-record file

The S-record standard is invented by Motorola in the 1980's. It consists of formatted lines, called records. Each line can be seen as a record. A record starts with `S` followed by a single digit. `S0` is used as header record. This record is ignored by the bootloader (skipped). `S1`, `S2` and `S3` are data record using a 2-byte, 3-byte and 4-byte start address respectively. `S4` is reserved and skipped by the bootloader. `S5` and `S6` are count records and are ignored. `S7`, `S8` and `S9` are termination records with a start address incorporated, with 4-byte, 3-byte and 2-byte address respectively. This start address is used by the bootloader to start the application. Records have a checksum at the end, this checksum is ignored by the bootloader.

3 Startup sequence

After loading the design in the FPGA, or after resetting the FPGA, the bootloader starts. It presents itself with a welcome string printed via the USART at default 9600 bps. Then the bootloader waits for about 5 seconds before starting the application at address 0x00000000. During these 5 seconds, at half second interval, a `*` is printed via the USART. At the same time, the 10 red leds on the DE0-CV board are lit and dimmed on half second interval from left (high led) to right (low led). If a character is received within the five seconds, either a S-record file can be uploaded or the bootloader falls to a simple monitor program.

4 Uploading an S-record file

A Motorola S-record file can be uploaded with the `upload` program found in the `CODE` directory. It is tested on Linux, Windows is currently not supported. S-record files for all RISC-V programs are generated as part of the `make` process by the RISC-V `objcopy` program. The `upload` program is invoked with:

```
upload -d <device> -t <timeout> -s <sleep> -v -j file
```

The default device is `/dev/ttyUSB0` which is the first plugged-in USB-to-USART converter. Timeout is the time the `uplead` program waits for expected data from the bootloader. The time is set in deciseconds (0.1 seconds) intervals. The default value is 5. Sleep is the time the `upload` program waits after transmitting a character tor the bootloader in microseconds intervals. The default value is 0. The option `-v` turns on verbose

mode. The option `-j` instructs `upload` to send a "start application" command to the bootloader after the S-record file is uploaded. File must be a valid S-record file.

To upload an S-record file, reset the FPGA or program the FPGA design in the FPGA. Then, within the 5 seconds interval, start the `upload` program with options and file name supplied. If the `upload` programs manages the contact the bootloader, the S-record file will be uploaded. Depending on the size, uploading may take as short as a few seconds to minutes for a large file. As a rule of thumb, about 700 file characters per seconds are send. Make sure that *no* terminal program (e.g. Putty) is active. If the `upload` program cannot contact the bootloader, it exits with an error message. If during sending the records, a response from the bootloader is not read, the `upload` exits with an error message. This is mostly due to an open terminal connection. To start the application after the upload, supply the `-j` option to the `upload` program, otherwise the monitor is started.

5 Using the monitor

If within the 5 seconds grace period a character is received by the bootloader, the bootloader falls to a simple monitor program. The monitor recognized some simple commands. Each command is terminated by an enter key.

`r`

Run the program a address 0x00000000.

`rw <address`

Read and print word af address. Address must be on a 4-byte interval. Data is presented in big endian.

`dw <address>`

Dump 16 words from memory to the terminal. Address must be on a 4-byte interval. After each word, 4 ASCII characters are printed, if printable. If not printable, a dot is printed. Useful for finding strings in memory. Data is presented in big endian.

`n`

Dump next 16 words from memory to the terminal, and ASCII characters.

`rw <address> <data>`

Write 4-byte data at address. Address must be on a 4-byte interval. Data must be in big endian.

`h`

A simple help menu is presented.

Note: the capabilities of the monitor may be extended.

6 Upload protocol

Uploading an S-record file uses a simple handshake protocol. The `upload` program sends a single exclamation mark (`!`). The bootloader responds with an question mark (`?`) and a newline (`\n`). Now each S-record line is transmitted character by character, including the end-of-line termination character (`\r` and/or `\n`). After a line is processed, the bootloader responds with a question mark and a newline. After all S-record lines are transmitted, the `upload` program either sends a `J` to start the application, or a `#` to start the monitor.

7 Implications on the hardware design

The design has a separate ROM that incorporates the bootloader. The original ROM, at address 0x00000000 is extended with a write port, together with the instruction read port and the data read port. In fact, the ROM has become a (program) RAM. Because the FPGA ROMs (and RAMs) can only have two ports (out/out or in/out), the original ROM hardware is duplicated (by the synthesizer). This takes up some onboard RAM blocks, but very few ALMs (cells). The speed decrements by a few MHz.

Note that the ROM can only be (over)written with words on a 4-byte boundary.