Imparé

Imparé RISC-V Reference Model User Guide

System Requirements & Setup Overview

Before beginning installation and simulation, ensure your system meets the following requirements

- A Linux system (Rocky, AlmaLinux, CentOS, RHEL, Ubuntu, Debian, or openSUSE).
- The Synopsys VCS simulator installed and licensed — required to compile and run the Verification IP simulation.

This user manual guides clients through the exact information required to obtain a license and how to retrieve MAC addresses from their devices.

Required Information (Step-by-step)

1. **Select Verification IP**
   Tell us which Verification IP you want access to (name/version).
2. **License Expiry Date**
   Provide the expiry date for the license (e.g., 2025-12-31).
3. **Number of Devices**
   Specify on how many devices the license should be active.
4. **Device MAC Addresses**
   Provide the MAC address for each device (one MAC per device).

Important notes:

Make sure MAC addresses are entered exactly (hex digits and separators). Examples: 00-1A-2B-3C-4D-5E or 00:1a:2b:3c:4d:5e.

Finding Your Device's MAC Address

Below are simple step-by-step instructions for common platforms. Each section includes an illustrative screenshot to help layman users.

Imparé RISC-V Reference Model User Guide

Linux (Desktop / Server)

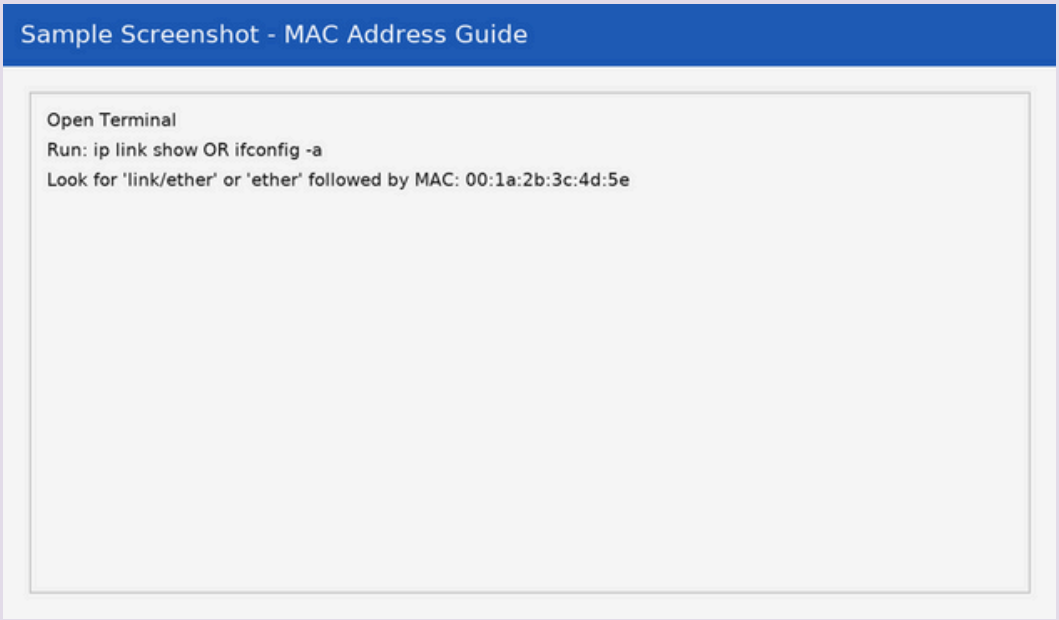
1. Open Terminal.

2. Run one of the commands:

ip link show

ifconfig -a

3. Look for 'link/ether' or 'ether' followed by the MAC address.



Libraries Installation Guide

This guide provides a complete and easy-to-follow installation procedure on any Linux distribution, including minimal, desktop, and enterprise environments.

After you have provided details, you will be provided with complete package.

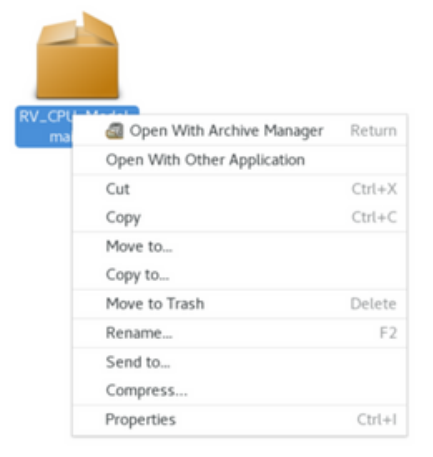
This guide will help you check and install required libraries for the IP package provided, step by step.

Imparé

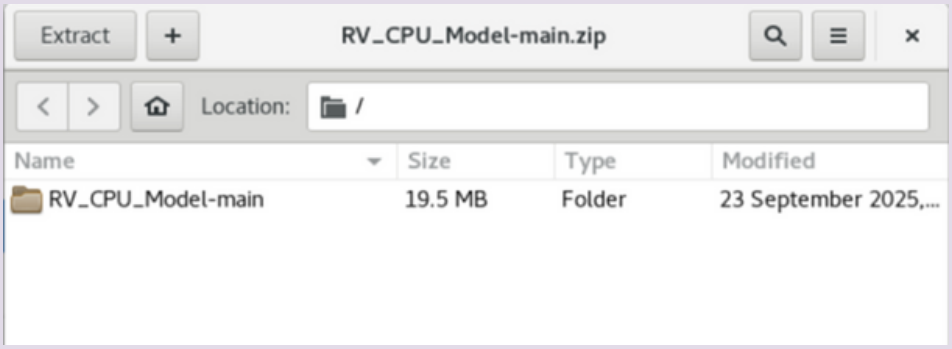
Imparé RISC-V Reference Model User Guide

1. Extract the IP Package

- Once you receive the IP package, right-click on the compressed file.
- Select “Open with Archive Manager.”



- Click Extract to unzip the contents.



- After extraction, open the main IP folder.

Inside this folder, you’ll find several subfolders

2. Locate the Verification Folder

- Open the folder named **verif**.

- Inside it, locate and open the **tb** folder.

- This folder contains the libsclic.so file that has the information of libraries requirements

- Open the terminal in the sim folder.
- Run in terminal:

```
ldd libsclic.so
```

Imparé RISC-V Reference Model User Guide

```
[redacted] ldd libsclic.so
        linux-vdso.so.1 (0x00007fff01c4b000)
        libcrypto.so.3 => /lib/x86_64-linux-gnu/libcrypto.so.3 (0x00007f8b24f38000)
        libssl.so.3 => /lib/x86_64-linux-gnu/libssl.so.3 (0x00007f8b24e94000)
        libcurl.so.4 => /lib/x86_64-linux-gnu/libcurl.so.4 (0x00007f8b24ded000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f8b24bc4000)
        libnghttp2.so.14 => /lib/x86_64-linux-gnu/libnghttp2.so.14 (0x00007f8b24b9a000)
        libidn2.so.0 => /lib/x86_64-linux-gnu/libidn2.so.0 (0x00007f8b24b77000)
        librtmp.so.1 => /lib/x86_64-linux-gnu/librtmp.so.1 (0x00007f8b24b58000)
        libssh.so.4 => /lib/x86_64-linux-gnu/libssh.so.4 (0x00007f8b24aea000)
        libpsl.so.5 => /lib/x86_64-linux-gnu/libpsl.so.5 (0x00007f8b24ad6000)
        libgssapi_krb5.so.2 => /lib/x86_64-linux-gnu/libgssapi_krb5.so.2 (0x00007f8b24a82000)
        libldap-2.5.so.0 => /lib/x86_64-linux-gnu/libldap-2.5.so.0 (0x00007f8b24a22000)
        liblber-2.5.so.0 => /lib/x86_64-linux-gnu/liblber-2.5.so.0 (0x00007f8b24a0f000)
        libzstd.so.1 => /lib/x86_64-linux-gnu/libzstd.so.1 (0x00007f8b24940000)
        libbrotlidec.so.1 => /lib/x86_64-linux-gnu/libbrotlidec.so.1 (0x00007f8b24932000)
        libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007f8b24916000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f8b2538c000)
        libunistring.so.2 => /lib/x86_64-linux-gnu/libunistring.so.2 (0x00007f8b2476c000)
        libgnutls.so.30 => /lib/x86_64-linux-gnu/libgnutls.so.30 (0x00007f8b2457f000)
        libhogweed.so.6 => /lib/x86_64-linux-gnu/libhogweed.so.6 (0x00007f8b24537000)
        libnettle.so.8 => /lib/x86_64-linux-gnu/libnettle.so.8 (0x00007f8b244f1000)
        libgmp.so.10 => /lib/x86_64-linux-gnu/libgmp.so.10 (0x00007f8b2446f000)
        libkrb5.so.3 => /lib/x86_64-linux-gnu/libkrb5.so.3 (0x00007f8b243a4000)
        libk5crypto.so.3 => /lib/x86_64-linux-gnu/libk5crypto.so.3 (0x00007f8b24375000)
        libcom_err.so.2 => /lib/x86_64-linux-gnu/libcom_err.so.2 (0x00007f8b2436d000)
        libkrb5support.so.0 => /lib/x86_64-linux-gnu/libkrb5support.so.0 (0x00007f8b2435f000)
        libsasl2.so.2 => /lib/x86_64-linux-gnu/libsasl2.so.2 (0x00007f8b24344000)
        libbrotlicommon.so.1 => /lib/x86_64-linux-gnu/libbrotlicommon.so.1 (0x00007f8b24321000)
        libp11-kit.so.0 => /lib/x86_64-linux-gnu/libp11-kit.so.0 (0x00007f8b241e6000)
        libtasn1.so.6 => /lib/x86_64-linux-gnu/libtasn1.so.6 (0x00007f8b241ce000)
        libkeyutils.so.1 => /lib/x86_64-linux-gnu/libkeyutils.so.1 (0x00007f8b241c5000)
        libresolv.so.2 => /lib/x86_64-linux-gnu/libresolv.so.2 (0x00007f8b241b1000)
        libffi.so.8 => /lib/x86_64-linux-gnu/libffi.so.8 (0x00007f8b241a4000)
```

In case a not found line appear in the output of the ldd command follow the following steps.

Carefully read the output:

- Look for these keywords to identify which one is missing:
  - libcurl.so.4 → not found (cURL is missing).
  - libssl.so or libcrypto.so → not found (OpenSSL is missing).

If no '**not found**' lines appear in the ldd libsclic.so output, all the dependencies are installed and all libraries show valid paths (for example /lib/x86_64-linux-gnu/libssl.so.3). You can move on to **RISC-V Reference Model Integration**.

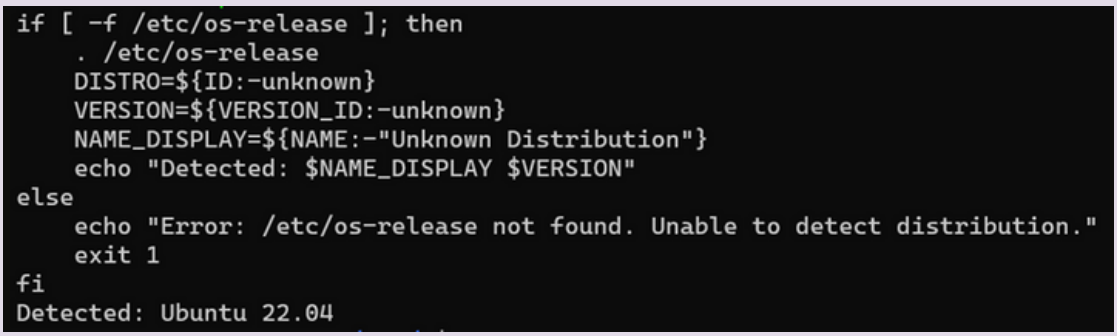
- If you see lines containing "**not found**", it means some required libraries are missing.

Follow the instructions below to install libcurl and openssl.

First of all we need to identify what linux version we are working with, the following code will print out the linux distro and version.

```
# Detect Linux distribution
if [ -f /etc/os-release ]; then
 . /etc/os-release
 DISTRO=${ID:-unknown}
 VERSION=${VERSION_ID:-unknown}
 NAME_DISPLAY=${NAME:-"Unknown Distribution"}
 echo "Detected: $NAME_DISPLAY $VERSION"
else
 echo "Error: /etc/os-release not found. Unable to detect distribution."
 exit 1
fi
```

After running in terminal above code will show “Detected: <Linux_name>”



How to check sudo access

You can check if you have sudo access by running:

```
sudo -v
```

- If it asks for your password and continues without error — you have sudo access, install using package manager.
- If it says “user is not in the sudoers file” — you don’t have sudo access, use the **Manual Installation** section.

Imparé RISC-V Reference Model User Guide

Verify and Install Build Tools

Before installing **OpenSSL** and **cURL**, make sure your system has the basic build tools installed.
Run the following command in the terminal to verify:

```
gcc --version && make --version && perl -v
```

If all three commands show a version number, your system is ready.
If any command shows "**command not found**", install the required tools using the command for your Linux distribution below:

For RHEL / Rocky Linux / AlmaLinux / CentOS

```
sudo dnf groupinstall -y "Development Tools"
sudo dnf install -y perl wget tar
sudo dnf install libpsl-devel
```

(Use yum instead of dnf for older versions, e.g. CentOS 7.)

For Ubuntu / Debian

```
sudo apt update
sudo apt install -y build-essential perl wget tar

sudo apt install libpsl-dev
```

For openSUSE

```
sudo zypper install -t pattern devel_basis
sudo zypper install -y perl wget tar
sudo zypper install libpsl-devel
```

After Installation Check

Re-run the verification command to confirm installation:

```
gcc --version && make --version && perl -v
```

You should now see version numbers such as:

```
gcc (GCC) 11.4.0
GNU Make 4.3
perl 5.32.1
```

- Exact numbers may vary depending on the latest version. You just need these installed not specific version.
- If these appear, all required build tools are successfully installed.

Install via Package Manager

If you have administrator (sudo) access, the simplest method is to install OpenSSL and cURL directly from your Linux distribution's package manager.

- For RHEL / Rocky / AlmaLinux / CentOS:

```
sudo yum install -y openssl openssl-devel curl libcurl libcurl-devel
```

- For Ubuntu / Debian:

```
sudo apt update
sudo apt install -y openssl libssl-dev curl libcurl4 libcurl4-openssl-dev
```

- For openSUSE:

```
sudo zypper install -y libopenssl3 libopenssl-devel curl libcurl-devel
```

Imparé RISC-V Reference Model User Guide

Manual Installation

If you do not have administrator privileges (for example, on shared servers or EDA environments), you can install OpenSSL and cURL manually in your home directory.

- Run these commands one by one:

```
mkdir -p $HOME/tools/{openssl,curl,src}
 cd $HOME/tools/src

 # Download and build OpenSSL
 wget https://www.openssl.org/source/openssl-3.3.2.tar.gz
 tar xzf openssl-3.3.2.tar.gz
 cd openssl-3.3.2
 ./Configure --prefix=$HOME/tools/openssl --openssldir=$HOME/tools/openssl shared
 make -j$(nproc)
 make install_sw

 # Download and build cURL
 cd $HOME/tools/src
 wget https://curl.se/download/curl-8.10.1.tar.gz
 tar xzf curl-8.10.1.tar.gz
 cd curl-8.10.1
 ./configure --prefix=$HOME/tools/curl --with-ssl=$HOME/tools/openssl --enable-shared
 make -j$(nproc)
 make install
```

After installing this manually we need to update environment variables.

1. Update Environment Variables

After installation, you need to tell your system where to find the newly installed libraries: Run the following commands in the same terminal window.

```
export PATH=$HOME/tools/curl/bin:$HOME/tools/openssl/bin:$PATH
 export
LD_LIBRARY_PATH=$HOME/tools/curl/lib:$HOME/tools/openssl/lib64:$LD_LIBRARY_PATH
```

Expected Output:

```
echo $PATH
echo $LD_LIBRARY_PATH
```

You should see the added paths in the output.

To make it permanent, run:

Permanent means your system will automatically remember these environment variable settings every time you open a new terminal or restart your system — so you don't have to export them manually again.

Run the following commands:

```
echo 'export PATH=$HOME/tools/curl/bin:$HOME/tools/openssl/bin:$PATH' >> ~/.bashrc
```

```
echo 'export
LD_LIBRARY_PATH=$HOME/tools/curl/lib:$HOME/tools/openssl/lib64:$LD_LIBRARY_PATH' >>
~/.bashrc
```

```
source ~/.bashrc
```

Verify Installation

After manually installing or using package manager use the following commands to verify that installation was successful.

Run the following commands in the terminal:

```
which curl
which openssl
curl --version
openssl version
```

Expected Output:

- which curl → /usr/bin/curl or $HOME/tools/curl/bin/curl
- which openssl → /usr/bin/openssl or $HOME/tools/openssl/bin/openssl
- curl --version → shows version info (e.g., curl 8.10.1 (OpenSSL/3.3.2))
- openssl version → displays OpenSSL version (e.g., OpenSSL 3.3.2 1 Jul 2024)

If everything is installed correctly, you will see version information for curl and OpenSSL without any errors.

Troubleshooting Common Issues

Problem: libcurl.so.4 not found

Cause: The system cannot find the cURL library path.

Fix: Update your environment variable so Linux knows where the library is.

```
export
LD_LIBRARY_PATH=$HOME/tools/curl/lib:$HOME/tools/openssl/lib64:$LD_LIBRARY_PATH
```

What is LD_LIBRARY_PATH?

It's a system variable that tells Linux where to look for shared libraries (.so files) during execution.

Problem: Undefined symbol errors

Cause: cURL and OpenSSL were built separately with mismatched versions.

Fix: Rebuild both together so cURL links with the same OpenSSL you compiled manually.

```
# Rebuild cURL using your local OpenSSL
cd $HOME/tools/src/curl-8.10.1
./configure --prefix=$HOME/tools/curl --with-ssl=$HOME/tools/openssl --enable-shared
make -j$(nproc)
make install
```

What does "Rebuild" mean?

It means cleaning and recompiling the source code to re-link libraries properly.

Final Verification

Run the following command by opening terminal in the sim folder to verify that all required libraries are linked properly:

```
ldd libsclic.so
```

All dependencies should now resolve correctly without any 'not found' messages.

```
linux-vdso.so.1 (0x00007fff68fef000)
libcrypto.so.3 => /lib/x86_64-linux-gnu/libcrypto.so.3 (0x00007f7de7c46000)
libssl.so.3 => /lib/x86_64-linux-gnu/libssl.so.3 (0x00007f7de7ba2000)
libcurl.so.4 => /lib/x86_64-linux-gnu/libcurl.so.4 (0x00007f7de7afb000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f7de78d2000)
libnghttp2.so.14 => /lib/x86_64-linux-gnu/libnghttp2.so.14 (0x00007f7de78a8000)
libidn2.so.0 => /lib/x86_64-linux-gnu/libidn2.so.0 (0x00007f7de7885000)
librtmp.so.1 => /lib/x86_64-linux-gnu/librtmp.so.1 (0x00007f7de7866000)
libssh.so.4 => /lib/x86_64-linux-gnu/libssh.so.4 (0x00007f7de77f8000)
libpsl.so.5 => /lib/x86_64-linux-gnu/libpsl.so.5 (0x00007f7de77e4000)
libgssapi_krb5.so.2 => /lib/x86_64-linux-gnu/libgssapi_krb5.so.2 (0x00007f7de7790000)
libldap-2.5.so.0 => /lib/x86_64-linux-gnu/libldap-2.5.so.0 (0x00007f7de7730000)
liblber-2.5.so.0 => /lib/x86_64-linux-gnu/liblber-2.5.so.0 (0x00007f7de771d000)
libzstd.so.1 => /lib/x86_64-linux-gnu/libzstd.so.1 (0x00007f7de764e000)
libbrotlidec.so.1 => /lib/x86_64-linux-gnu/libbrotlidec.so.1 (0x00007f7de7640000)
libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007f7de7624000)
/lib64/ld-linux-x86-64.so.2 (0x00007f7de809a000)
libunistring.so.2 => /lib/x86_64-linux-gnu/libunistring.so.2 (0x00007f7de747a000)
libgnutls.so.30 => /lib/x86_64-linux-gnu/libgnutls.so.30 (0x00007f7de728d000)
libhogweed.so.6 => /lib/x86_64-linux-gnu/libhogweed.so.6 (0x00007f7de7245000)
libnettle.so.8 => /lib/x86_64-linux-gnu/libnettle.so.8 (0x00007f7de71ff000)
libgmp.so.10 => /lib/x86_64-linux-gnu/libgmp.so.10 (0x00007f7de717d000)
libkrb5.so.3 => /lib/x86_64-linux-gnu/libkrb5.so.3 (0x00007f7de70b2000)
libk5crypto.so.3 => /lib/x86_64-linux-gnu/libk5crypto.so.3 (0x00007f7de7083000)
libcom_err.so.2 => /lib/x86_64-linux-gnu/libcom_err.so.2 (0x00007f7de707b000)
libkrb5support.so.0 => /lib/x86_64-linux-gnu/libkrb5support.so.0 (0x00007f7de706d000)
libsasl2.so.2 => /lib/x86_64-linux-gnu/libsasl2.so.2 (0x00007f7de7052000)
libbrotlicommon.so.1 => /lib/x86_64-linux-gnu/libbrotlicommon.so.1 (0x00007f7de702f000)
libp11-kit.so.0 => /lib/x86_64-linux-gnu/libp11-kit.so.0 (0x00007f7de6ef4000)
libtasn1.so.6 => /lib/x86_64-linux-gnu/libtasn1.so.6 (0x00007f7de6edc000)
libkeyutils.so.1 => /lib/x86_64-linux-gnu/libkeyutils.so.1 (0x00007f7de6ed3000)
libresolv.so.2 => /lib/x86_64-linux-gnu/libresolv.so.2 (0x00007f7de6ebf000)
libffi.so.8 => /lib/x86_64-linux-gnu/libffi.so.8 (0x00007f7de6eb2000)
```

Output may vary based on the linux version. But there should be no “not found” text in front of any dependencies. You can check this automatically using the following command or just look manually at the above output.

Imparé RISC-V Reference Model User Guide

For following automated checking copy paste whole following command in one go into the terminal.

```
if ldd libsclic.so | grep -q "not found"; then
echo "FAILED: Missing dependencies found\!";
ldd libsclic.so | grep "not found";
else
echo "SUCCESS: All dependencies resolved\!";
echo "";
echo "Library summary:";
ldd libsclic.so | wc -l;
echo "libraries linked";
fi
```

If above automated checking if script just look manually at the output of the ldd libsclic.so that no “not found” text is found.

In case you have followed the above instructions not found text would not appear.

All dependencies are installed, and you are ready to proceed to **RISC-V Reference Model Integration**.