---

# Troubleshooting Guide: Building OpenROAD-flow-scripts on a Low-Memory VM

## UBUNTU 24.04

By: Akhilesh Kumar P
Github: https://github.com/akhileshkumarp

This guide will help you resolve a cascade of common build failures, from initial setup to running the flow.

---

### Phase 1: Get a Clean Copy of the Code

You will face many errors if you copy the files from Windows or have an unstable network connection. Start by cloning the repository *inside your VM*.

#### Error 1: git clone fails with fatal: early EOF or curl 56

- **Cause:** The repository is very large (over 4GB with all history). Your VM's network connection is dropping before the download completes.
- **Solution:** Use a "shallow clone" (--depth 1) to download only the latest version, which is much smaller.

```Bash
# Remove any old, broken folder first
rm -rf OpenROAD-flow-scripts

# Use a shallow, recursive clone
git clone --recursive --depth 1 https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts.git
cd OpenROAD-flow-scripts
```

### Error 2: ./setup.sh: command not found or /usr/bin/env: ‘bash\r’: No such file or directory

- **Cause:** This happens if you copied the files from Windows. Windows uses different file permissions and line endings (\r\n) than Linux.
- **Solution:** Do not copy from Windows. Use the git clone command (above) inside your Linux VM to get files with the correct format.

---

## Phase 2: System Setup (setup.sh)

Before you can build, you must install the system dependencies.

### Step 1: Increase Virtual Memory (Swap)

- **Cause:** Your VM (e.g., 4-6GB RAM) does not have enough memory to compile. Adding swap space uses your disk as "slow RAM" and prevents crashes.
- **Solution:** Create and enable an 8GB swap file.
  Bash cmd:
  ```
  sudo fallocate -l 8G /swapfile
  sudo chmod 600 /swapfile
  sudo mkswap /swapfile
  sudo swapon /swapfile
  ```

### Step 2: Run the Setup Script

- **Command:**
  Bash
  ```
  sudo ./setup.sh
  ```

- **Interactive Prompt:** You may be stopped by rm: remove write-protected regular file ...?
- **Solution:** This is normal. The script is cleaning up temporary files. Type **y** and press **Enter** for each file it asks about.

---

## Phase 3: Building the Tools (build_openroad.sh)

This is the most common place to fail.

### Error 1: g++: fatal error: Killed signal terminated program cc1plus

- **Cause: This is the main error.** Your system ran out of memory. By default, the build uses multiple processor cores (-j4 or more), which consumes a lot of RAM. The Linux "OOM Killer" terminates the compiler to save your VM from crashing.
- **Solution:** Force the build to use **only one thread** with the --threads 1 flag.
  Bash
  ```
  ./build_openroad.sh --local --threads 1
  ```

  *Note: This will be very slow (over an hour), but it is the only way to succeed on a low-RAM machine.*

### Error 2: ls: .../tools/yosys/yosys: No such file or directory

- **Cause:** The build script (build_openroad.sh) failed to build Yosys, even though it continued and built OpenROAD. This can be caused by a subtle dependency issue or a corrupted state.
- **Solution:** Manually build Yosys.
  Bash
  ```
  cd tools/yosys
  make -j1
  cd ../..
  ```

### Error 3: undefined reference to 'testing::internal::MakeAndRegisterTestInfo(...)'

- **Cause:** The build succeeded in creating the main openroad executable but failed at 97% while trying to link the optional unit tests. This is a linker error, often from a corrupted Google Test (gtest) dependency.
- **Solution: Ignore this error.** The main tools (openroad and yosys) are already built. You can proceed.

---

## Phase 4: Running the Flow (make)

After a successful build, you will get errors when you try to run the flow.

### Error: make: *** [Makefile:...] Error 127 or .../openroad: No such file or directory

- **Cause:** You built with --local, which places the executables in a local folder, not in your system PATH. The Makefile is looking for openroad and yosys in the default system locations and can't find them.
- **Solution:** You must **manually provide the paths** to make every time you run it.

**Example 1: Running the gcd flow:**

Bash

```
# Navigate to the flow directory
cd ~/OpenROAD-flow-scripts/flow

# Run make, providing the paths to both executables
YOSYS_EXE=~/OpenROAD-flow-scripts/tools/yosys/yosys \
OPENROAD_EXE=~/OpenROAD-flow-scripts/tools/OpenROAD/build/bin/openroad \
make
```

**Example 2: Opening the GUI:**

Bash

```
# Navigate to the flow directory
cd ~/OpenROAD-flow-scripts/flow

# Run the GUI, providing the path to the openroad executable
OPENROAD_EXE=~/OpenROAD-flow-scripts/tools/OpenROAD/build/bin/openroad make gui_final
```