AIMS暑期实习_Gemmini+MLIR

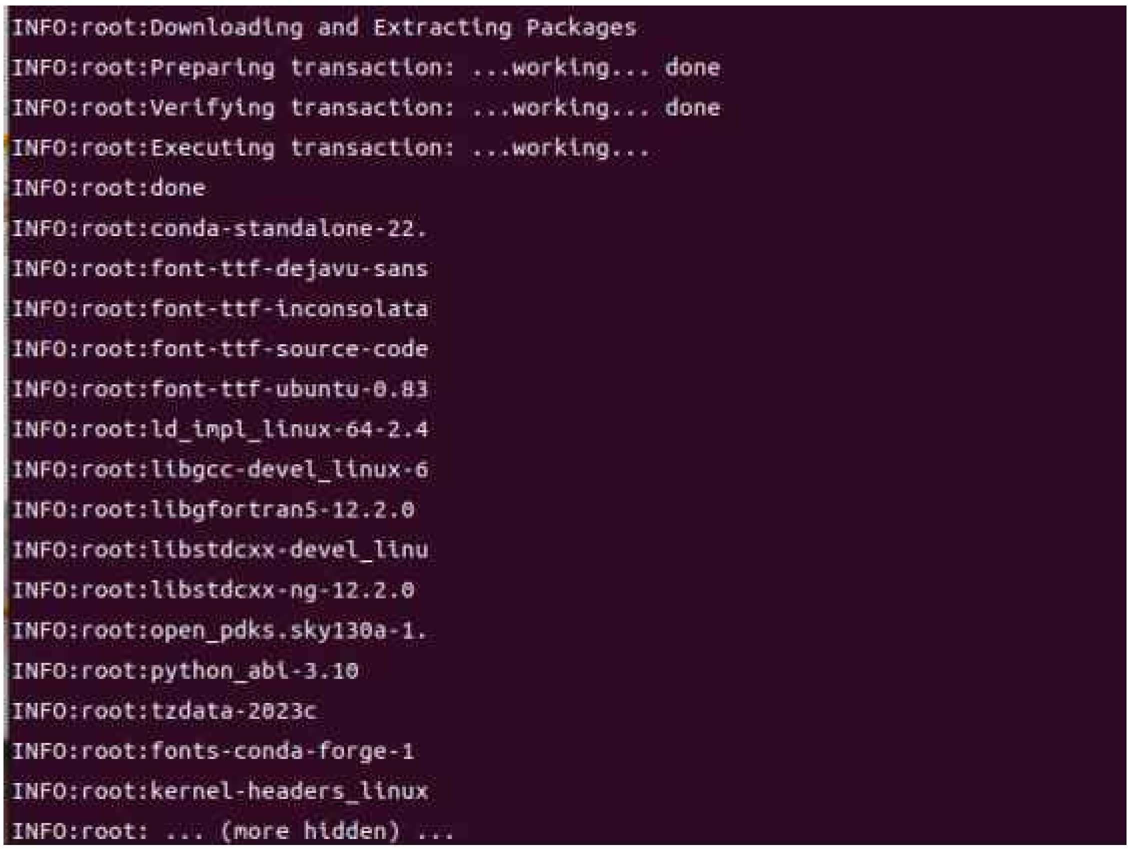
1.环境配置

一、Gemmini环境配置

1.配置Chipyard环境：

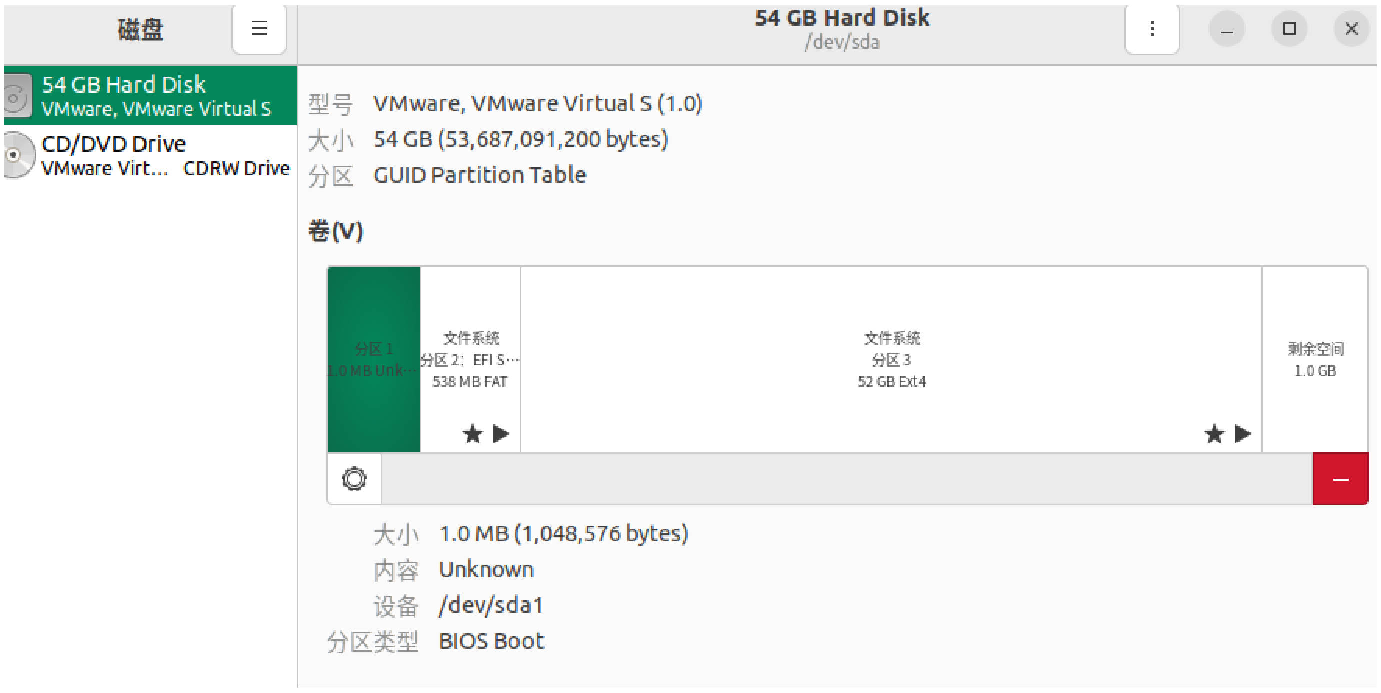
根据安装提示，必须在linux中进行。第一次没注意，使用windows结果卡在“Collecting package metadata”。

由于conda和git之前已经安装过，现在只要安装conda-lock。执行“./build-setup.sh riscv-tools”指令后会有漫长的等待。



```
INFO:root:Downloading and Extracting Packages
INFO:root:Preparing transaction: ...working... done
INFO:root:Verifying transaction: ...working... done
INFO:root:Executing transaction: ...working...
INFO:root:done
INFO:root:conda-standalone-22.
INFO:root:font-ttf-dejavu-sans
INFO:root:font-ttf-inconsolata
INFO:root:font-ttf-source-code
INFO:root:font-ttf-ubuntu-0.83
INFO:root:ld_impl_linux-64-2.4
INFO:root:libgcc-devel_linux-6
INFO:root:libgfortran5-12.2.0
INFO:root:libstdcxx-devel_linu
INFO:root:libstdcxx-ng-12.2.0
INFO:root:open_pdks.sky130a-1.
INFO:root:python_abi-3.10
INFO:root:tzdata-2023c
INFO:root:fonts-conda-forge-1
INFO:root:kernel-headers_linux
INFO:root: ... (more hidden) ...
```

中途虚拟机内存不足了，不得不再来一次。直接在VMware设置扩容是不够的，还要在虚拟机中设置文件分区大小。

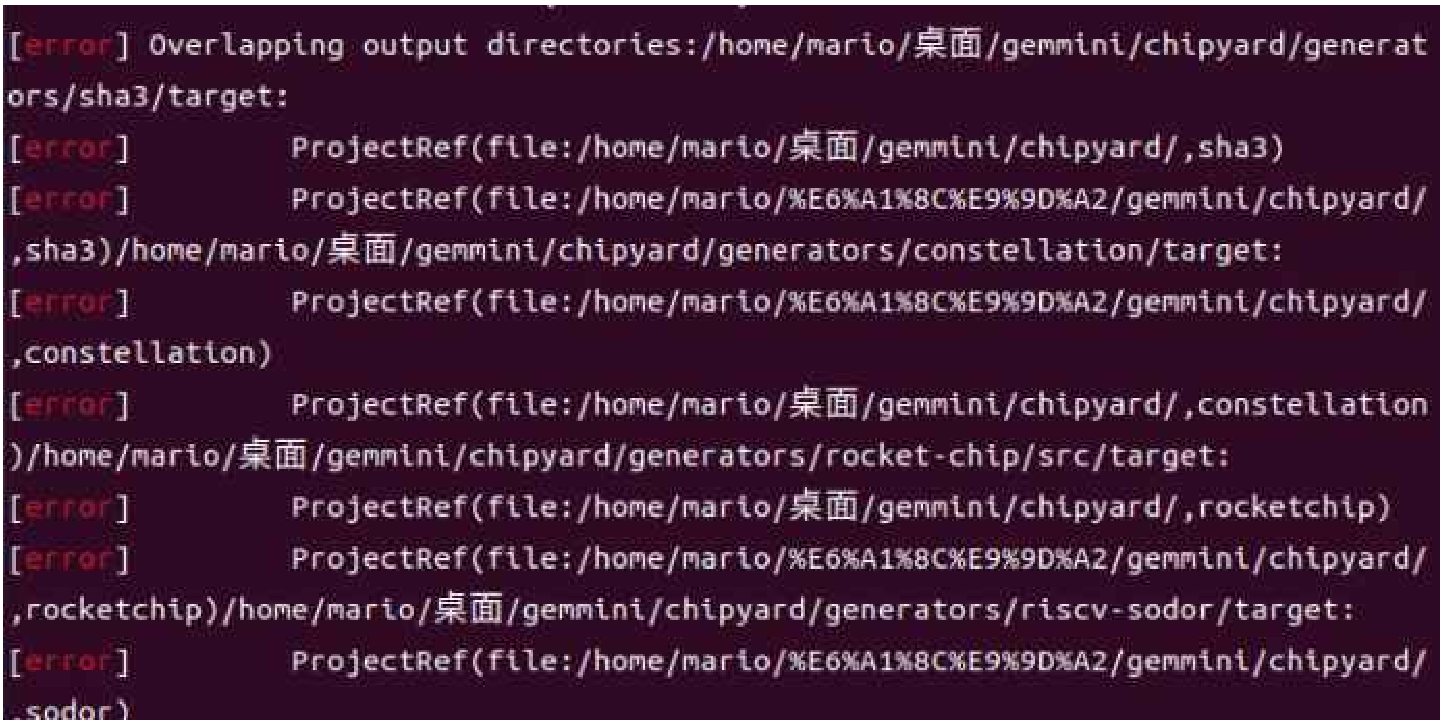


型号 VMware, VMware Virtual S (1.0)
大小 54 GB (53,687,091,200 bytes)
分区 GUID Partition Table

卷(V)

大小 1.0 MB (1,048,576 bytes)
内容 Unknown
设备 /dev/sda1
分区类型 BIOS Boot

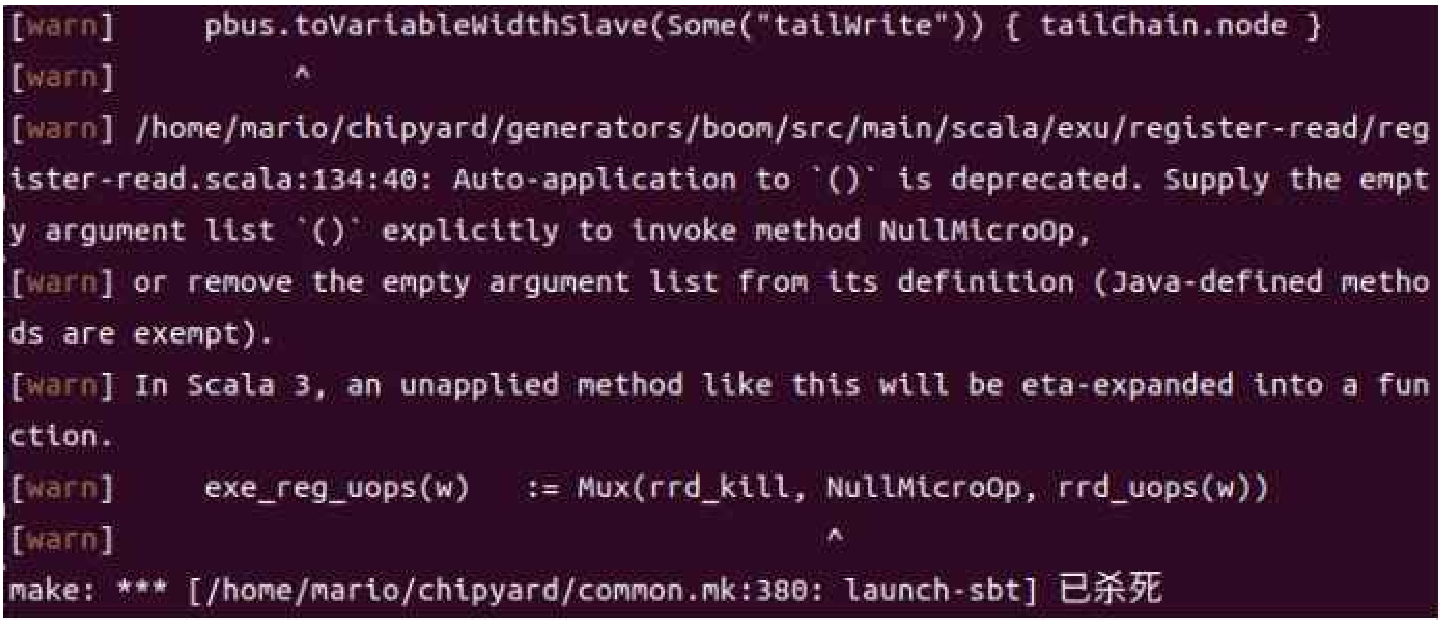
报错：./build-setup.sh riscv-tools运行时报错重叠的输出路径



```
[error] Overlapping output directories:/home/mario/桌面/gemmini/chipyard/generat
ors/sha3/target:
[error]         ProjectRef(file:/home/mario/桌面/gemmini/chipyard/,sha3)
[error]         ProjectRef(file:/home/mario/%E6%A1%8C%E9%9D%A2/gemmini/chipyard/
,sha3)/home/mario/桌面/gemmini/chipyard/generators/constellation/target:
[error]         ProjectRef(file:/home/mario/%E6%A1%8C%E9%9D%A2/gemmini/chipyard/
,constellation)
[error]         ProjectRef(file:/home/mario/桌面/gemmini/chipyard/,constellation
)/home/mario/桌面/gemmini/chipyard/generators/rocket-chip/src/target:
[error]         ProjectRef(file:/home/mario/桌面/gemmini/chipyard/,rocketchip)
[error]         ProjectRef(file:/home/mario/%E6%A1%8C%E9%9D%A2/gemmini/chipyard/
,rocketchip)/home/mario/桌面/gemmini/chipyard/generators/riscv-sodor/target:
[error]         ProjectRef(file:/home/mario/%E6%A1%8C%E9%9D%A2/gemmini/chipyard/
,sodor)
```

重新安装无效，改chipyard版本无效，若暂且搁置后面能运行到第4步。

后来尝试在根目录下安装，结果出现一大堆警告，最后在common.mk的380行强行终止。

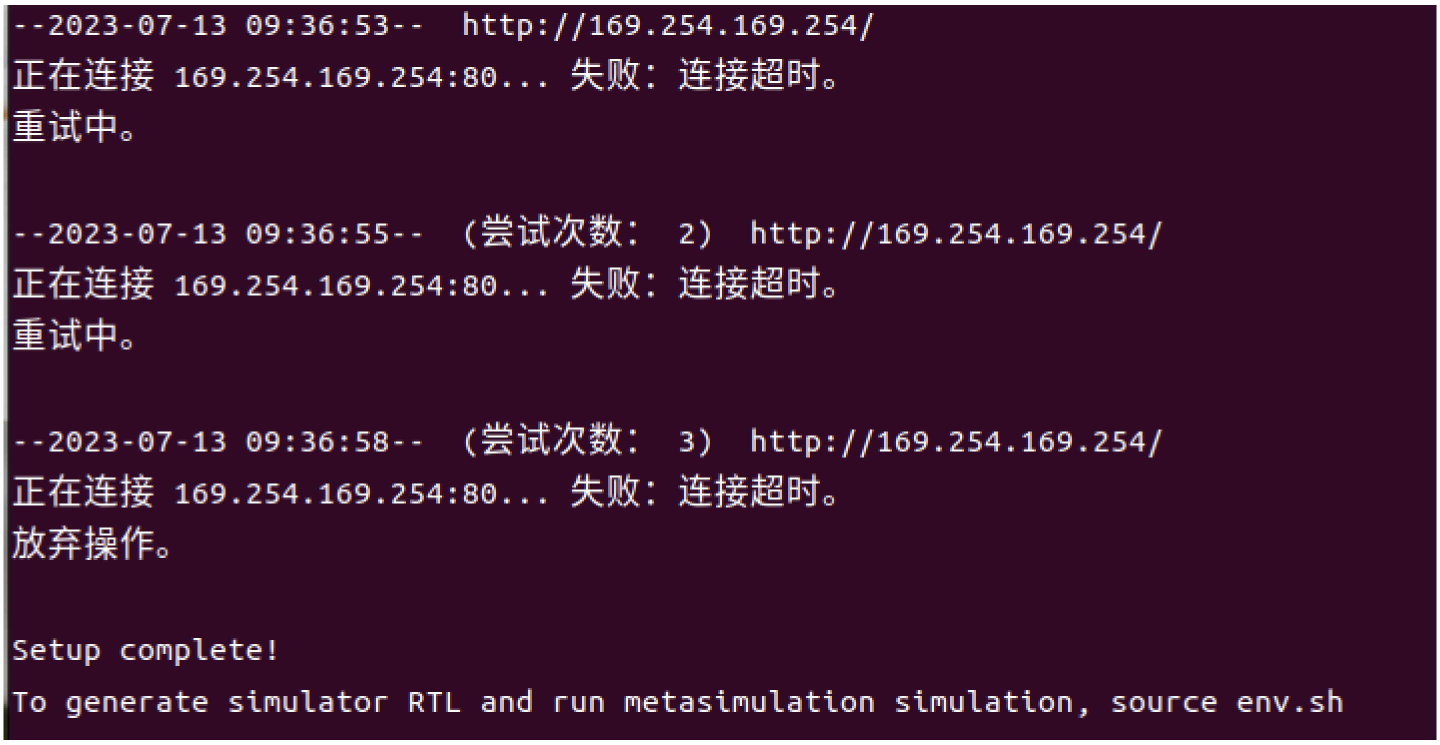


```
[warn]     pbus.toVariableWidthSlave(Some("tailWrite")) { tailChain.node }
[warn]          ^
[warn] /home/mario/chipyard/generators/boom/src/main/scala/exu/register-read/reg
ister-read.scala:134:40: Auto-application to `()` is deprecated. Supply the empt
y argument list `()` explicitly to invoke method NullMicroOp,
[warn] or remove the empty argument list from its definition (Java-defined metho
ds are exempt).
[warn] In Scala 3, an unapplied method like this will be eta-expanded into a fun
ction.
[warn]     exe_reg_uops(w)   := Mux(rrd_kill, NullMicroOp, rrd_uops(w))
[warn]                                      ^
make: *** [/home/mario/chipyard/common.mk:380: launch-sbt] 已杀死
```

此时再继续会导致“make -C software/libgemmini install”指令不通过。

解决方法：多运行几次init-submodules-no-riscv-tools.sh 和 build-toolchain-extra.sh 两个脚本后再次尝试，发现正常了。

其中发现有因为网络问题无法连接的情况，科学 上网也没用。



```
--2023-07-13 09:36:53--  http://169.254.169.254/
正在连接 169.254.169.254:80... 失败：连接超时。
重试中。

--2023-07-13 09:36:55--  (尝试次数： 2)  http://169.254.169.254/
正在连接 169.254.169.254:80... 失败：连接超时。
重试中。

--2023-07-13 09:36:58--  (尝试次数： 3)  http://169.254.169.254/
正在连接 169.254.169.254:80... 失败：连接超时。
放弃操作。

Setup complete!
To generate simulator RTL and run metasimulation simulation, source env.sh
```

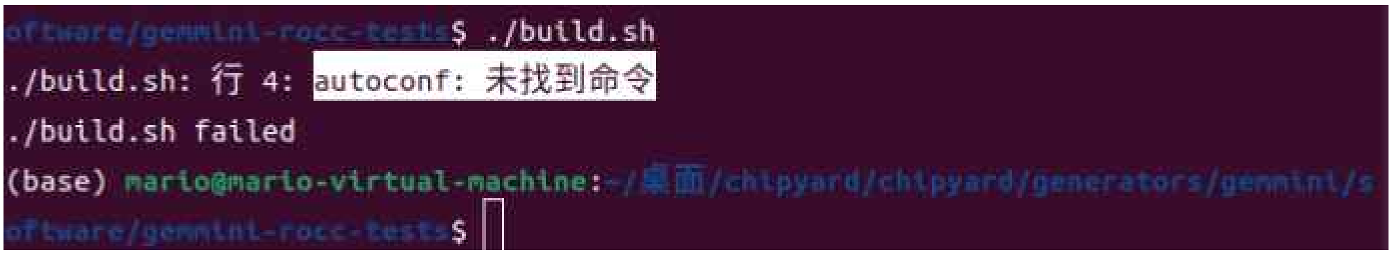
不过好像不影响。

2.Setting Up Gemmini

这一步很顺利，按流程即可。

3.Building Gemmini Software

按照流程执行到“./build.sh”时，**报错“autoconf: 未找到命令”**

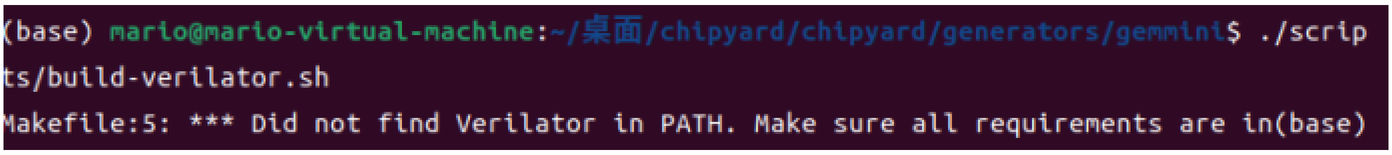


```
oftware/gemmini-rocc-tests$ ./build.sh
./build.sh: 行 4: autoconf: 未找到命令
./build.sh failed
(base) mario@mario-virtual-machine:~/桌面/chipyard/chipyard/generators/gemmini/s
oftware/gemmini-rocc-tests$ 
```

解决方法：安装缺少的依赖，输入“sudo apt install -y autoconf”

4.Building Gemmini Hardware and Cycle-Accurate Simulators

报错：环境变量PATH找不到verilator

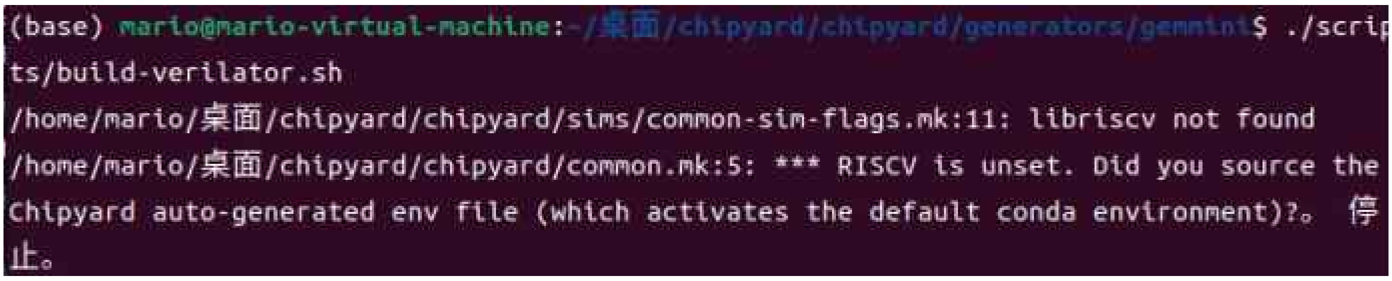


```
(base) mario@mario-virtual-machine:~/桌面/chipyard/chipyard/generators/gemmini$ ./scrip
ts/build-verilator.sh
Makefile:5: *** Did not find Verilator in PATH. Make sure all requirements are in(base)
```

解决方法：sudo apt-get install verilator

update：改conda环境

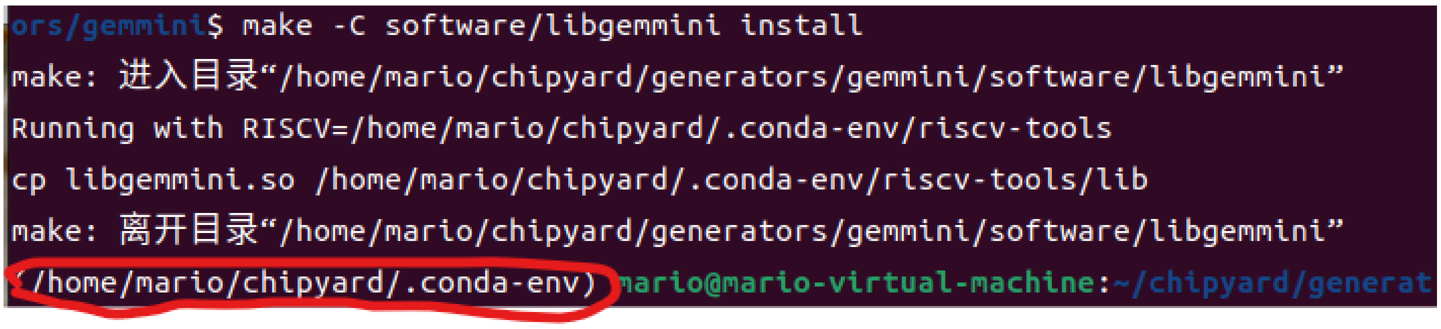
报错：找不到libriscv



```
(base) mario@mario-virtual-machine:~/桌面/chipyard/chipyard/generators/gemmini$ ./scrip
ts/build-verilator.sh
/home/mario/桌面/chipyard/chipyard/sims/common-sim-flags.mk:11: libriscv not found
/home/mario/桌面/chipyard/chipyard/common.mk:5: *** RISCV is unset. Did you source the
Chipyard auto-generated env file (which activates the default conda environment)?。 停
止。
```

原因：可能是之前报错的指令“./build-setup.sh riscv-tools”导致，缺少某些包。重新加载env.sh无效

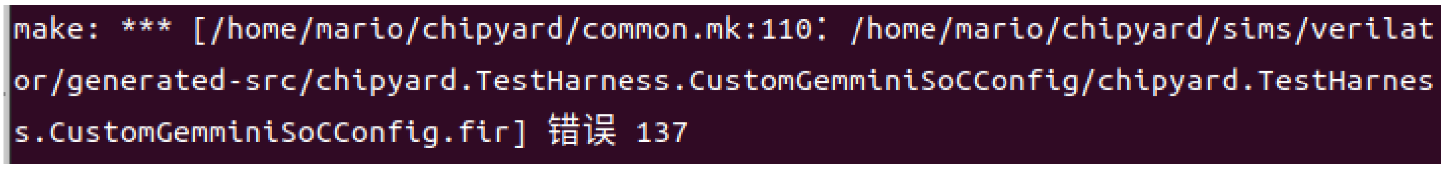
解决方法：发现并不是上述原因，是没有改conda环境，应该改成如下所示，之后的操作必须改conda！



```
ors/gemmini$ make -C software/libgemmini install
make: 进入目录“/home/mario/chipyard/generators/gemmini/software/libgemmini”
Running with RISCV=/home/mario/chipyard/.conda-env/riscv-tools
cp libgemmini.so /home/mario/chipyard/.conda-env/riscv-tools/lib
make: 离开目录“/home/mario/chipyard/generators/gemmini/software/libgemmini”
(/home/mario/chipyard/.conda-env) mario@mario-virtual-machine:~/chipyard/generat
```

5.Building Gemmini Functional Simulators

报错：make错误

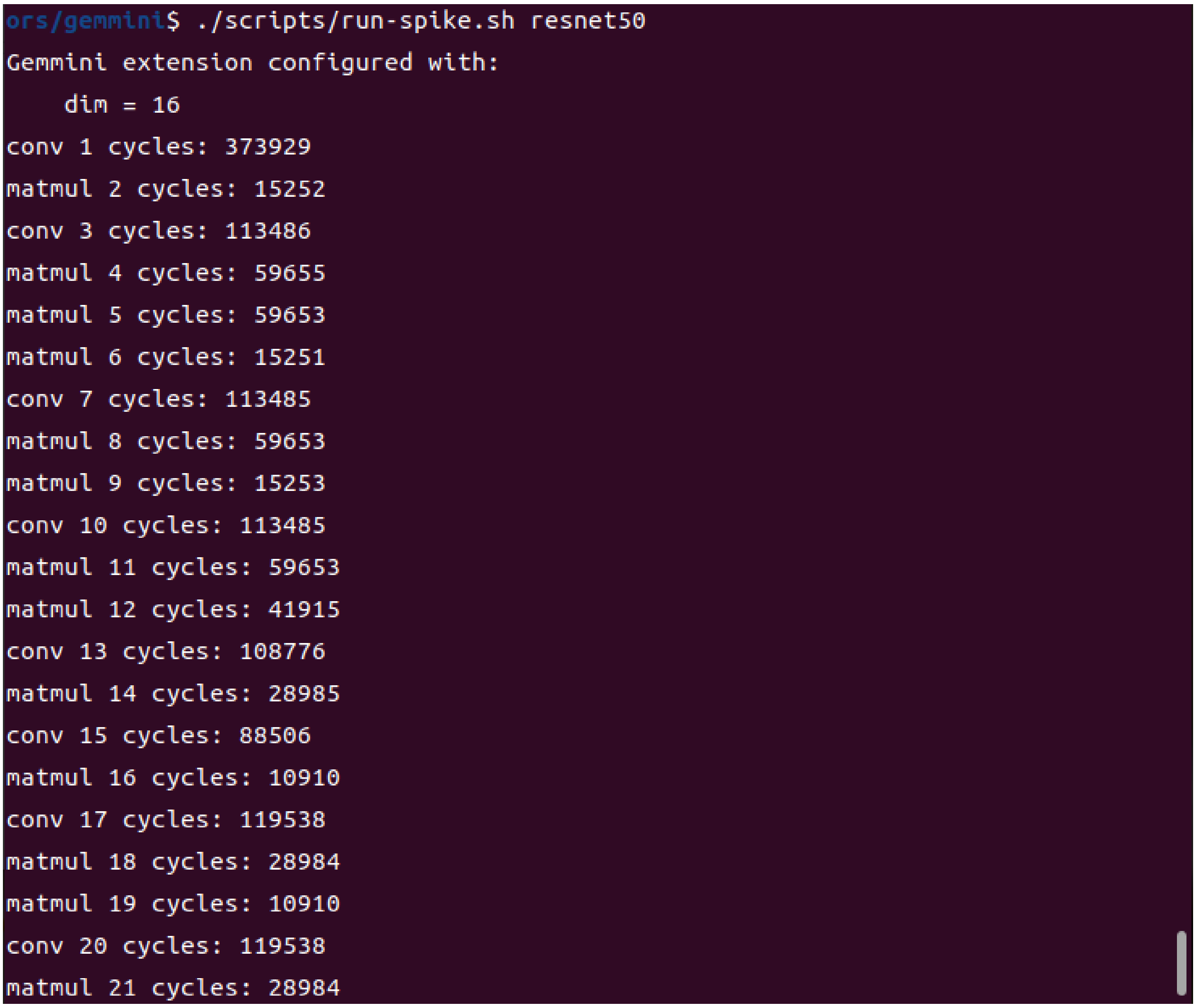


```
make: *** [/home/mario/chipyard/common.mk:110: /home/mario/chipyard/sims/verilat
or/generated-src/chipyard.TestHarness.CustomGemminiSoCConfig/chipyard.TestHarnes
s.CustomGemminiSoCConfig.fir] 错误 137
```

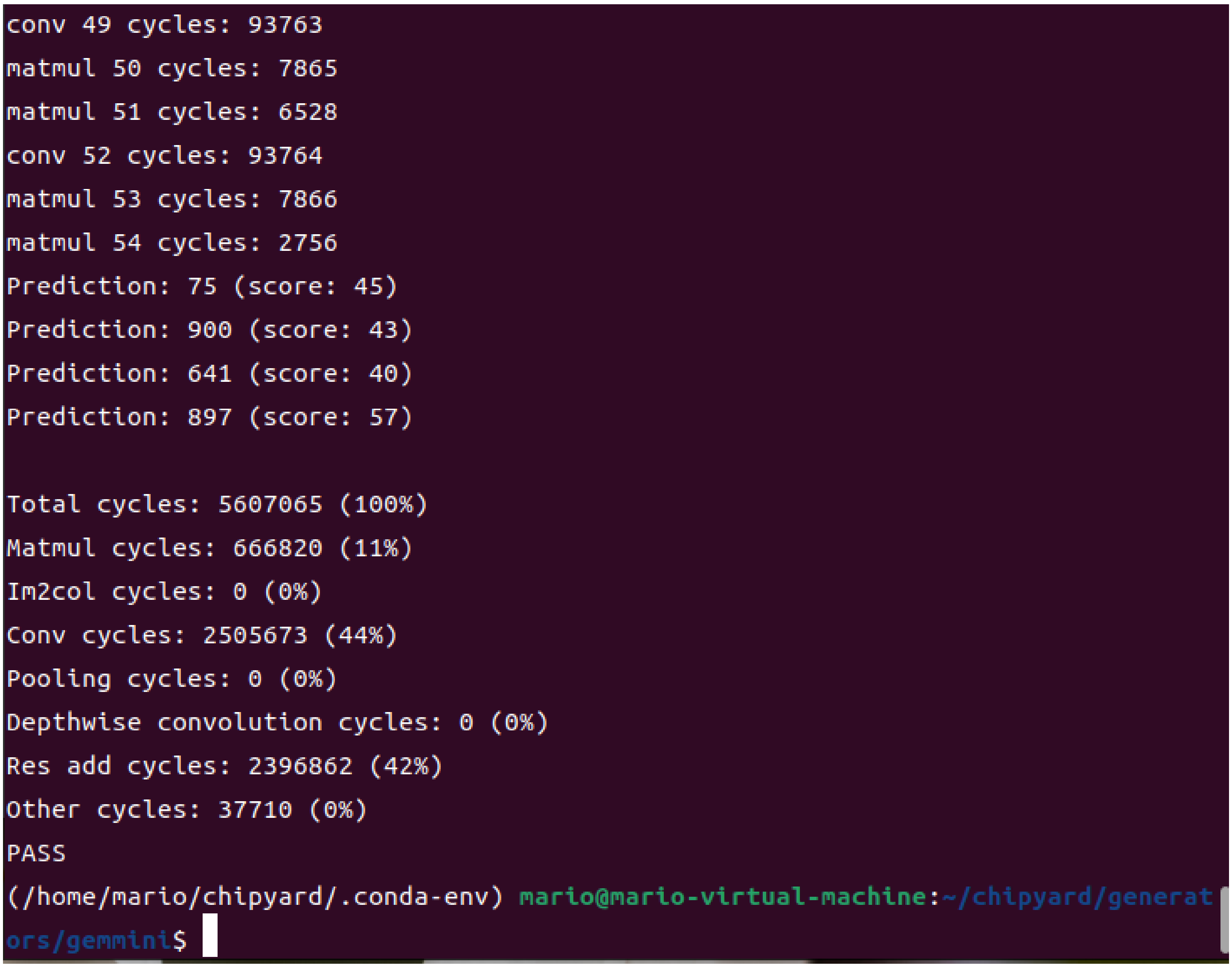
会影响6(2)操作，若6(2)报错找不到上图文件，4、5可以多执行几次。注意conda环境。

6.Run Simulators

（1）Run a large DNN workload in the functional simulator


```
ors/gemmini$ ./scripts/run-spike.sh resnet50
Gemmini extension configured with:
    dim = 16
conv 1 cycles: 373929
matmul 2 cycles: 15252
conv 3 cycles: 113486
matmul 4 cycles: 59655
matmul 5 cycles: 59653
matmul 6 cycles: 15251
conv 7 cycles: 113485
matmul 8 cycles: 59653
matmul 9 cycles: 15253
conv 10 cycles: 113485
matmul 11 cycles: 59653
matmul 12 cycles: 41915
conv 13 cycles: 108776
matmul 14 cycles: 28985
conv 15 cycles: 88506
matmul 16 cycles: 10910
conv 17 cycles: 119538
matmul 18 cycles: 28984
matmul 19 cycles: 10910
conv 20 cycles: 119538
matmul 21 cycles: 28984
```

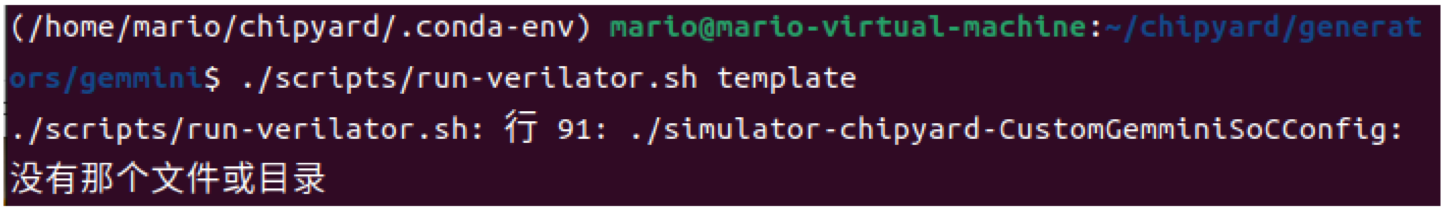
运行结果：



```
conv 49 cycles: 93763
matmul 50 cycles: 7865
matmul 51 cycles: 6528
conv 52 cycles: 93764
matmul 53 cycles: 7866
matmul 54 cycles: 2756
Prediction: 75 (score: 45)
Prediction: 900 (score: 43)
Prediction: 641 (score: 40)
Prediction: 897 (score: 57)

Total cycles: 5607065 (100%)
Matmul cycles: 666820 (11%)
Im2col cycles: 0 (0%)
Conv cycles: 2505673 (44%)
Pooling cycles: 0 (0%)
Depthwise convolution cycles: 0 (0%)
Res add cycles: 2396862 (42%)
Other cycles: 37710 (0%)
PASS
(/home/mario/chipyard/.conda-env) mario@mario-virtual-machine:~/chipyard/generat
ors/gemmini$
```

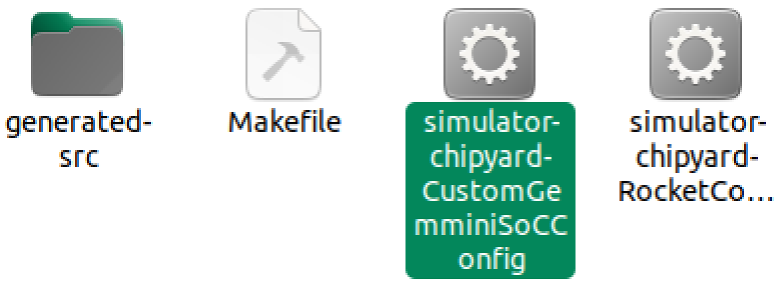
（2）Run a smaller workload in baremetal mode, on a cycle-accurate simulator

报错：找不到文件

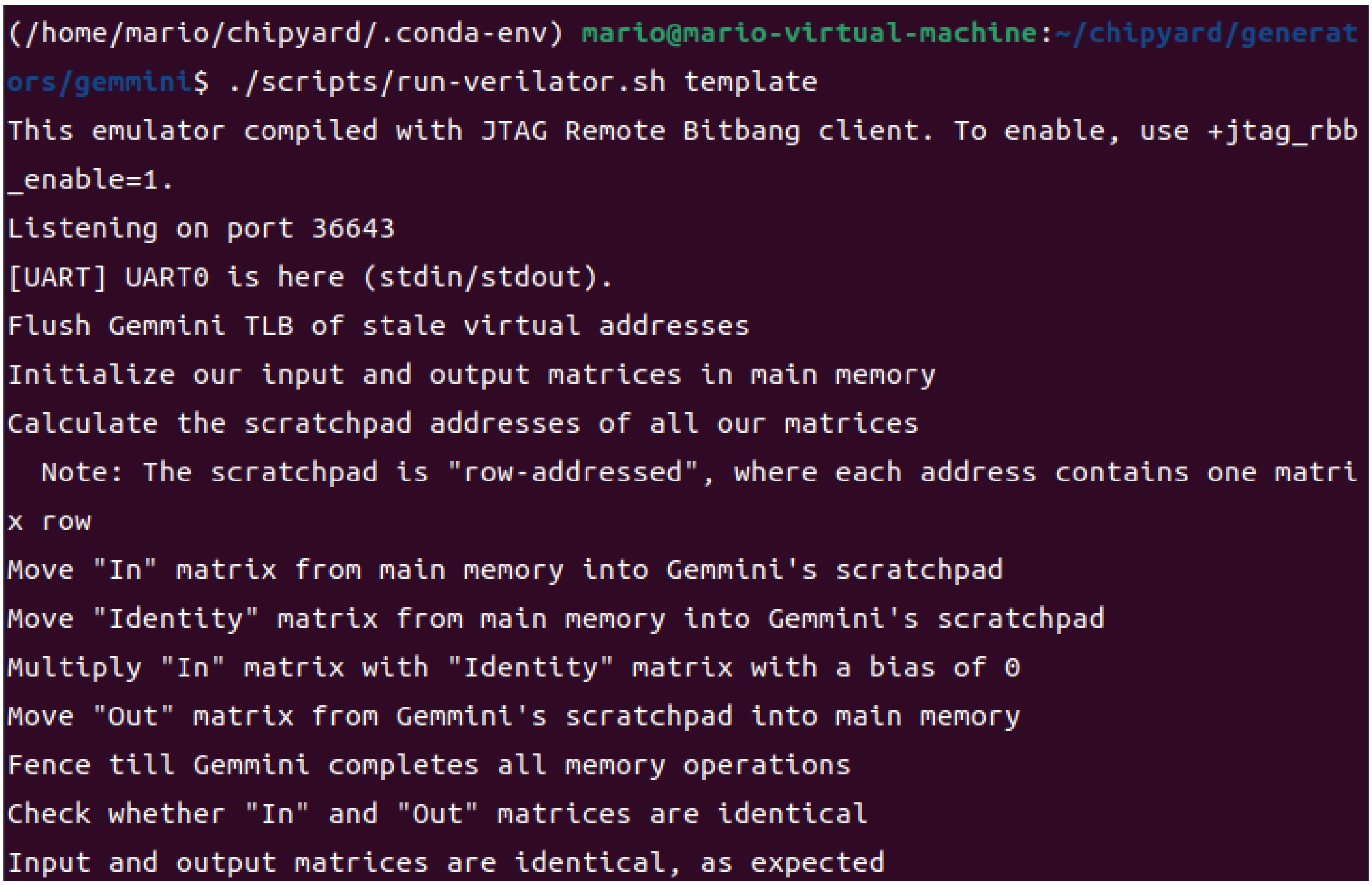


```
(/home/mario/chipyard/.conda-env) mario@mario-virtual-machine:~/chipyard/generat
ors/gemmini$ ./scripts/run-verilator.sh template
./scripts/run-verilator.sh: 行 91: ./simulator-chipyard-CustomGemminiSoCConfig:
没有那个文件或目录
```

解决方法：重复4、5中的操作，多make几次，会出现下面这个文件


generated-src
Makefile
simulator-chipyard-CustomGemminiSoCConfig
simulator-chipyard-RocketCo...

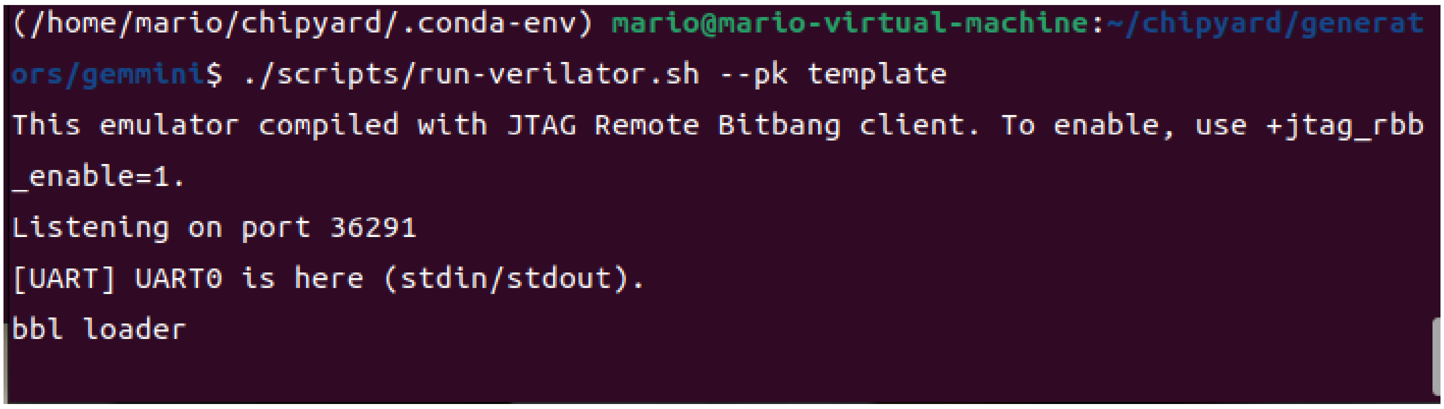
运行结果：



```
(/home/mario/chipyard/.conda-env) mario@mario-virtual-machine:~/chipyard/generat
ors/gemmini$ ./scripts/run-verilator.sh template
This emulator compiled with JTAG Remote Bitbang client. To enable, use +jtag_rbb
_enable=1.
Listening on port 36643
[UART] UART0 is here (stdin/stdout).
Flush Gemmini TLB of stale virtual addresses
Initialize our input and output matrices in main memory
Calculate the scratchpad addresses of all our matrices
  Note: The scratchpad is "row-addressed", where each address contains one matri
x row
Move "In" matrix from main memory into Gemmini's scratchpad
Move "Identity" matrix from main memory into Gemmini's scratchpad
Multiply "In" matrix with "Identity" matrix with a bias of 0
Move "Out" matrix from Gemmini's scratchpad into main memory
Fence till Gemmini completes all memory operations
Check whether "In" and "Out" matrices are identical
Input and output matrices are identical, as expected
```

（3）Run a smaller workload with the proxy-kernel, on a cycle accurate simulator

运行结果：

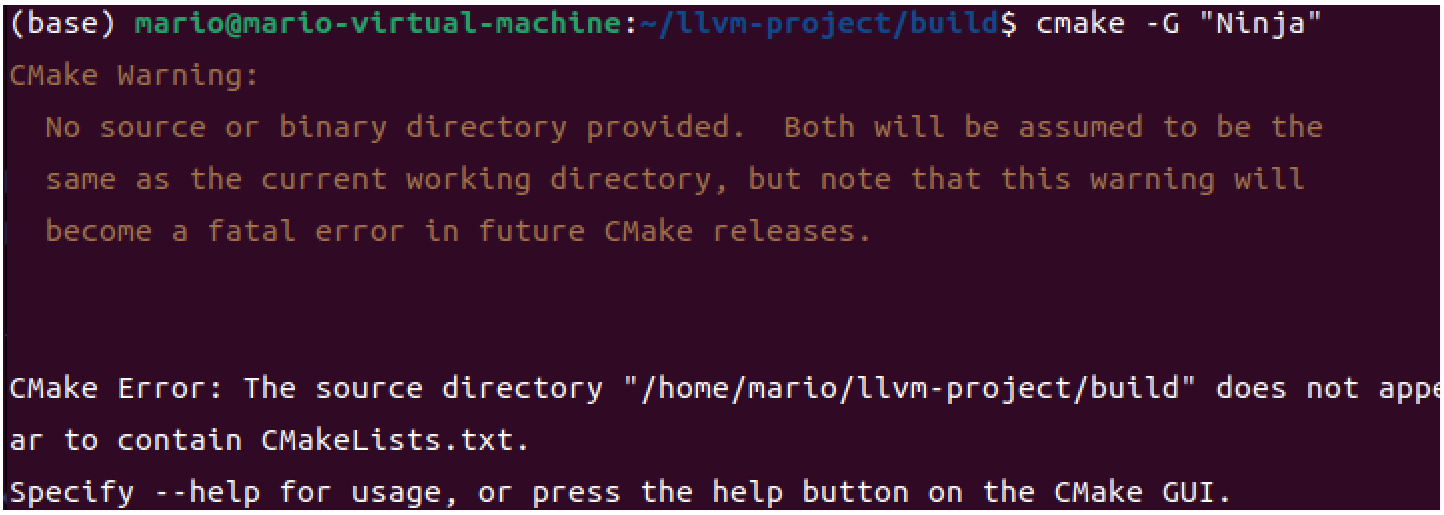


```
(/home/mario/chipyard/.conda-env) mario@mario-virtual-machine:~/chipyard/generat
ors/gemmini$ ./scripts/run-verilator.sh --pk template
This emulator compiled with JTAG Remote Bitbang client. To enable, use +jtag_rbb
_enable=1.
Listening on port 36291
[UART] UART0 is here (stdin/stdout).
bbl loader
```

二、buddy-mlir环境配置

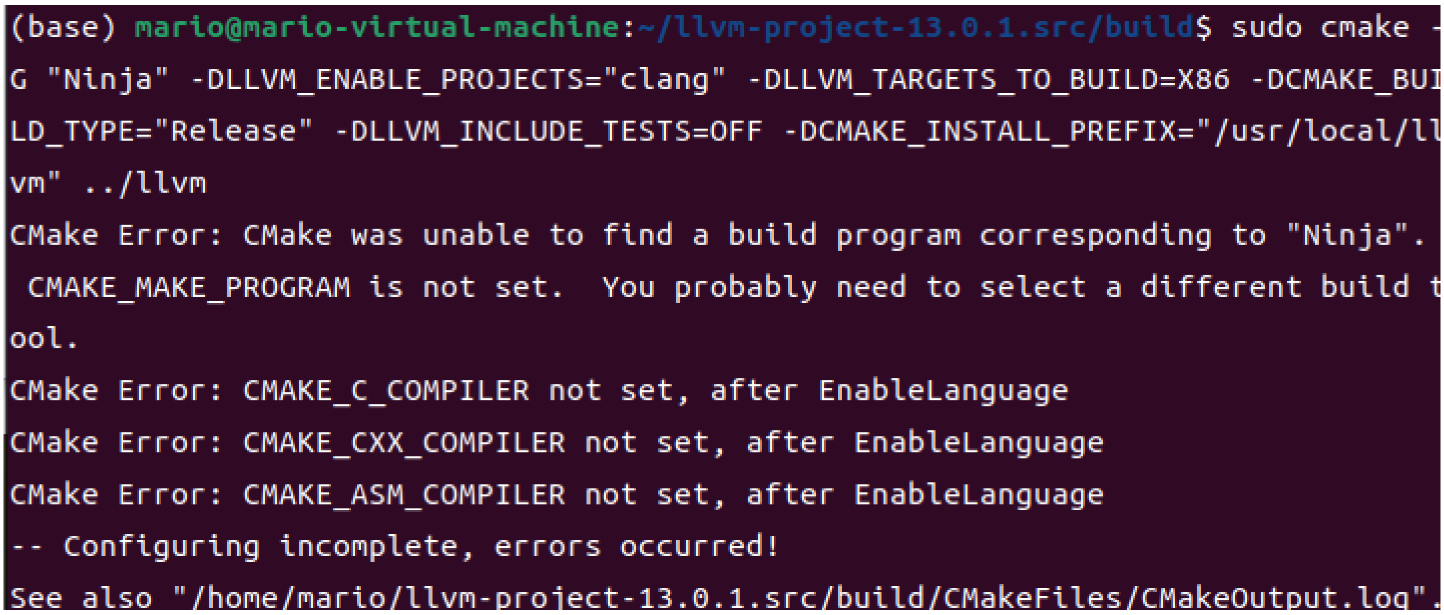
1.LLVM/MLIR Dependencies

安装llvm时报错：cmake失败



```
(base) mario@mario-virtual-machine:~/llvm-project/build$ cmake -G "Ninja"
CMake Warning:
  No source or binary directory provided.  Both will be assumed to be the
  same as the current working directory, but note that this warning will
  become a fatal error in future CMake releases.


CMake Error: The source directory "/home/mario/llvm-project/build" does not appe
ar to contain CMakeLists.txt.
Specify --help for usage, or press the help button on the CMake GUI.
```

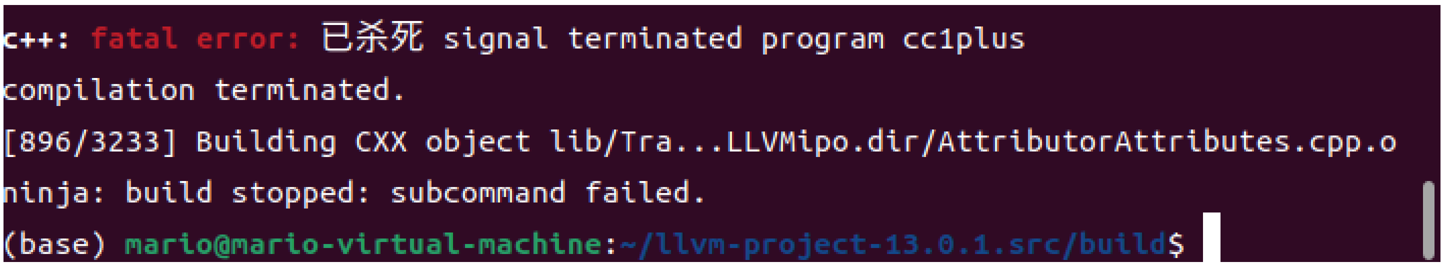
想办法重新装llvm，结果又**报错：用不了Ninja**



```
(base) mario@mario-virtual-machine:~/llvm-project-13.0.1.src/build$ sudo cmake -
G "Ninja" -DLLVM_ENABLE_PROJECTS="clang" -DLLVM_TARGETS_TO_BUILD=X86 -DCMAKE_BUI
LD_TYPE="Release" -DLLVM_INCLUDE_TESTS=OFF -DCMAKE_INSTALL_PREFIX="/usr/local/ll
vm" ../llvm
CMake Error: CMake was unable to find a build program corresponding to "Ninja".
 CMAKE_MAKE_PROGRAM is not set.  You probably need to select a different build t
ool.
CMake Error: CMAKE_C_COMPILER not set, after EnableLanguage
CMake Error: CMAKE_CXX_COMPILER not set, after EnableLanguage
CMake Error: CMAKE_ASM_COMPILER not set, after EnableLanguage
-- Configuring incomplete, errors occurred!
See also "/home/mario/llvm-project-13.0.1.src/build/CMakeFiles/CMakeOutput.log".
```

解决方法：降低cmake版本无效，安装ninja-build后成功

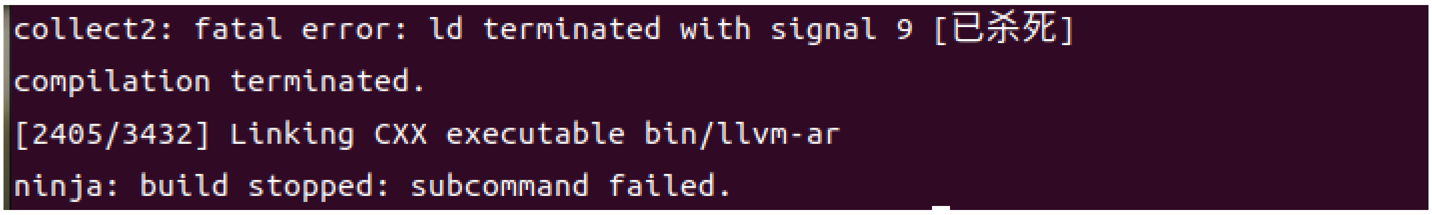
报错：内存不足，build被终止



```
c++: fatal error: 已杀死 signal terminated program cc1plus
compilation terminated.
[896/3233] Building CXX object lib/Tra...LLVMipo.dir/AttributorAttributes.cpp.o
ninja: build stopped: subcommand failed.
(base) mario@mario-virtual-machine:~/llvm-project-13.0.1.src/build$
```

解决方法：限制进程数，`ninja -C build check-llvm -j4` 限制4进程

链接报错：



```
collect2: fatal error: ld terminated with signal 9 [已杀死]
compilation terminated.
[2405/3432] Linking CXX executable bin/llvm-ar
ninja: build stopped: subcommand failed.
```

解决方法：可能是子任务出错，重新跑一遍就行。

报错： `ctime:80:11: error: 'timespec_get' has not been declared in '::'`

解决方法：论坛上说是conda环境和系统环境冲突，尝试用 `conda upgrade -c conda-forge --all` 更新conda无效。搞到后来又发现环境不一致问题，服了。考虑到一开始用的是miniconda3，最后改为安装miniforge3就不报错了。

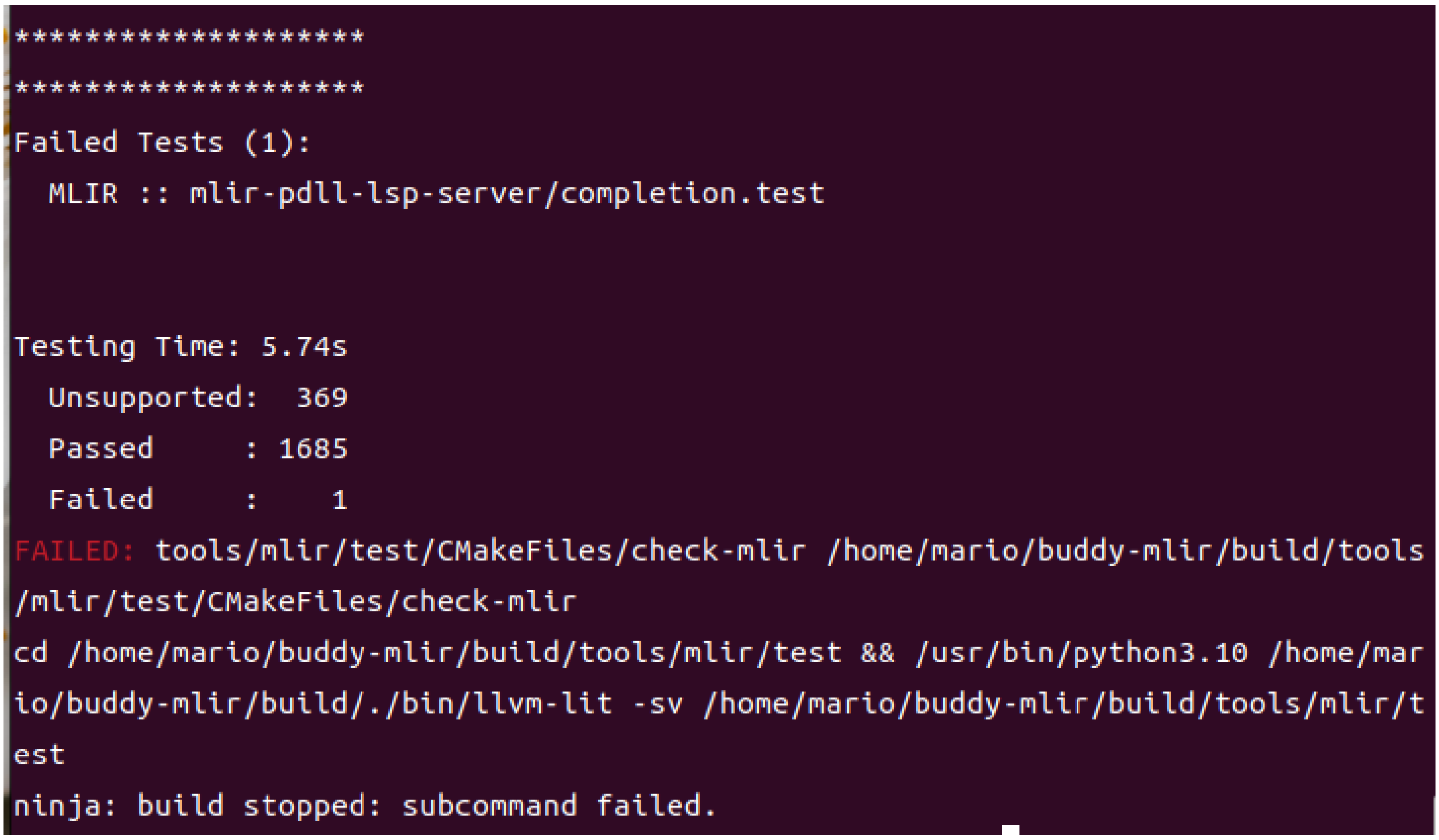
2.Clone and Initialize

很顺利

3.Build and Test LLVM/MLIR/CLANG

等待时间较长，还是一步安装的那个比较好hh

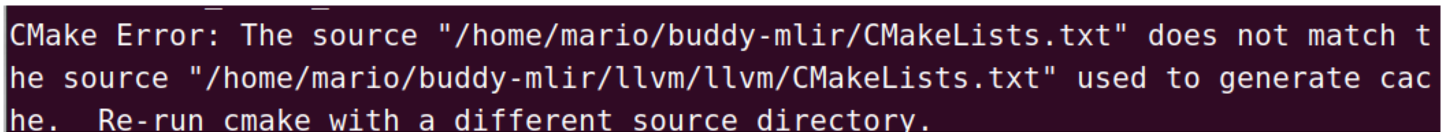
其中运行 `ninja check-mlir check-clang` 时有一个没pass



```
********************
********************
Failed Tests (1):
  MLIR :: mlir-pdll-lsp-server/completion.test


Testing Time: 5.74s
  Unsupported:  369
  Passed     : 1685
  Failed     :    1
FAILED: tools/mlir/test/CMakeFiles/check-mlir /home/mario/buddy-mlir/build/tools
/mlir/test/CMakeFiles/check-mlir
cd /home/mario/buddy-mlir/build/tools/mlir/test && /usr/bin/python3.10 /home/mar
io/buddy-mlir/build/./bin/llvm-lit -sv /home/mario/buddy-mlir/build/tools/mlir/t
est
ninja: build stopped: subcommand failed.
```

解决方法还没找到。它是一个无关紧要的包，不影响后续流程。

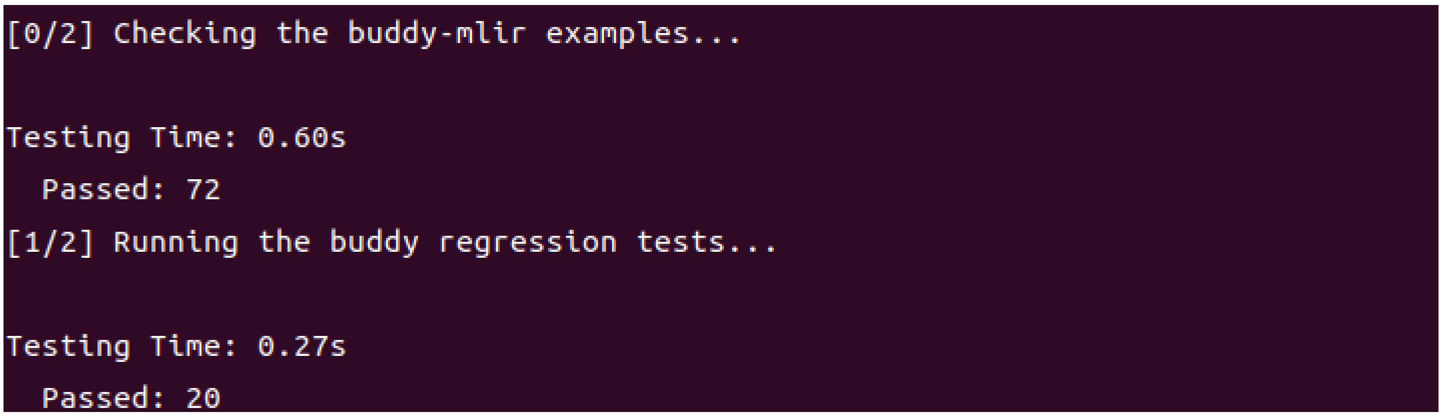
报错：CMakeLists.txt不匹配


```
CMake Error: The source "/home/mario/buddy-mlir/CMakeLists.txt" does not match t
he source "/home/mario/buddy-mlir/llvm/llvm/CMakeLists.txt" used to generate cac
he.  Re-run cmake with a different source directory.
```

解决方法：这是由于重复cmake且两次缓存不符导致的，删除CMakeCache.txt文件即可。

4.Build buddy-mlir

后面的都能正常运行，最后一条测试 ninja check-buddy结果：


```
[0/2] Checking the buddy-mlir examples...

Testing Time: 0.60s
  Passed: 72
[1/2] Running the buddy regression tests...

Testing Time: 0.27s
  Passed: 20
```

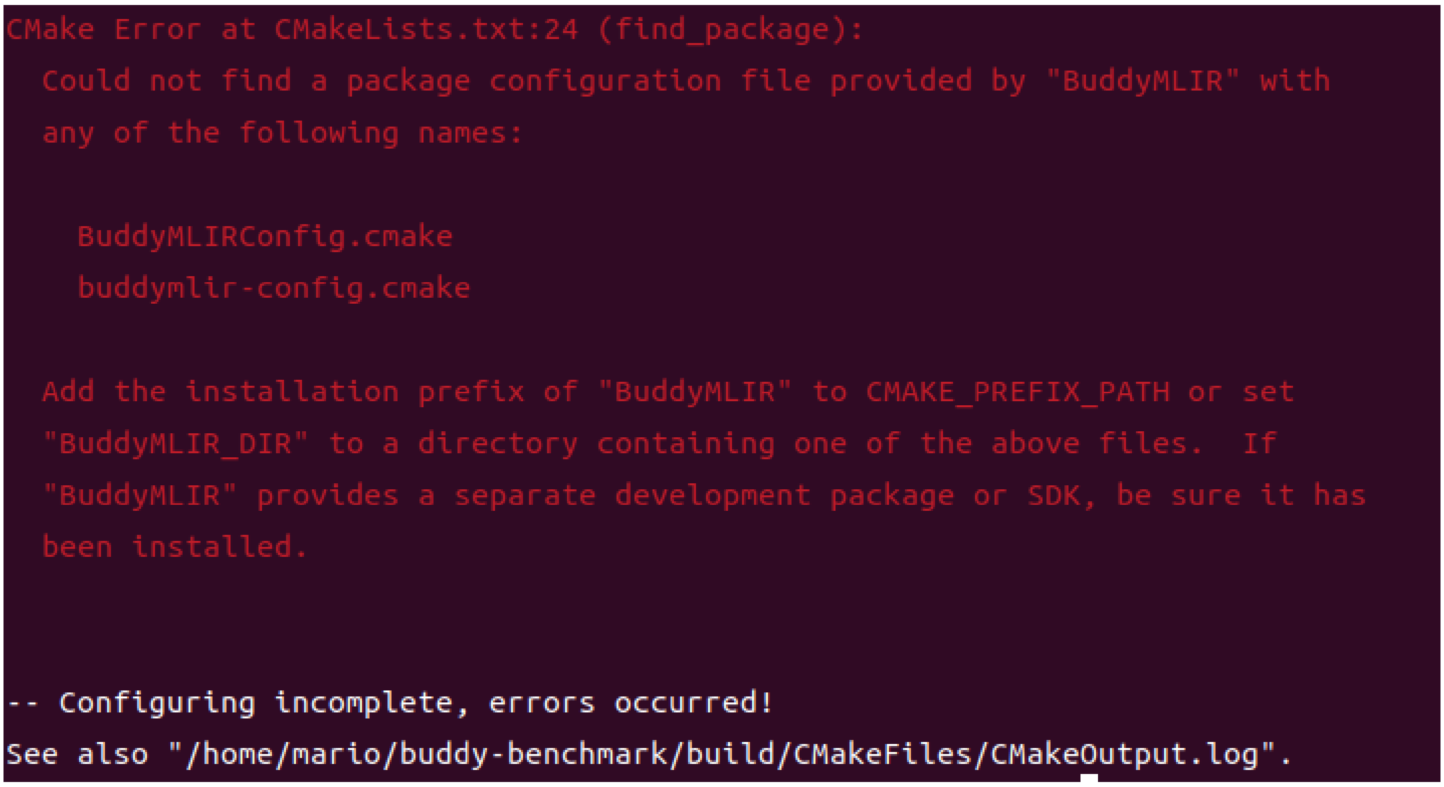
三、buddy-benchmark环境配置

1.Choose and Build Dependencies

很顺利

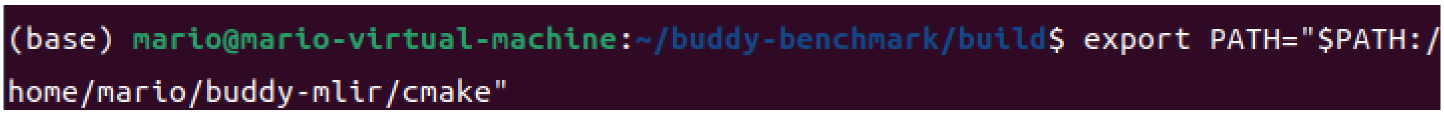
2.Gemmini Benchmark

报错：找不到buddy-mlir的包



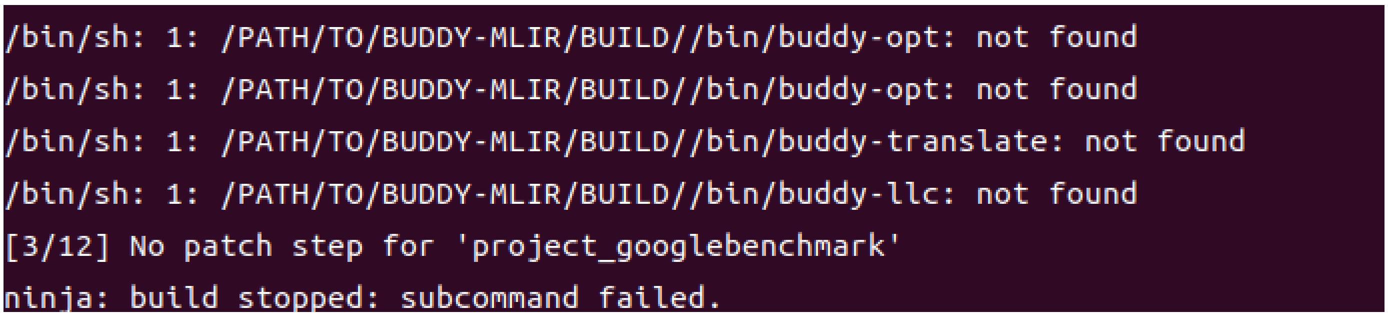
```
CMake Error at CMakeLists.txt:24 (find_package):
  Could not find a package configuration file provided by "BuddyMLIR" with
  any of the following names:

    BuddyMLIRConfig.cmake
    buddymlir-config.cmake

  Add the installation prefix of "BuddyMLIR" to CMAKE_PREFIX_PATH or set
  "BuddyMLIR_DIR" to a directory containing one of the above files.  If
  "BuddyMLIR" provides a separate development package or SDK, be sure it has
  been installed.


-- Configuring incomplete, errors occurred!
See also "/home/mario/buddy-benchmark/build/CMakeFiles/CMakeOutput.log".
```

解决方法：添加一下环境变量



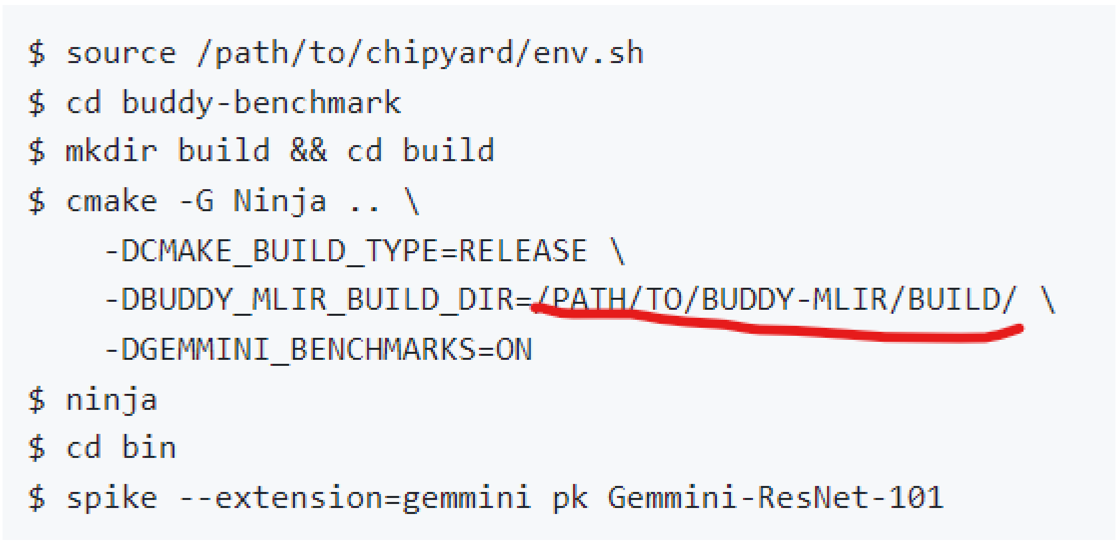
```
(base) mario@mario-virtual-machine:~/buddy-benchmark/build$ export PATH="$PATH:/
home/mario/buddy-mlir/cmake"
```

报错：找不到路径



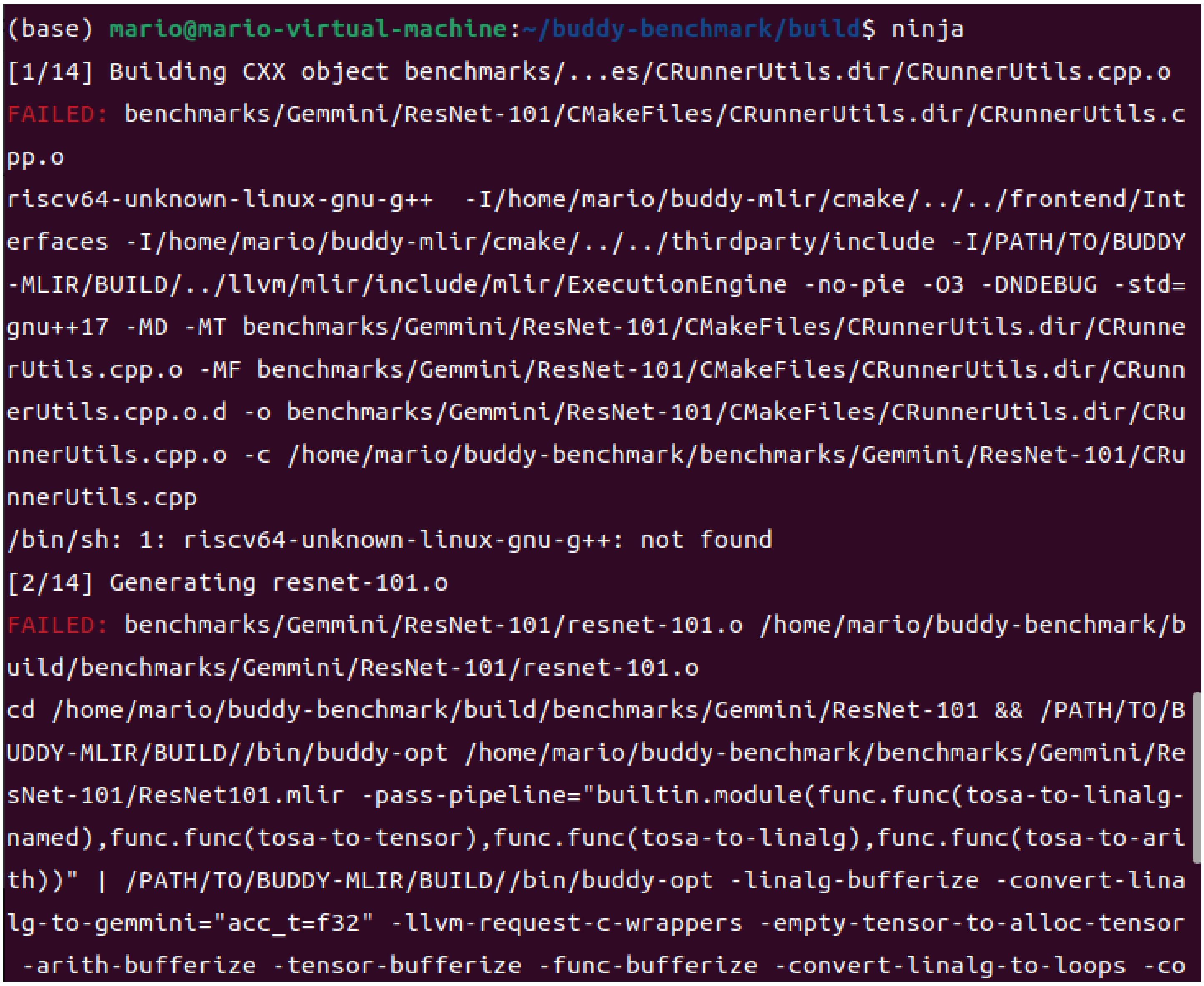
```
/bin/sh: 1: /PATH/TO/BUDDY-MLIR/BUILD//bin/buddy-opt: not found
/bin/sh: 1: /PATH/TO/BUDDY-MLIR/BUILD//bin/buddy-opt: not found
/bin/sh: 1: /PATH/TO/BUDDY-MLIR/BUILD//bin/buddy-translate: not found
/bin/sh: 1: /PATH/TO/BUDDY-MLIR/BUILD//bin/buddy-llc: not found
[3/12] No patch step for 'project_googlebenchmark'
ninja: build stopped: subcommand failed.
```

解决方法：上一步中的目录改成自己build的地址



```
$ source /path/to/chipyard/env.sh
$ cd buddy-benchmark
$ mkdir build && cd build
$ cmake -G Ninja .. \
    -DCMAKE_BUILD_TYPE=RELEASE \
    -DBUDDY_MLIR_BUILD_DIR=/PATH/TO/BUDDY-MLIR/BUILD/ \
    -DGEMMINI_BENCHMARKS=ON
$ ninja
$ cd bin
$ spike --extension=gemmini pk Gemmini-ResNet-101
```

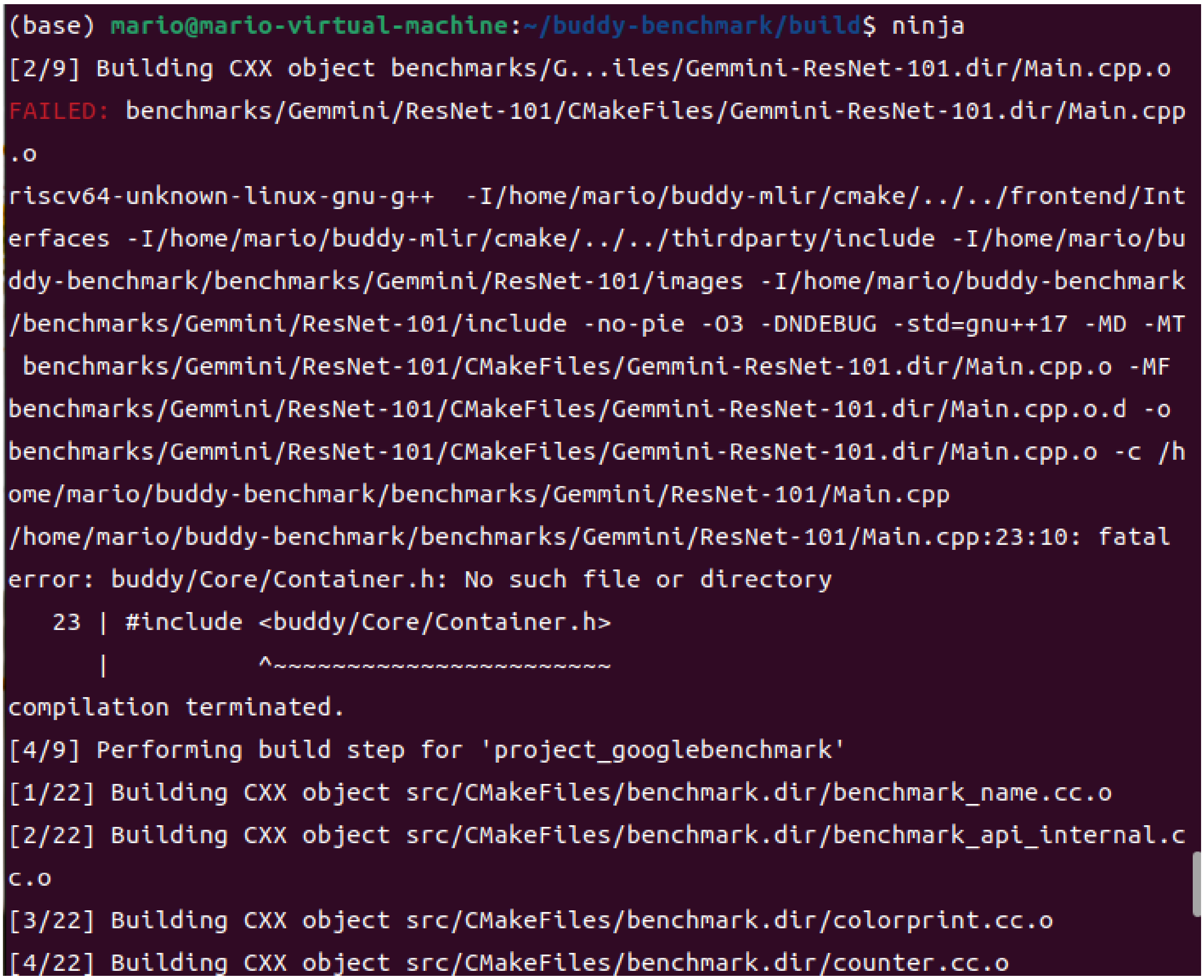
报错：找不到riscv64-unknown-linux-gnu-g++



```
(base) mario@mario-virtual-machine:~/buddy-benchmark/build$ ninja
[1/14] Building CXX object benchmarks/...es/CRunnerUtils.dir/CRunnerUtils.cpp.o
FAILED: benchmarks/Gemmini/ResNet-101/CMakeFiles/CRunnerUtils.dir/CRunnerUtils.c
pp.o
riscv64-unknown-linux-gnu-g++  -I/home/mario/buddy-mlir/cmake/../../frontend/Int
erfaces -I/home/mario/buddy-mlir/cmake/../../thirdparty/include -I/PATH/TO/BUDDY
-MLIR/BUILD/../llvm/mlir/include/mlir/ExecutionEngine -no-pie -O3 -DNDEBUG -std=
gnu++17 -MD -MT benchmarks/Gemmini/ResNet-101/CMakeFiles/CRunnerUtils.dir/CRunne
rUtils.cpp.o -MF benchmarks/Gemmini/ResNet-101/CMakeFiles/CRunnerUtils.dir/CRunn
erUtils.cpp.o.d -o benchmarks/Gemmini/ResNet-101/CMakeFiles/CRunnerUtils.dir/CRu
nnerUtils.cpp.o -c /home/mario/buddy-benchmark/benchmarks/Gemmini/ResNet-101/CRu
nnerUtils.cpp
/bin/sh: 1: riscv64-unknown-linux-gnu-g++: not found
[2/14] Generating resnet-101.o
FAILED: benchmarks/Gemmini/ResNet-101/resnet-101.o /home/mario/buddy-benchmark/b
uild/benchmarks/Gemmini/ResNet-101/resnet-101.o
cd /home/mario/buddy-benchmark/build/benchmarks/Gemmini/ResNet-101 && /PATH/TO/B
UDDY-MLIR/BUILD//bin/buddy-opt /home/mario/buddy-benchmark/benchmarks/Gemmini/Re
sNet-101/ResNet101.mlir -pass-pipeline="builtin.module(func.func(tosa-to-linalg-
named),func.func(tosa-to-tensor),func.func(tosa-to-linalg),func.func(tosa-to-ari
th))" | /PATH/TO/BUDDY-MLIR/BUILD//bin/buddy-opt -linalg-bufferize -convert-lina
lg-to-gemmini="acc_t=f32" -llvm-request-c-wrappers -empty-tensor-to-alloc-tensor
 -arith-bufferize -tensor-bufferize -func-bufferize -convert-linalg-to-loops -co
```

解决方法：换成chipyard中的conda环境就行

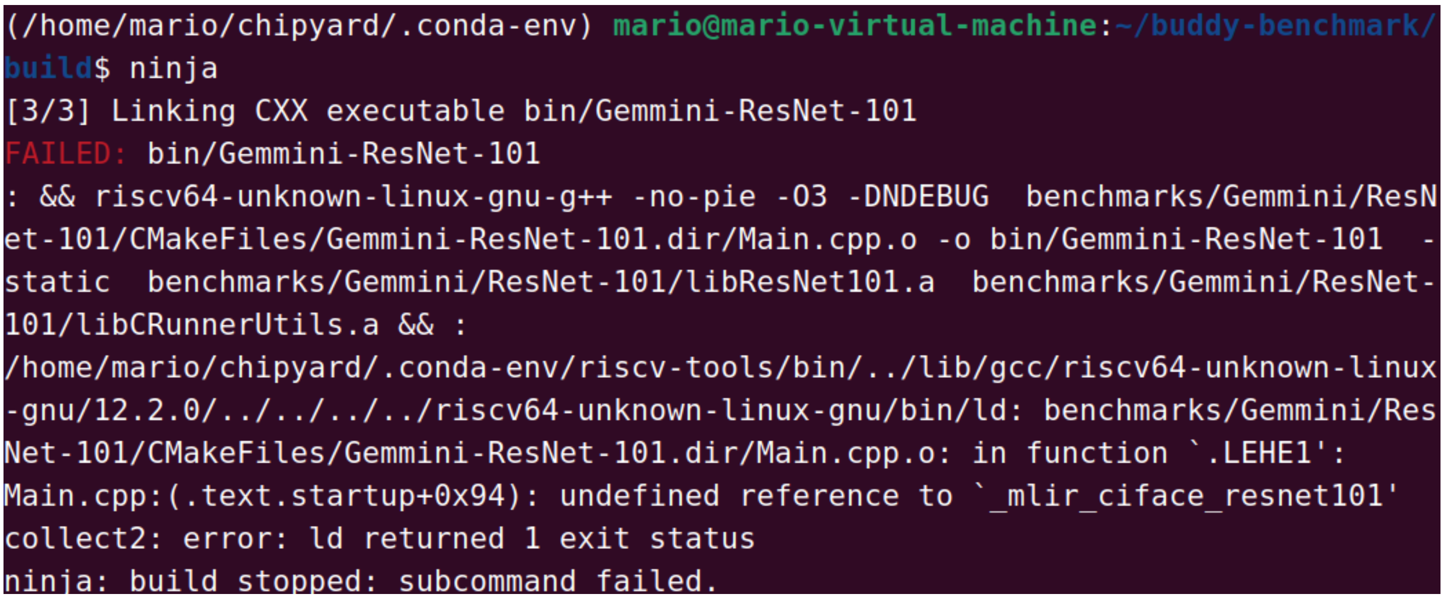
报错：找不到头文件



```
(base) mario@mario-virtual-machine:~/buddy-benchmark/build$ ninja
[2/9] Building CXX object benchmarks/G...iles/Gemmini-ResNet-101.dir/Main.cpp.o
FAILED: benchmarks/Gemmini/ResNet-101/CMakeFiles/Gemmini-ResNet-101.dir/Main.cpp
.o
riscv64-unknown-linux-gnu-g++  -I/home/mario/buddy-mlir/cmake/../../frontend/Int
erfaces -I/home/mario/buddy-mlir/cmake/../../thirdparty/include -I/home/mario/bu
ddy-benchmark/benchmarks/Gemmini/ResNet-101/images -I/home/mario/buddy-benchmark
/benchmarks/Gemmini/ResNet-101/include -no-pie -O3 -DNDEBUG -std=gnu++17 -MD -MT
 benchmarks/Gemmini/ResNet-101/CMakeFiles/Gemmini-ResNet-101.dir/Main.cpp.o -MF
benchmarks/Gemmini/ResNet-101/CMakeFiles/Gemmini-ResNet-101.dir/Main.cpp.o.d -o
benchmarks/Gemmini/ResNet-101/CMakeFiles/Gemmini-ResNet-101.dir/Main.cpp.o -c /h
ome/mario/buddy-benchmark/benchmarks/Gemmini/ResNet-101/Main.cpp
/home/mario/buddy-benchmark/benchmarks/Gemmini/ResNet-101/Main.cpp:23:10: fatal
error: buddy/Core/Container.h: No such file or directory
   23 | #include <buddy/Core/Container.h>
      |          ^~~~~~~~~~~~~~~~~~~~~~~~
compilation terminated.
[4/9] Performing build step for 'project_googlebenchmark'
[1/22] Building CXX object src/CMakeFiles/benchmark.dir/benchmark_name.cc.o
[2/22] Building CXX object src/CMakeFiles/benchmark.dir/benchmark_api_internal.c
c.o
[3/22] Building CXX object src/CMakeFiles/benchmark.dir/colorprint.cc.o
[4/22] Building CXX object src/CMakeFiles/benchmark.dir/counter.cc.o
```

解决方法：尝试从另一个文件中获取同名文件看看能不能用，能通过但是最后有一个其它地方报错了，说明不仅仅是缺少这一个头文件的问题，应该是之前的安装有些疏忽。换了miniforge环境，重复一遍buddy-mlir的安装，再走一遍，头文件就能找到了。

头文件解决后，ninja时又出现如下报错：

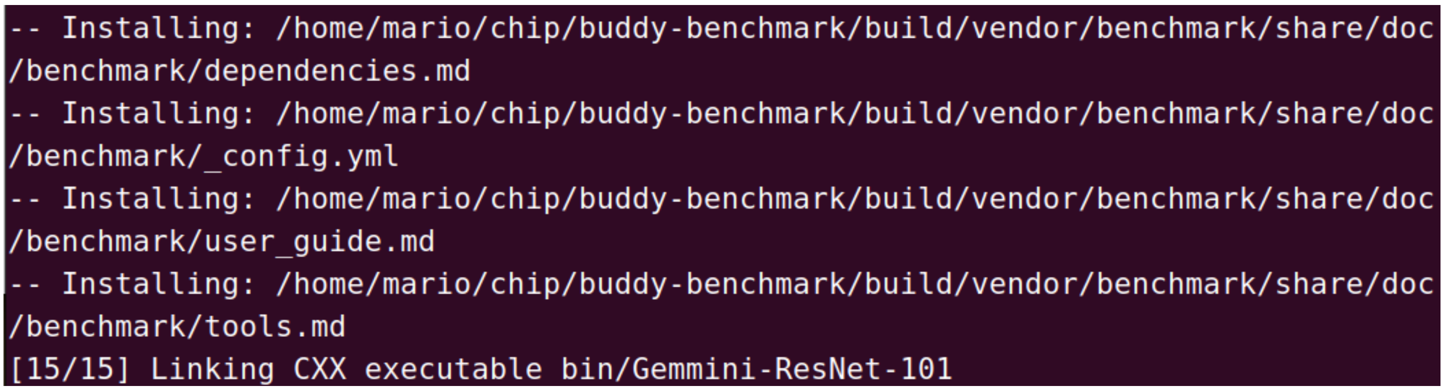


```
(/home/mario/chipyard/.conda-env) mario@mario-virtual-machine:~/buddy-benchmark/build$ ninja
[3/3] Linking CXX executable bin/Gemmini-ResNet-101
FAILED: bin/Gemmini-ResNet-101
: && riscv64-unknown-linux-gnu-g++ -no-pie -O3 -DNDEBUG  benchmarks/Gemmini/ResNet-101/CMakeFiles/Gemmini-ResNet-101.dir/Main.cpp.o -o bin/Gemmini-ResNet-101  -static  benchmarks/Gemmini/ResNet-101/libResNet101.a  benchmarks/Gemmini/ResNet-101/libCRunnerUtils.a && :
/home/mario/chipyard/.conda-env/riscv-tools/bin/../lib/gcc/riscv64-unknown-linux-gnu/12.2.0/../../../../riscv64-unknown-linux-gnu/bin/ld: benchmarks/Gemmini/ResNet-101/CMakeFiles/Gemmini-ResNet-101.dir/Main.cpp.o: in function `.LEHE1':
Main.cpp:(.text.startup+0x94): undefined reference to `_mlir_ciface_resnet101'
collect2: error: ld returned 1 exit status
ninja: build stopped: subcommand failed.
```

解决方法：尝试过重装chipyard、buddy-mlir、自行配置llvm均无效，最后在issue中找到方案。需要安装git lfs，保证ResNet-101被成功clone，具体代码如下：

```
curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.s
sudo apt-get install git-lfs
git lfs install
```

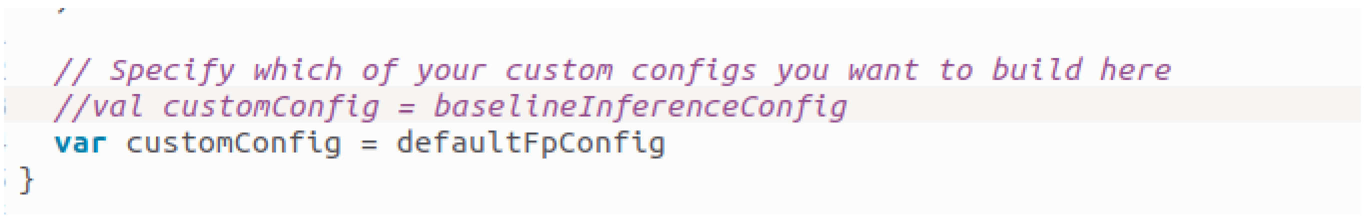
重新clone一下buddy-benchmark，发现这次可以运行了



```
-- Installing: /home/mario/chip/buddy-benchmark/build/vendor/benchmark/share/doc/benchmark/dependencies.md
-- Installing: /home/mario/chip/buddy-benchmark/build/vendor/benchmark/share/doc/benchmark/_config.yml
-- Installing: /home/mario/chip/buddy-benchmark/build/vendor/benchmark/share/doc/benchmark/user_guide.md
-- Installing: /home/mario/chip/buddy-benchmark/build/vendor/benchmark/share/doc/benchmark/tools.md
[15/15] Linking CXX executable bin/Gemmini-ResNet-101
```

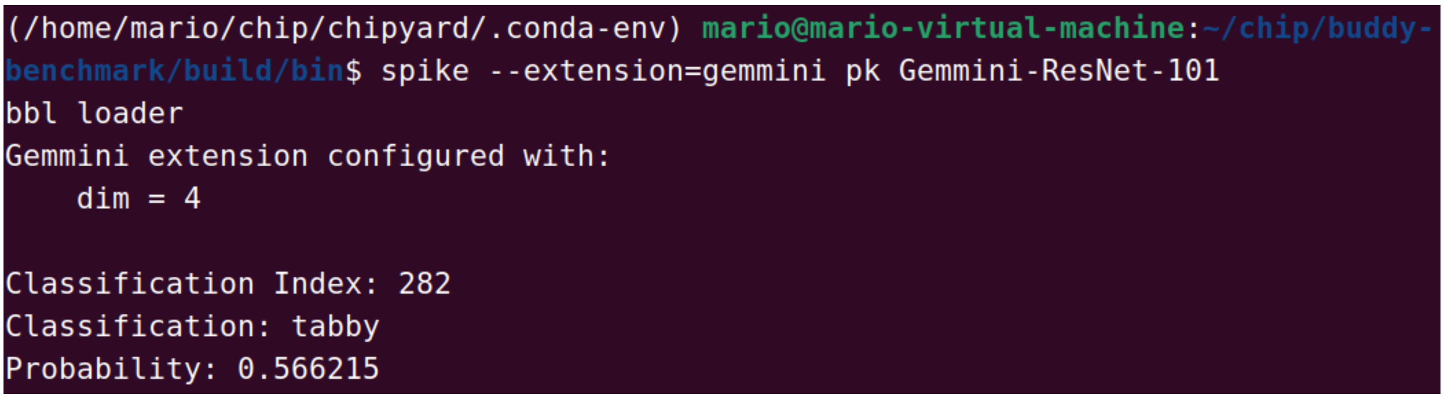
果然是缺少文件。

最后，修改一下chipyard中GemminiCustomConfigs.scala文件配置，将baselineInferenceConfig改为defaultFpConfig，重新生成spike。



```
  // Specify which of your custom configs you want to build here
  //val customConfig = baselineInferenceConfig
  var customConfig = defaultFpConfig
}
```

完结撒花~~~

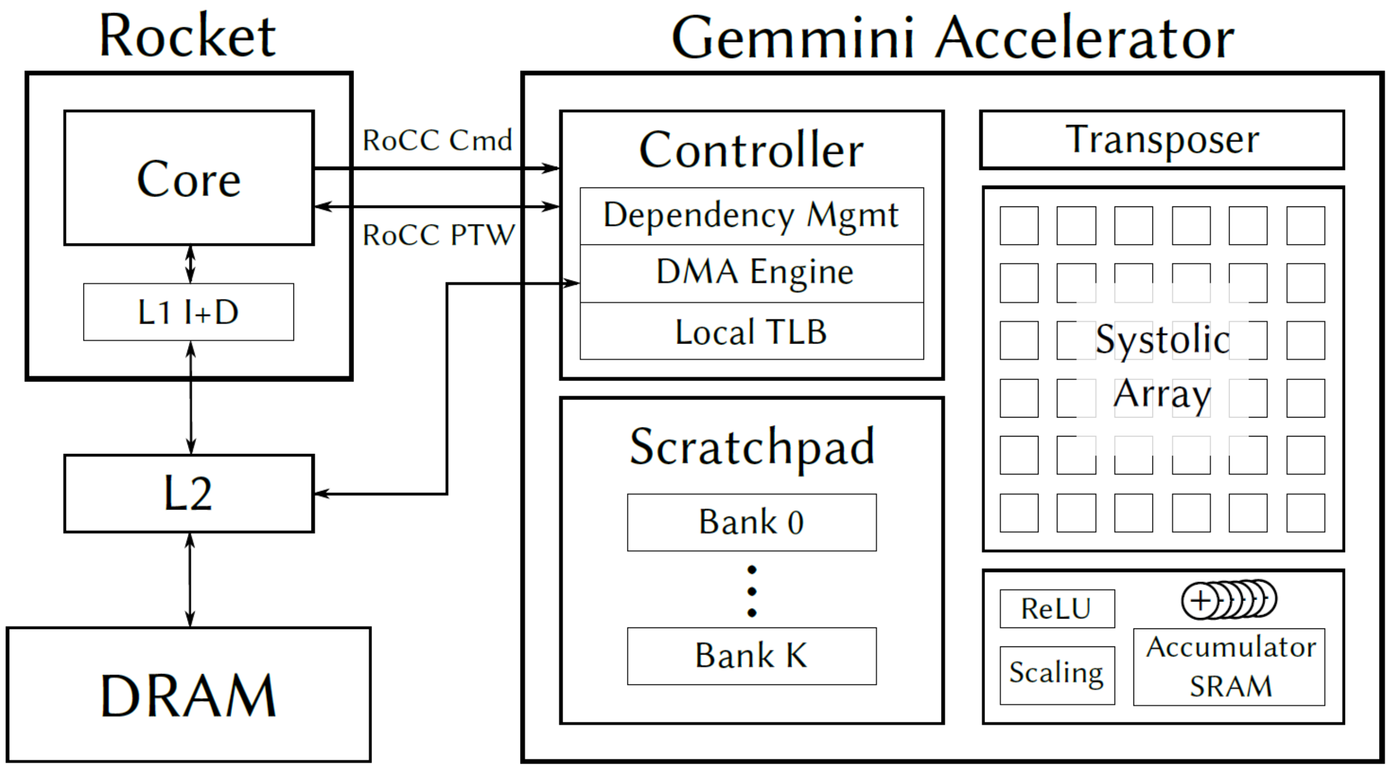


```
(/home/mario/chip/chipyard/.conda-env) mario@mario-virtual-machine:~/chip/buddy-benchmark/build/bin$ spike --extension=gemmini pk Gemmini-ResNet-101
bbl loader
Gemmini extension configured with:
    dim = 4

Classification Index: 282
Classification: tabby
Probability: 0.566215
```

2.Gemmini架构

一、Gemmini概述

Gemmini项目是一个全系统、全栈的DNN硬件探索和评估平台，使得我们能够深入了解系统和软件堆栈之间的相互作用对DNN性能的影响。结构图如下所示：



Gemmini使用RISC-V自定义指令实现RoCC加速器，其单元使用 Rocket 或 BOOM块的 RoCC 端口，默认情况下通过系统总线连接到内存系统。

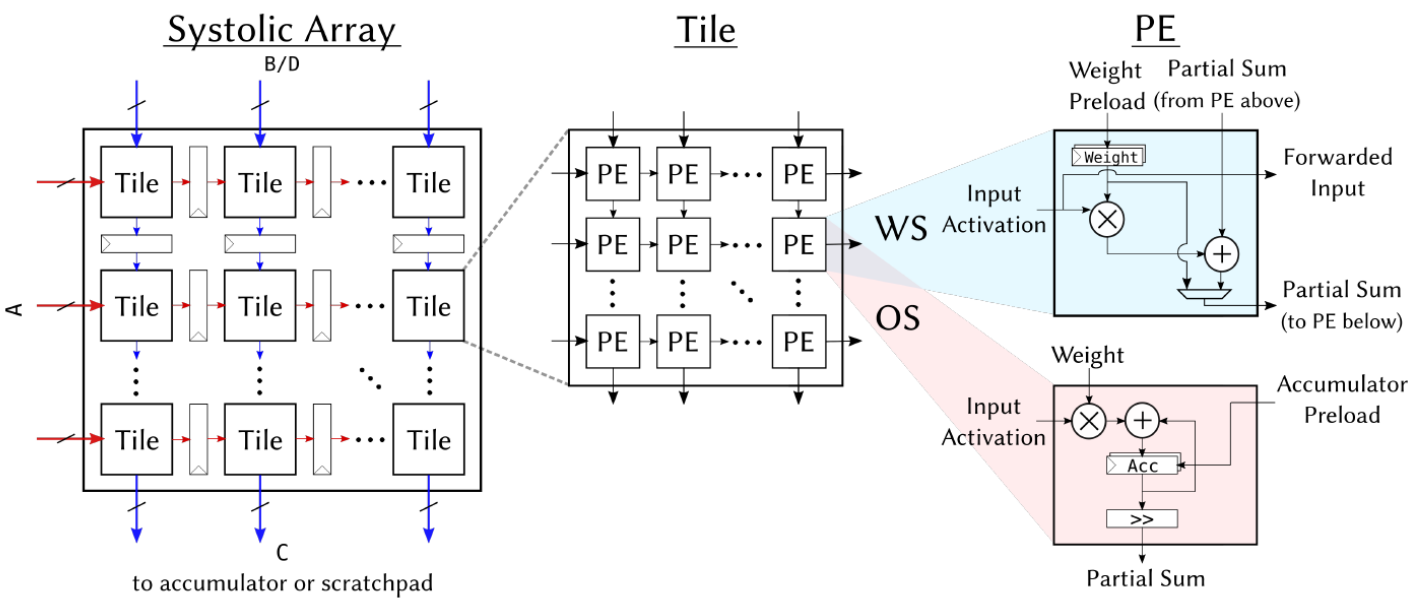
该加速器的核心是脉动阵列，主要用于执行矩阵乘法，支持输出平稳和权重平稳数据流。

阵列的输入和输出存储在由成组SRAM组成的显式管理暂存器中，由DMA加速暂存器数据和主存数据的传输。

此外，由于在处理权重平稳数据流时需要使用脉动阵列外部的累加器，所以需要添加一个配备加法器单元的最终SRAM组，这样脉动数组就能将结果存入任意地址的累加器或者从累加器读取数据。

二、脉动阵列

脉动阵列是加速器的核心组成部分，结构如下：



其主要参数有：

1. 脉动数组维度 (tileRows, tileColumns, meshRows, meshColumns)：脉动数组由 2 级层次结构组成，其中每个图块都是完全组合的，而图块网格在每个图块之间具有管道寄存器。
2. 暂存器和累加器存储器参数 (sp_banks、sp_capacity、acc_capacity)：暂存器或累加器的总容量（单位KB）
3. 数据流参数 (dateflow)：确定 Gemmini 中的脉动数组是输出平稳还是重量平稳。
4. 类型参数 (inputType、outputType、accType)：确定流经 Gemmini 加速器不同部分的数据类型，例如8位定点数、32位整数等。
5. 访问执行队列参数（ld_queue_length、st_queue_length、ex_queue_length、rob_entries）：为实现访问执行解耦，Gemmini 加速器具有加载指令队列、存储指令队列和执行指令队列。这些队列的相对大小决定了访问执行解耦的级别。
6. DMA 参数 (dma_maxbytes、dma_buswidth、mem_pipeline)：Gemmini 实现 DMA 将数据从主存储器移动到 Gemmini 暂存器，以及从 Gemmini 累加器移动到主存储器。这些 DMA 事务的大小由 DMA 参数决定。这些DMA参数与Rocket Chip SoC系统参数紧密耦合。

脉动阵列中最基础的单元为PE，功能是实现一维乘加运算。大量PE排列成方形，组成一个Tile，Tile之间彼此相连，两两之间有寄存器可以利用。计算前需要将数据存入脉动阵列或者周围（图中红线和蓝线所连接）的存储器中。

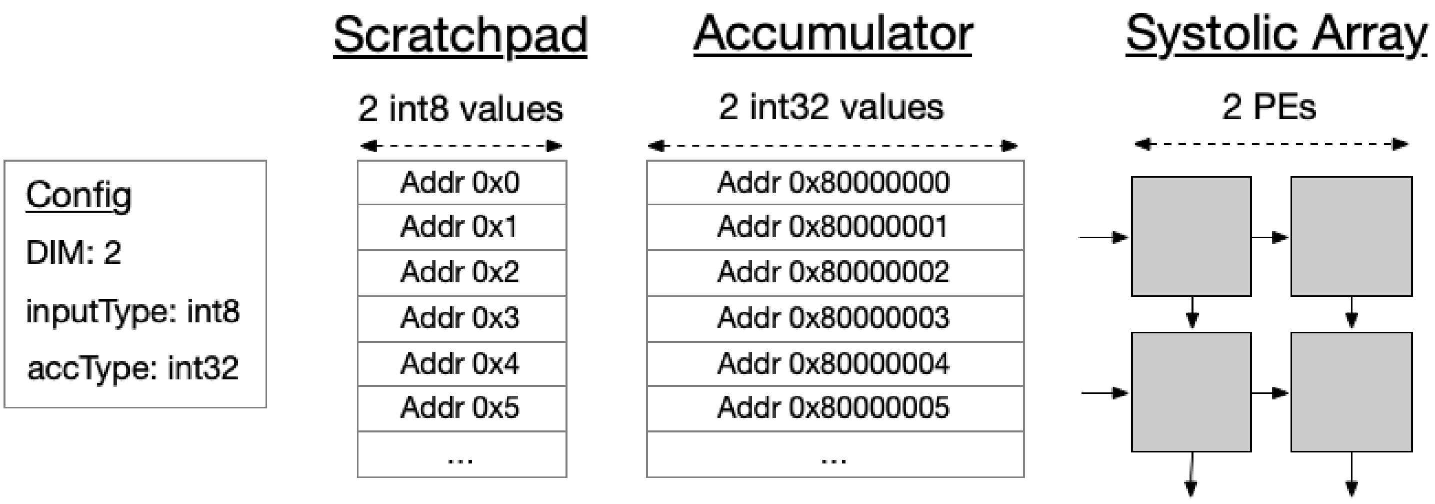
三、软件部分

Gemmini在配置指令、数据移动指令和矩阵乘法执行指令方面指定指令集。然后生成器将Gemmini自定义指令包装到DNN运算符中，相关宏定义可见 `software/gemmini-rocc-tests/include/gemmini.h` 。生成器还会根据参数生成C头文件，共同编译，调整库性能。此外，还可以通过Microsoft ONNX-Runtime框架的端口运行ONNX指定的神经网络。

四、内存寻址

下图是2*2脉动阵列内存寻址的示意图。内存寻址按行进行，图中Tile一行有2个PE单元，则宽度DIM为2。其中暂存器数据类型是8位的int值，累加器数据类型是32位的int值。

Gemmini中的内存编码均为32位长，最高的3位是保留位，用于区分寻址累加器还是暂存器、写入时是覆盖还是继续累加、从何处读取数据、是否读取按比例缩小的数据等等。

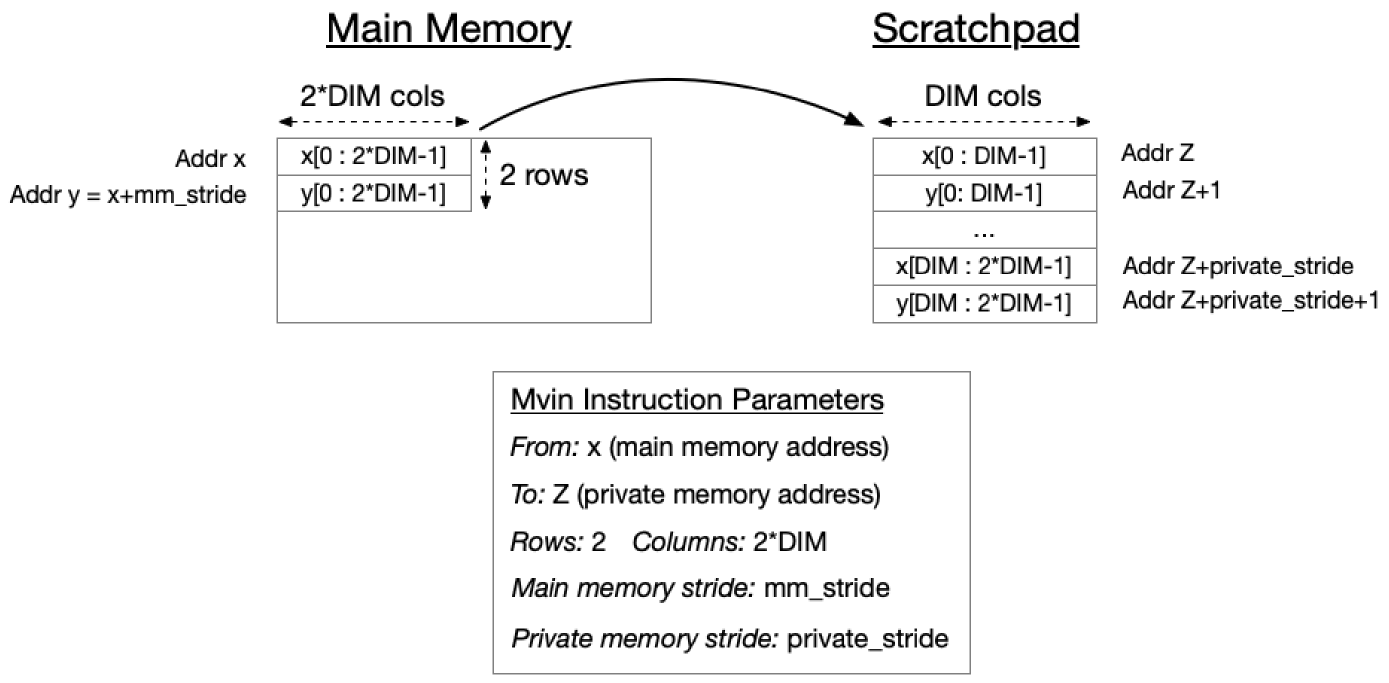


五、指令

这里只列举两个数据移动指令，更多的指令可以在Gemmini仓库查看。

（i）将数据从主存储器移至暂存器：mvin

指令 `mvin rs1, rs2` 中，rs1是加载到暂存器的虚拟 DRAM 地址，rs2的0~31位记录本地暂存器或累加器地址，32~63位记录要加载的行列数。即数据从rs1流向rs2。其中加载的列数必须小于Tile中每行TE宽度DIM，否则会拆分后分块加载。



（ii）将数据从暂存器移至 L2/DRAM：mvout

指令 `mvout rs1, rs2` 中，rs1是从暂存器写入的虚拟 DRAM 地址，rs2的0~31位记录本地暂存器地址，32~63位同样记录加载行列数。即数据从rs2流向rs1。

3.MLIR

一、MLIR概述

IR是深度学习中模型的中间表示，包括算子和数据。但由于当前IR众多，造成维护和迁移的困难，而MLIR的出现就是为解决这个问题。

首先，MLIR 提出了 Dialect，即各种 IR 需要学习的语言，一旦某种 IR 学会这种语言，就可以基于这种语言将其重写为 MLIR。

此外，MLIR 还通过 Dialect 抽象出了多种不同级别的 MLIR，以应对IR跨度大的难题。例如源程序在经过语法树分析和MLIR分析后，得到的目标程序再作为下一次编译的源程序，通过MLIR Dialect 多次下降，最终得到可执行的机器码，类似于渐进式学习。

二、toy语言

MLIR中提供了toy语言以说明MLIR的执行流程，是验证和说明说用，因此语法和功能都非常简单。它是一种基于张量的语言，只支持64位浮点类型数据，适合于进行函数定义和数学计算。具体语法见官网。

按照官网的语法教程看如下一个MLIR编译流程的例子，主要进行矩阵转置相乘。

```
# User defined generic function that operates on unknown shaped arguments.
```

```
def multiply_transpose(a, b) {
  return transpose(a) * transpose(b);
}

def main() {
  # Define a variable `a` with shape <2, 3>, initialized with the literal value.
  var a = [[1, 2, 3], [4, 5, 6]];
  var b<2, 3> = [1, 2, 3, 4, 5, 6];

  # This call will specialize `multiply_transpose` with <2, 3> for both
  # arguments and deduce a return type of <3, 2> in initialization of `c`.
  var c = multiply_transpose(a, b);

  # A second call to `multiply_transpose` with <2, 3> for both arguments will
  # reuse the previously specialized and inferred version and return <3, 2>.
  var d = multiply_transpose(b, a);

  # A new call with <3, 2> (instead of <2, 3>) for both dimensions will
  # trigger another specialization of `multiply_transpose`.
  var e = multiply_transpose(b, c);

  # Finally, calling into `multiply_transpose` with incompatible shape will
  # trigger a shape inference error.
  var f = multiply_transpose(transpose(a), c);
}
```

该程序生成AST再生成初级MLIR如下：

```
module  {
  func @multiply_transpose(%arg0: tensor<*xf64> loc("../../mlir/test/Examples/To
    %0 = toy.transpose(%arg0 : tensor<*xf64>) to tensor<*xf64> loc("../../mlir/t
    %1 = toy.transpose(%arg1 : tensor<*xf64>) to tensor<*xf64> loc("../../mlir/t
    %2 = toy.mul %0, %1 : tensor<*xf64> loc("../../mlir/test/Examples/Toy/Ch2/co
    toy.return %2 : tensor<*xf64> loc("../../mlir/test/Examples/Toy/Ch2/codegen.
  } loc("../../mlir/test/Examples/Toy/Ch2/codegen.toy":4:1)
  func @main() {
    %0 = toy.constant dense<[[1.000000e+00, 2.000000e+00, 3.000000e+00], [4.0000
    %1 = toy.reshape(%0 : tensor<2x3xf64>) to tensor<2x3xf64> loc("../../mlir/te
    %2 = toy.constant dense<[1.000000e+00, 2.000000e+00, 3.000000e+00, 4.000000e
    %3 = toy.reshape(%2 : tensor<6xf64>) to tensor<2x3xf64> loc("../../mlir/test
    %4 = toy.generic_call @multiply_transpose(%1, %3) : (tensor<2x3xf64>, tensor
    %5 = toy.generic_call @multiply_transpose(%3, %1) : (tensor<2x3xf64>, tensor
    toy.print %5 : tensor<*xf64> loc("../../mlir/test/Examples/Toy/Ch2/codegen.t
    toy.return loc("../../mlir/test/Examples/Toy/Ch2/codegen.toy":8:1)
  } loc("../../mlir/test/Examples/Toy/Ch2/codegen.toy":8:1)
} loc(unknown)
```

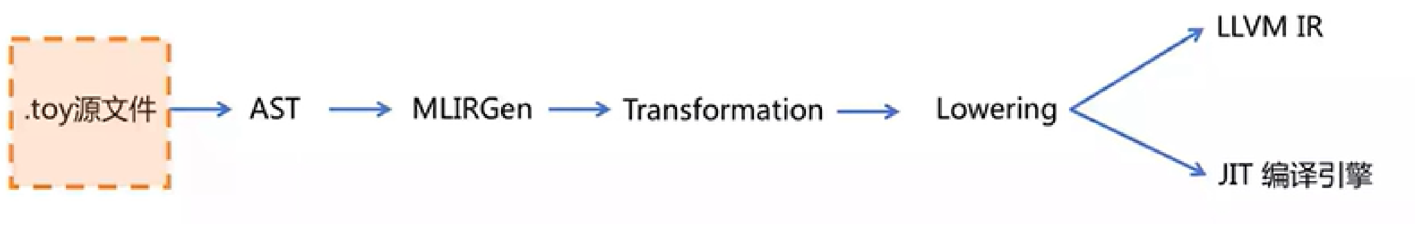
代码中红色部分是冗余的reshape操作，需要通过Pass来识别和优化冗余。此时有两种途径，一种是使用C++编写匹配和重写函数来消除冗余，另一种是采用DRR自动生成函数。

优化后main函数部分表达式如下，可见已经非常简化了。

```
module  {
  func @main() {
    %0 = toy.constant dense<[[1.000000e+00, 2.000000e+00, 3.000000e+00], [4.0000
    %1 = toy.transpose(%0 : tensor<2x3xf64>) to tensor<3x2xf64>
    %2 = toy.mul %1, %1 : tensor<3x2xf64>
    toy.print %2 : tensor<3x2xf64>
    toy.return
  }
}
```

该段代码还是比较高层的，还可以继续部分Lowering，混合不同的Dialect Lowering到LLVM IR。

总体流程可以如下表示：



三、Gemmini方言分析

由于指令较多，这里只选取几个详细讲解。

（1）mvin-mvout

```
mvin-mvout-run:
	@${BUDDY_OPT} ./mvin-mvout.mlir -lower-gemmini | \
	${BUDDY_TRANSLATE} --buddy-to-llvmir | \
	${BUDDY_LLC} -filetype=obj -mtriple=riscv64 \
		-mattr=+buddyext,+D -float-abi=hard \
		-o log.o
	@riscv64-unknown-linux-gnu-gcc log.o -O2 -static -o a.out
	@spike --extension=gemmini pk a.out

```

在上述代码的第2行，操作是执行了mvin-mvout.mlir，然后有一个lower操作，下降到llvmir。比较简单的例子都是这样的结构，只是前面操作的mlir文件有所不同。在此打开mvin-mvout.mlir文件。

```
 1 // RUN: buddy-opt %s \
 2 // RUN:     --lower-gemmini | \
 3 // RUN: FileCheck %s
 4 
 5 memref.global "private" @gv : memref<2x16xi8> = dense<[[0, 1, 2, 3, 4, 5, 6, 7,
 6                                                         [16, 17, 18, 19, 20, 21,
 7 
 8 func.func @main() -> i64 {
 9   %0 = arith.constant 0 : i64
10   %stride16 = arith.constant 16 : i64
11   %stride8 = arith.constant 8 : i64
12   %spadAddr = arith.constant 0 : i64 //暂存器地址
13   %arrayA = memref.get_global @gv : memref<2x16xi8>   //获取全局变量
14   %arrayB = memref.alloc() : memref<3x16xi8>   //分配内存
15   %arrayC = memref.alloc() : memref<2x8xi8>
16   gemmini.print %arrayB : memref<3x16xi8>
17   gemmini.print %arrayC : memref<2x8xi8>
18   // CHECK: "gemmini.intr.config_st"
19   // The mvout op's stride is 16.
20   gemmini.config_st %stride16 : i64     //确定mvout的步长16
21   // CHECK: "gemmini.intr.config_ld"
22   // The mvin op's stride is 16
23   gemmini.config_ld %stride16 : i64     //确定mvin的步长16
24   // CHECK: "gemmini.intr.mvin"
25   gemmini.mvin %arrayA %spadAddr : memref<2x16xi8> i64
26   // CHECK: "gemmini.intr.mvout"
27   gemmini.mvout %arrayB %spadAddr : memref<3x16xi8> i64
28   // CHECK: "gemmini.intr.config_st"
29   // The mvout op's stride is 8
30   gemmini.config_st %stride8 : i64     //步长为8,因为C矩阵大小为2*8,按行寻址,DIM为每行
31   // CHECK: "gemmini.intr.mvout"
32   gemmini.mvout %arrayC %spadAddr : memref<2x8xi8> i64
33   gemmini.print %arrayB : memref<3x16xi8>
34   gemmini.print %arrayC : memref<2x8xi8>
35   return %0 : i64
36 }
```

从上往下，第5行是全局变量的声明，然后main函数中有三个矩阵，分别是A、B、C，其中A赋值为全局变量，B、C仅分配了内存。然后在第20行有一条config_st指令，需要查阅td文件。在td文件中找到config_st的定义：

```
def ConfigStOp : Gemmini_Op<"config_st"> {
  let summary = "Config store operation";
  let description = [{
    TODO: consider the attribute type according to Gemmini spec.
  }];
  let arguments = (ins I64:$stride,
                       DefaultValuedAttr<I64Attr, "0">:$activation,
                       DefaultValuedAttr<F32Attr, "1.0">:$scale);
  let assemblyFormat = "$stride attr-dict `:` type($stride)";
}
```

从description中可以看出，这条指令用于根据特定的Gemmini确定mvin步长参数。

第23行的config_ld同理也是确定步长的，只不过是确定mvout的步长而已。

接下来就是矩阵A数据移动到暂存器，暂存器数据移动到矩阵B的过程。当暂存器中的数据移动到矩阵C时，需要指定步长为8，因为C矩阵的大小为2*8，根据之前所说的按行寻址模式，宽度为8。当然，宽度不能超过脉动矩阵的最大宽度，即一个Tile中一行的PE元件数量，否则会发生切割后分块传输。

最后是一些打印过程，不再赘述。

（2）matmul-os

```
matmul-os-run:
	@${BUDDY_OPT} ./matmul-os.mlir -lower-gemmini | \
	${BUDDY_TRANSLATE} --buddy-to-llvmir | \
	${BUDDY_LLC} -filetype=obj -mtriple=riscv64 \
		-mattr=+buddyext,+D -float-abi=hard \
		-o log.o
	@riscv64-unknown-linux-gnu-gcc log.o -O2 -static -o a.out
	@spike --extension=gemmini pk a.out
```

这是一个矩阵相乘，并且输出稳定的例子。同样地，打开matmul-os.mlir文件如下：

```
// RUN: buddy-opt %s \
// RUN:     --lower-gemmini | \
// RUN: FileCheck %s

memref.global "private" @gv1 : memref<4x4xi8> = dense<[[1, 2, 3, 4],
                                                       [5, 6, 7, 8],
                                                       [9, 10, 11, 12],
                                                       [13, 14, 15, 16]]>
memref.global "private" @gv2 : memref<4x4xi8> = dense<[[1, 1, 1, 1],
                                                       [1, 1, 1, 1],
                                                       [1, 1, 1, 1],
                                                       [1, 1, 1, 1]]>

func.func @main() -> i64 {
  %in = memref.get_global @gv1 : memref<4x4xi8>   //获取全局变量
  %identity = memref.get_global @gv2 : memref<4x4xi8>  //获取全局变量
  %out = memref.alloc() : memref<4x4xi8>  //分配内存
  gemmini.print %out : memref<4x4xi8>
  %inSpAddr = arith.constant 0 : i64   //指定输入暂存器的地址
  %outSpAddr = arith.constant 4 : i64   //指定输出暂存器的地址
  %identitySpAddr = arith.constant 8 : i64
  %cst4 = arith.constant 4 : i64
  %cst0 = arith.constant 0 : i64
  // CHECK: "gemmini.intr.config_st"
  gemmini.config_st %cst4 : i64  //指定输入暂存器的步长
  // CHECK: "gemmini.intr.config_ld"
  gemmini.config_ld %cst4 : i64  //指定输出暂存器的步长
  // CHECK: "gemmini.intr.mvin"
  gemmini.mvin %in %inSpAddr : memref<4x4xi8> i64
  // CHECK: "gemmini.intr.config_ld"
  gemmini.config_ld %cst4 : i64
  // CHECK: "gemmini.intr.mvin"
  gemmini.mvin %identity %identitySpAddr : memref<4x4xi8> i64
  // CHECK: "gemmini.intr.config_ex"
  gemmini.config_ex {dataflow = 0 }   //配置执行管道
  // CHECK: "gemmini.intr.preload"
  gemmini.preload_zeros %outSpAddr %cst4 %cst4 : i64 i64 i64  //预加载为0
  // CHECK: "gemmini.intr.compute_preloaded"
  //执行矩阵乘法
  gemmini.compute_preloaded %inSpAddr %identitySpAddr %cst4 %cst4 %cst4 %cst4 :
  // CHECK: "gemmini.intr.mvout"
  gemmini.mvout %out %outSpAddr : memref<4x4xi8> i64  //输出到主存
  gemmini.print %out : memref<4x4xi8>
  return %cst0 : i64
}
```

其中，config_ex指令用于配置执行管道，包括配置数据流是输出稳定还是权重稳定，是否转置等等。值得一提的是，转置操作也是直接通过config_ex指令实现的。preload_zeros指令用于在输出的暂存器地址上预置0，因为默认是会将计算结果直接加上当前数。compute_preloaded指令则是用于计算矩阵相乘的结果，最后从outAddr取出计算结果。