The OpenROAD Project

OpenROAD seeks to develop and foster an autonomous, 24-hour, open-source layout generation flow (RTL-to-GDS).

https://theopenroadproject.org/

使用OpenROAD构建蜂鸟E203处理器核GDS

张洪滨

2020.06.04

《高级计算机系统结构》课程设计 – 一生一芯

- QFlow 历史最为悠久，上手难度最低
- VSDFLOW 基于 QFlow 构建，也存在相同问题
- OpenROAD 处于活跃期，非常值得探索

使用开源的 EDA 工具链，每位同学探索完成小规模芯片从 RTL 代码设计到 GDS 的全流程，包括: 架构设计、前端 RTL 设计、综合、后端物理设计(含布局、 时钟树综合、布线、时序分析等)。课程 Project 的 RTL 设计可以是同学自行设计，也可以是任何开源芯片的 RTL 设计代码。

参考自国科大《高级计算机系统结构》课程设计指导文档

《高级计算机系统结构》课程设计 – 一生一芯

硬件描述语言
HDL层次 - - - - - - - → 记录版图信息的
二进制文件

使用开源的 EDA 工具链，每位同学探索完成小规模芯片从 RTL 代码设计到 GDS 的全流程，包括: 架构设计、前端 RTL 设计、综合、后端物理设计(含布局、 时钟树综合、布线、时序分析等)。课程 Project 的 RTL 设计可以是同学自行设计，也可以是任何开源芯片的 RTL 设计代码。

参考自国科大《高级计算机系统结构》课程设计指导文档

《高级计算机系统结构》课程设计 – 一生一芯

使用开源的 EDA 工具链，每位同学探索完成小规模芯片从 RTL 代码设计到 GDS 的全流程，包括: 架构设计、前端 RTL 设计、综合、后端物理设计(含布局、 时钟树综合、布线、时序分析等)。课程 Project 的 RTL 设计可以是同学自行设计，也可以是任何开源芯片的 RTL 设计代码。

蜂鸟E203开源RISC-V处理器
课设中只使用了其中的处理器核部分

参考自国科大《高级计算机系统结构》课程设计指导文档

OpenROAD工具链

方法一：
使用面向CentOS 7的二进制工具链
openroad工具报错

Get the tools

There are currently two options to get OpenROAD tools.

Option 1: download pre-build binaries

We currently support pre-built binaries on CentOS 7. Please, refer to the releases page on GitHub.

Option 2: build from sources

OpenROAD is divided into a number of tools that are orchestrated together to achieve RTL-to-GDS. As of the current implementation, the flow is divided into three stages:

1. Logic Synthesis: is performed by yosys.
2. Floorplanning through Global Routing: are performed by OpenROAD App.
3. Detailed Routing: is performed by TritonRoute.

In order to integrate the flow steps, OpenROAD-flow repository includes the necessary scripts to build and test the flow.

OpenROAD工具链

Get the tools

There are currently two options to get OpenROAD tools.

Option 1: download pre-build binaries

We currently support pre-built binaries on CentOS 7. Please, refer to the releases page on GitHub.

Option 2: build from sources

OpenROAD is divided into a number of tools that are orchestrated together to achieve RTL-to-GDS. As of the current implementation, the flow is divided into three stages:

1. Logic Synthesis: is performed by yosys.
2. Floorplanning through Global Routing: are performed by OpenROAD App.
3. Detailed Routing: is performed by TritonRoute.

In order to integrate the flow steps, OpenROAD-flow repository includes the necessary scripts to build and test the flow.

方法二：
从源代码构建工具链
构建成功可以正常使用

OpenROAD工具链

Get the tools

There are currently two options to get OpenROAD tools.

Option 1: download pre-build binaries

We currently support pre-built binaries on CentOS 7. Please, refer to the releases page on GitHub.

Option 2: build from sources

OpenROAD is divided into a number of tools that are orchestrated together to achieve RTL-to-GDS. As of the current implementation, the flow is divided into three stages:

1. Logic Synthesis: is performed by yosys.
2. Floorplanning through Global Routing: are performed by OpenROAD App.
3. Detailed Routing: is performed by TritonRoute.

- 逻辑综合
- 全局布局布线
- 细节布局布线

In order to integrate the flow steps, OpenROAD-flow repository includes the necessary scripts to build and test the flow.

构建OpenROAD工具链 – 环境

VULTR cloud server

CPU: 双核Intel Xeon处理器

RAM: 4096MB

SSD: 80GB

CentOS Linux release 7.8.2003

Docker version 19.03.8

PLCT实验室提供OpenROAD国内镜像：https://mirror.iscas.ac.cn/plct/OpenROAD-flow.snapshot-20200531.tbz

构建OpenROAD工具链

1. 选择Docker的方式构建，首先开启Docker：

   systemctl start docker

2. 克隆仓库开始构建

   git clone --recursive https://github.com/The-OpenROAD-Project/OpenROAD-flow

3. 修改 IUS 地址： https://repo.ius.io/ius-release-el7.rpm

   修改./tools/OpenROAD/Dockerfile，将文件中的git2u修改为git224

4. 执行脚本对OpenROAD工具链进行构建：

   ./build_openroad.sh

构建OpenROAD工具链

1. 选择Docker的方式构建，首先开启Docker：

   systemctl start docker

2. 克隆仓库开始构建

   git clone --recursive https://github.com/The-OpenROAD-Project/OpenROAD-flow

3. 修改 IUS 地址： https://repo.ius.io/ius-release-el7.rpm

   修改./tools/OpenROAD/Dockerfile，将文件中的git2u修改为git224

4. 执行脚本对OpenROAD工具链进行构建：

   ./build_openroad.sh

构建OpenROAD工具链 – 填坑

执行脚本对OpenROAD工具链进行构建：

./build_openroad.sh ------→ Cannot open:

https://centos7.iuscommunity.org/ius-release.rpm.

原因：

iuscommunity.org的重定向于2020-06-01废止

解决办法：

将Docker文件中的 IUS 地址替换为

https://repo.ius.io/ius-release-el7.rpm

构建OpenROAD工具链 – 填坑

执行脚本对OpenROAD工具链进行构建：

./build_openroad.sh → No package git2u available.

查看CentOS 7 RPM中的git版本：

```
# repoquery --whatprovides git
git-0:1.8.3.1-21.el7_7.x86_64
git-0:1.8.3.1-22.el7_8.x86_64
git222-0:2.22.2-1.el7.ius.x86_64
git222-0:2.22.3-1.el7.ius.x86_64
git224-0:2.24.2-1.el7.ius.x86_64
```

查找哪个Dockerfile中希望安装git2u：

```
# find . -type f -name "*" | xargs grep "git2u"
./tools/OpenROAD/Dockerfile:RUN yum -y remove
git && yum install -y git2u
```

将文件中的 git2u 修改为 git224 即可

构建OpenROAD工具链 – 填坑

IUS的问题

发现已经有人提交issue

以及解决办法

ilesser commented 3 days ago • edited

Describe the bug

Following the *Option 2: Building the tools using docker* fails due to `https://centos7.iuscommunity.org/ius-release.rpm` link being broken.

As stated in this iusrepo issue some links will be decommissioned on 2020-06-01, in particular `https://centos7.iuscommunity.org/ius-release.rpm` which is used in many Dockerfiles.

Expected behavior

Dockerfile should build without errors.

Screenshots

Error after running `./build_openroad.sh`

```
Dependency Updated:
  systemd.x86_64 0:219-73.el7_8.6      systemd-libs.x86_64 0:219-73.el7_8.6

Complete!
Loaded plugins: fastestmirror, ovl
Cannot open: https://centos7.iuscommunity.org/ius-release.rpm. Skipping.
Error: Nothing to do
The command '/bin/sh -c yum group install -y "Development Tools"     && yum install -y https://centos7.iu
scommunity.org/ius-release.rpm     && yum install -y wget centos-release-scl devtoolset-8     devtoolset-
8-libatomic-devel tcl-devel tcl tk libstdc++ tk-devel pcre-devel     python36u python36u-libs python36u-d
evel python36u-pip &&     yum clean -y all &&     rm -rf /var/lib/apt/lists/*' returned a non-zero code:
1
```

Environment:

Please only report issues for supported OSs.

- OS: Linux Mint 19.1 Tessa [CentOS 7 in docker]
- OpenROAD-flow commit [`229a1f4`]

图片来源：https://github.com/The-OpenROAD-Project/OpenROAD-flow/issues/79

构建OpenROAD工具链 – 填坑

同一个issue下

他遇到了git2u的坑

ilesser commented 3 days ago Author

Updating the links fixed the first error but now there is another error since it can't find `git2u` :

```
Complete!
Loaded plugins: fastestmirror, ovl
Determining fastest mirrors
 * base: repo.nixval.com
 * centos-sclo-rh: mirror.airenetworks.es
 * centos-sclo-sclo: mirror.airenetworks.es
 * epel: epel.mirrors.arminco.com
 * extras: mirror.airenetworks.es
 * updates: mirror.airenetworks.es
No package git2u available.
Error: Nothing to do
The command '/bin/sh -c yum -y remove git && yum install -y git2u' returned a non-zero code: 1
```

图片来源：https://github.com/The-OpenROAD-Project/OpenROAD-flow/issues/79

构建OpenROAD工具链 – 填坑

恰好之前遇到过这个坑

回复了解决办法

zhanghb97 commented 1 minute ago

> Updating the links fixed the first error but now there is another error since it can't find `git2u`:
>
> ```
> Complete!
> Loaded plugins: fastestmirror, ovl
> Determining fastest mirrors
> * base: repo.nixval.com
> * centos-sclo-rh: mirror.airenetworks.es
> * centos-sclo-sclo: mirror.airenetworks.es
> * epel: epel.mirrors.arminco.com
> * extras: mirror.airenetworks.es
> * updates: mirror.airenetworks.es
> No package git2u available.
> Error: Nothing to do
> The command '/bin/sh -c yum -y remove git && yum install -y git2u' returned a non-zero code: 1
> ```

You can find the available git version with `repoquery --whatprovides git`.

As for me, I can find the following versions:

```
repoquery --whatprovides git
git-0:1.8.3.1-21.el7_7.x86_64
git-0:1.8.3.1-22.el7_8.x86_64
git222-0:2.22.2-1.el7.ius.x86_64
git222-0:2.22.3-1.el7.ius.x86_64
git224-0:2.24.2-1.el7.ius.x86_64
```

I solve the problem by using git224 (modify the Dockerfile `./tools/OpenROAD/Dockerfile`).

图片来源：https://github.com/The-OpenROAD-Project/OpenROAD-flow/issues/79

构建OpenROAD工具链 – 填坑

6 Dockerfile

```diff
@@ -3,7 +3,7 @@ LABEL maintainer="Abdelrahman Hosny <abdelrahman_hosny@brown.edu>"

  # Install dev and runtime dependencies
  RUN yum group install -y "Development Tools" \
-     && yum install -y https://centos7.iuscommunity.org/ius-release.rpm \
+     && yum install -y https://repo.ius.io/ius-release-el7.rpm \
      && yum install -y centos-release-scl \
      && yum install -y wget devtoolset-8 \
      devtoolset-8-libatomic-devel tcl-devel tcl tk libstdc++ tk-devel pcre-devel \
@@ -28,8 +28,8 @@ RUN wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
      yum install -y epel-release-latest-7.noarch.rpm && rm -rf epel-release-latest-7.noarch.rpm  \
      && yum clean -y all

- # Install git from epel
- RUN yum -y remove git && yum install -y git2u
+ # Install any git version > 2.6.5
+ RUN yum remove -y git* && yum install -y git224

  # Install SWIG
  RUN yum remove -y swig \
```

截至目前，修改仅在openroad分支上生效

使用openroad分支的代码的构建过程相见知乎文章：https://zhuanlan.zhihu.com/p/141713099

进入镜像验证环境

构建成功后，进入 openroad/flow镜像：

```
docker run -it openroad/flow
```

将工具链配置到环境变量：

```
source setup_env.sh
```

测试一下各种工具是否都正常工作：

```
yosys -h
openroad -h
TritonRoute -h
```

进入镜像验证环境

构建成功后，进入 openroad/flow镜像：

docker run -it openroad/flow

Setup environment:

1. `./setup_env.sh` if the build was done using Docker.
2. `./setup_env_bare_metal_build.sh` if the build was done on the bare-metal.

将工具链配置到环境变量：

source setup_env.sh

文档中的./setup_env.sh无法正确配置

此处需要source命令

测试一下各种工具是否都正常工作：

yosys -h

openroad -h

TritonRoute -h

进入镜像验证环境

构建成功后，进入 openroad/flow镜像：

docker run -it openroad/flow

将工具链配置到环境变量：

source setup_env.sh

测试一下各种工具是否都正常工作：

yosys -h

openroad -h

TritonRoute -h → 报错：Error reading param file!!!

是正常现象

蜂鸟E203处理器核代码预处理

在镜像中克隆蜂鸟处理器的仓库：

git clone https://github.com/SI-RISCV/e200_opensource.git

处理器核的verilog代码在/e200_opensource/rtl/e203/core路径下：

cd e200_opensource/rtl/e203/core

为了实现简单，我在源码中注释了处理器核中的SRAM部分

在e203_cpu_top.v文件中注释：

```
/*
e203_srams u_e203_srams(
  .. .. ..
);
 */
```

构建蜂鸟E203处理器核GDS

Adding a New Design

To add a new design, we recommend looking at the included designs for examples of how to set one up.

:warning: Please refer to the known issues and limitations document for information on conditioning your design/files for the flow. We are working to reduce the issues and limitations, but it will take time.

文档对于新手不够友好

添加一个新的设计没有详细的流程指导

需要自己仿照仓库里的例子进行构建 ☹️☹️☹️

构建蜂鸟E203处理器核GDS

1. 编写sdc文件

   新建/OpenROAD-flow/flow/designs/src/e200路径，并在路径下编写e200.sdc约束文件，此处需要注意，get_ports后的时钟名称需要与verilog中的时钟名称一致，蜂鸟E203中的名字为clk：

```
set_units -time ns
create_clock [get_ports clk] -name core_clock -period 5.6
```

构建蜂鸟E203处理器核GDS

2. 编写mk文件

   在/OpenROAD-flow/flow/designs/nangate45路径下编写e200.mk文档，

   此处需要注意把/OpenROAD-flow/e200_opensource/rtl/e203/路径下的core和general

   都写入VERILOG_FILES字段：

```
export DESIGN_NAME = e203_cpu_top
export PLATFORM = nangate45
export VERILOG_FILES = $(wildcard /OpenROAD-flow/e200_opensource/rtl/e203/core/*.v) \
                       $(wildcard /OpenROAD-flow/e200_opensource/rtl/e203/general/*.v)
export SDC_FILE = ./designs/src/e200/e200.sdc
export DIE_AREA = 0 0 2774.76 2398.2
export CORE_AREA = 30.21 29.4 2744.55 2368.8
export CLOCK_PERIOD = 5.600
```

构建蜂鸟E203处理器核GDS

2. 编写mk文件

在/OpenROAD-flow/flow/designs/nangate45路径下编写e200.mk文档，

此处需要注意把/OpenROAD-flow/e200_opensource/rtl/e203/路径下的core和general

都写入VERILOG_FILES字段：

```
export DESIGN_NAME = e203_cpu_top
export PLATFORM = nangate45
export VERILOG_FILES = $(wildcard /OpenROAD-flow/e200_opensource/rtl/e203/core/*.v) \
                       $(wildcard /OpenROAD-flow/e200_opensource/rtl/e203/general/*.v)
export SDC_FILE = ./designs/src/e200/e200.sdc
export DIE_AREA = 0 0 2774.76 2398.2
export CORE_AREA = 30.21 29.4 2744.55 2368.8
export CLOCK_PERIOD = 5.600
```

构建蜂鸟E203处理器核GDS

2. 编写mk文件

在/OpenROAD-flow/flow/designs/nangate45路径下编写e200.mk文档，

此处需要注意把/OpenROAD-flow/e200_opensource/rtl/e203/路径下的core和general

都写入VERILOG_FILES字段：

```
export DESIGN_NAME = e203_cpu_top
export PLATFORM = nangate45
export VERILOG_FILES = $(wildcard /OpenROAD-flow/e200_opensource/rtl/e203/core/*.v) \
                       $(wildcard /OpenROAD-flow/e200_opensource/rtl/e203/general/*.v)
export SDC_FILE = ./designs/src/e200/e200.sdc
export DIE_AREA = 0 0 2774.76 2398.2
export CORE_AREA = 30.21 29.4 2744.55 2368.8
export CLOCK_PERIOD = 5.600
```

构建蜂鸟E203处理器核GDS

2. 编写mk文件

在/OpenROAD-flow/flow/designs/nangate45路径下编写e200.mk文档，

此处需要注意把/OpenROAD-flow/e200_opensource/rtl/e203/路径下的core和general

都写入VERILOG_FILES字段：

```
export DESIGN_NAME = e203_cpu_top
export PLATFORM = nangate45
export VERILOG_FILES = $(wildcard /OpenROAD-flow/e200_opensource/rtl/e203/core/*.v) \
                       $(wildcard /OpenROAD-flow/e200_opensource/rtl/e203/general/*.v)
export SDC_FILE = ./designs/src/e200/e200.sdc
export DIE_AREA = 0 0 2774.76 2398.2
export CORE_AREA = 30.21 29.4 2744.55 2368.8
export CLOCK_PERIOD = 5.600
```

构建蜂鸟E203处理器核GDS

2. 编写mk文件

在/OpenROAD-flow/flow/designs/nangate45路径下编写e200.mk文档，

此处需要注意把/OpenROAD-flow/e200_opensource/rtl/e203/路径下的core和general

都写入VERILOG_FILES字段：

```
export DESIGN_NAME = e203_cpu_top
export PLATFORM = nangate45
export VERILOG_FILES = $(wildcard /OpenROAD-flow/e200_opensource/rtl/e203/core/*.v) \
                       $(wildcard /OpenROAD-flow/e200_opensource/rtl/e203/general/*.v)
export SDC_FILE = ./designs/src/e200/e200.sdc
export DIE_AREA = 0 0 2774.76 2398.2
export CORE_AREA = 30.21 29.4 2744.55 2368.8
export CLOCK_PERIOD = 5.600
```

构建蜂鸟E203处理器核GDS

2. 编写mk文件

在/OpenROAD-flow/flow/designs/nangate45路径下编写e200.mkmakefile文档，此处需要注意把/OpenROAD-flow/e200_opensource/rtl/e203/路径下的core和general都写入VERILOG_FILES字段：

```
export DESIGN_NAME = e203_cpu_top
export PLATFORM = nangate45
export VERILOG_FILES = $(wildcard /OpenROAD-flow/e200_opensource/rtl/e203/core/*.v) \
                       $(wildcard /OpenROAD-flow/e200_opensource/rtl/e203/general/*.v)
export SDC_FILE = ./designs/src/e200/e200.sdc
export DIE_AREA = 0 0 2774.76 2398.2
export CORE_AREA = 30.21 29.4 2744.55 2368.8
export CLOCK_PERIOD = 5.600
```

面积和时钟周期

由用户自己定义

此处数值参考示例进行定义

构建蜂鸟E203处理器核GDS

3. 在Makefile文件中指向e200.mk文件

   在/OpenROAD-flow/flow路径下修改Makefile文件，添加DESIGN_CONFIG字段指向e200.mk文件

```
# E200 design
DESIGN_CONFIG = ./designs/nangate45/e200.mk
```

4. 运行Makefile，构建GDS文件

```
cd /OpenROAD-flow/flow
make
```

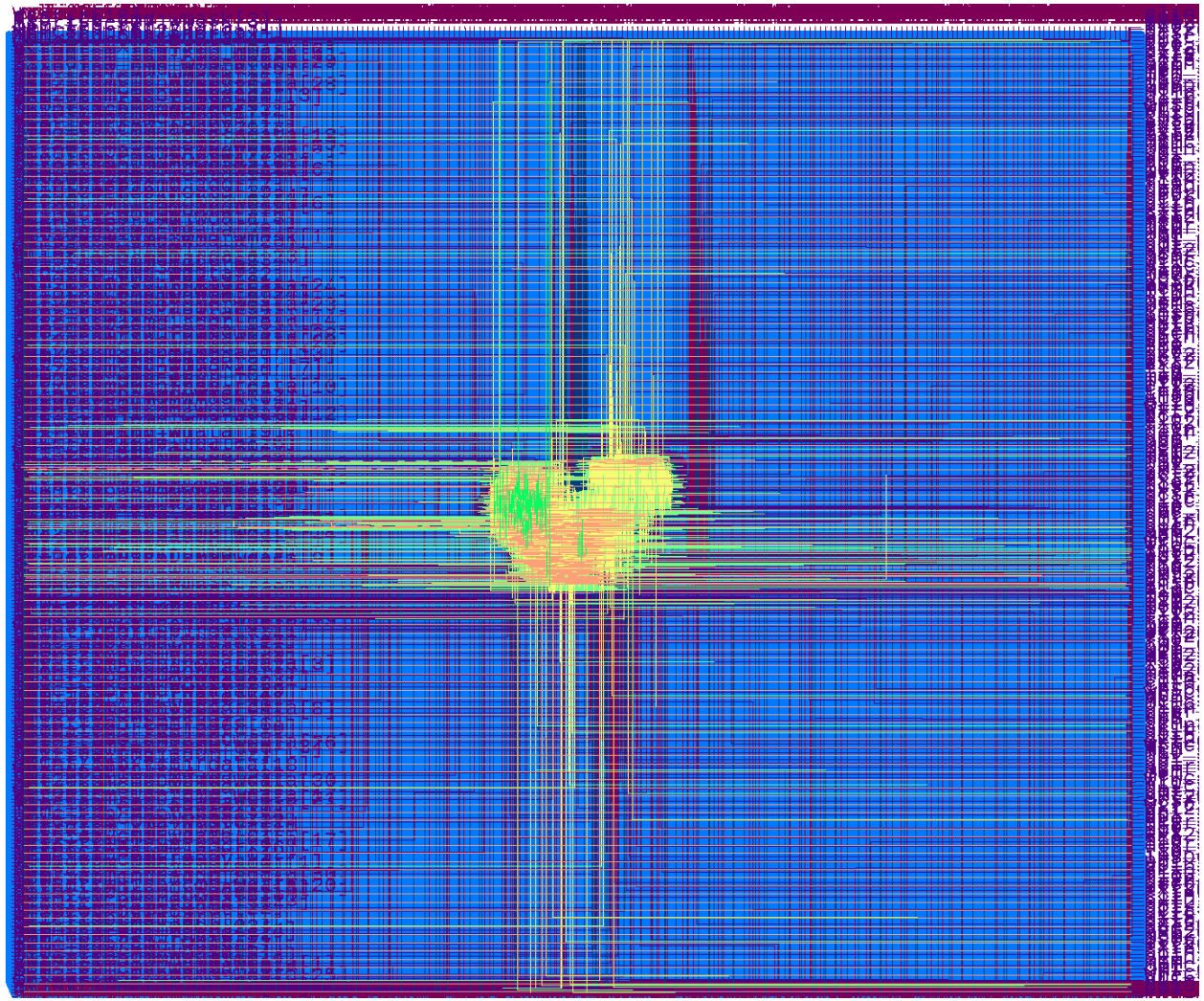
构建好的.gds文件会存放在

/OpenROAD-flow/flow/results/nangate45/e203_cpu_top 路径中

构建蜂鸟E203处理器核GDS

使用Klayout软件可以查看GDS文件

从图中看出，之前配置文件中的面积分配的可能过大了，调整面积以及时钟周期，修改配置文件即可。



国内镜像

PLCT实验室提供OpenROAD镜像：

https://mirror.iscas.ac.cn/plct/OpenROAD-flow.snapshot-20200531.tbz

相关文章

知乎文章：使用OpenROAD构建蜂鸟E203处理器核GDS

https://zhuanlan.zhihu.com/p/141713099

致谢

在做课设、写文章以及制作视频的过程中

感谢任课教师沈老师、解老师提供的指导

感谢PLCT实验室的吴老师、韩同学提供的帮助

感谢国科大的孙同学提供的帮助