Vivado Batch Mode Tool

v1.0

Michael

Vivado Batch Mode 就如其字面意思，就是 Vivado 的批处理模式，用另一种说法也就是使用 shell 的非 GUI（图形界面）交互模式。但是要注意，非 GUI 模式还有一种 tcl Mode，也就是使用 tcl 命令的交互模式，这不等同于我们这里所说的 Batch mode。或者你可以更简单的把它理解为通过命令行与 Vivado 进行交互。

在 Batch mode 下 Vivado 使用起来更加快速，操作更加明确简洁，可以大大的提高工作效率，提高生产力。

为此，为了实验室同学们能更加方便的使用 Vivado Batch mode，我写了这个小工具 Vivado Batch Mode Tool，通过使用它就可以很简单的做到从 GUI 过渡到命令行使用，希望大家都能用上更好更快的工具，在紧张的工作时间里创造更多的价值！

需要一提的是，本工具仅仅是一个简单的脚本，使用 Shell 和 tcl 进行实现（十分简陋且结构简单），主要用于实现我们日常开发的基础功能，这里作为抛砖引玉的作用，希望大家有更好的想法也可以动手去实现，推荐大家学习使用 Python 进行脚本编写，这样更有助于编写更加强大的脚本。

Michael 2019.09.08

修改日期	版本号	修改详情	作者
2019-09-08	v1.0	初版	Michael

目录

一、 为什么使用 Vivado Batch mode

这里从两个方面分别进行说明，为什么要使用 Vivado 的 Batch mode。

1、流程操作效率

我所谓的流程操作效率，是指在使用 Vivado 时，通过键盘输入、鼠标点击以及进行等待等实际外部操作的效率。例如我需要打开一个 Vivado 工程，并进行 Synthesis，那么在 GUI 下是这样的流程操作：点击打开 Vivado GUI 并等待----用 Vivado 点击打开对应的.xpr 文件并等待----点击 run Synthesis 并点击确认----等待----Synthesis 完成。

而如果使用 Batch mode，那么只需要在 Terminal 输入以下命令：

Vivado -mode batch -source syn.tcl XXX.xpr

其中 syn.tcl 是一个提前写好了 tcl 命令用于指示 Vivado 进行 Synthesis 操作的 tcl 文件，这也仅需要几行简单的代码即可。

从这里已经可以看出，在有一个提前准备好的脚本的情况下，通过脚本指挥，自动的去下达命令，可以极大地减少流程操作，可能只需要输入几个字母再按一下回车，就可以执行一套较为复杂的操作。

上面举例中的进行 Synthesis 还算是一个较为简单的操作，很明显在越复杂的操作中，使用 Batch mode 越能带来更高的流程操作效率提升。

一种简单的理解方式是，你提前将需要做的事情都告诉了电脑，然后你就玩去了，电脑按照你提供给他的事务列表逐个逐个自己去进行，这期间不会再来烦你。

更为生动的 GUI 和 Batch mode 在流程操作上的对比就像这样：

GUI 模式：

第一天

妈妈：你快回屋里去。　　我：好的。

妈妈：坐下来。　　我：好的。

妈妈：快写作业。　　我：好的。

我：我能玩电脑吗？　　妈妈：不行。

第二天、第三天……

相同的对话每天重复。

Batch 模式：

第一天

妈妈：你快回屋里去写作业，不准玩电脑。　　我：好的。

第二天、第三天……

妈妈：我之前说的你忘了？快去。　　我：好的。

原本每天重复的长对话后面都只能用一句就代替了。

经过上面的对比，相信你已经对 Batch mode 在流程操作效率上的提升有了明确的认识，那么接下来再说说另一方面。

2、运行效率

运行效率指的就是 Vivado 在执行具体操作的时候其自身运行计算的效率、速度。这里我并没有深究让运行效率提升的具体原因，个人猜测一部分原因是因为不使用 GUI 而释放了很多原先被 GUI 占用的资源，可能还有跟

GUI 占用的系统中断更多有关吧（把这一条归到前一条也没问题），有兴趣的同学可以去具体研究一下，我这里只说对比实验的结果。

我用一台 CPU 为 i7-8700,16G DDR4 2666MHz 内存的 PC，分别使用 GUI 和 Batch mode 对同一个 Vivado 工程从 Synthesis 跑到 Write Bitstream 这样一个完整的过程。

GUI 花了 45 分钟时间，Batch mode 花了 18 分钟。

GUI：“我起了，被秒了，有什么好说的。”

Batch mode 在运行效率上的优势太明显，不需要再说更多了吧。

综合以上两点，使用 Batch mode 既能提高流程操作效率，解放自己的双手，还可以大幅提高运行效率，缩短运行等待时间，直接有效的提高使用 Vivado 的整体效率，使开发速度大大加快。这些，就是我推荐使用 Batch mode 的原因。

二、 要使用 Batch Mode 的一些不便

正如前面提到的 syn.tcl 文件，要指挥 Vivado 执行任何操作，都需要用 tcl 语言来下达命令，也就是说你还需要会用 tcl 语言把要做的事情告诉 Vivado 才行，这就增加了难度。

TCL 语言全称是 Tool Command Language，一般简写成 tcl（读音同 tickle），它是许多的 EDA 工具都使用的一种语言，既有 tcl 原生的语法，也有各家 EDA 自己定义的语法，Vivado 的 tcl 语法可以参照 Xilinx UG835 文档，很全面的手册。或者你也可以参照着平常使用 Vivado 工程时出现的 vivado.jou 这个文件，它里面记录了你这一次从打开到关上 Vivado 整个过程中按顺序每一步使用的 tcl 命令，我个人更偏向于用后者作为主要材料学习，然后随时翻看 UG835 文档作为补充讲解。

另外，除了需要 tcl 语言来写具体的命令，还需要一个框架来进行流程控制，就好像你知道 Synthesis 的命令是什么，也知道 Implement 的命令是什么样，但是你还需要什么时候用什么命令，用完之后再做什么，什么时候停止.....等等这些都要考虑在内。这个框架根据需求可以很简单也可以很复杂，同时它也需要再使用另一种语言来搭建，常见的有 Shell、Perl 以及 Python。这些就又为写一个脚本增加了难度。

因为这些原因，许多人对 Batch mode 望而却步，毕竟都是从 Windows 过来的人，不看图形界面就发慌，关 GUI 是不可能关的，这辈子都不可能关的，写脚本又不会，就只有点来点去才能勉强应付的了生活这样子。

但是，不用担心！

我给你都准备好了，我把一切都放在那里了，One Piece 是.....(不好意思串场了)这是一个一个简陋但是又能实现基本功能的脚本，暂且叫它 Vivado Batch Mode Tool 吧，我尽可能让它的在 Terminal 中能显示出一个类似 GUI 的操作界面。可能你用了之后会觉得“Batch mode 比 GUI 好多了！里面各个东西都实用，界面又简洁，我超喜欢 Batch Mode 的！”

三、 Vivado Batch Mode Tool 介绍与使用方法

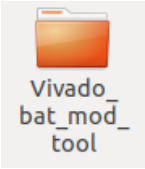
1、Vivado Batch Mode Tool 功能介绍

本工具的初衷是能为本人所在实验室的各位同学们提供一个方便的途径去接触使用 Vivado Batch mode，因此需要方便的与 Vivado 工程配合使用，于是便设计成了类似外置助手工具这样的框架。

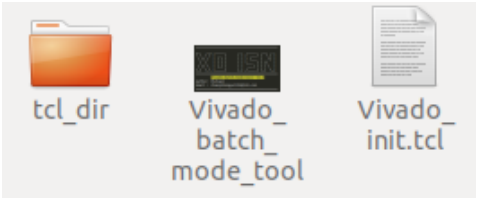
Vivado Batch Mode Tool 可以使用户方便的使用 Vivado 的 Batch mode，仅需要输入提供的选项就可以做到基础的 Synthesis、Implement、Write Bitstream 以及 Program Device 这些操作，相当于为用户准备了一个不同于 GUI 的操作界面，可以较好的服务于习惯 GUI 的用户。本工具仅能实现一些基础必要的功能，一些更为复杂的特别是需要用户自己定义的（例如 Debug Core）仍然需要通过 GUI 完成，在工具中也提供了进入 GUI 的选项。

2、Vivado Batch Mode Tool 文件总览

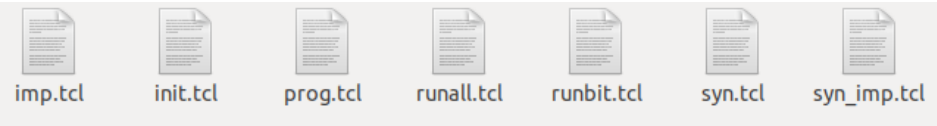
Vivado Batch Mode Tool 我将其放入一个文件夹内，如下图。



其中包含了这样的一个文件夹和两个文件，如下图。这其中 Vivado_init.tcl 的作用我已经写在它的注释里了，不希望每一次打开 vivado 都产生两个备份的.log 和.jou 的同学可以看一看它。



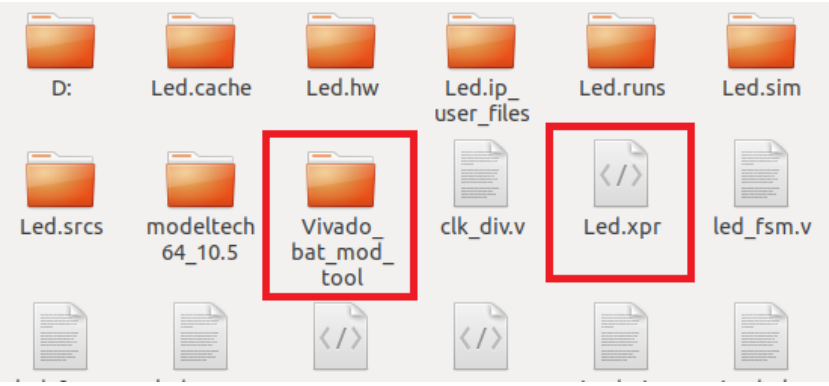
在 tcl_dir 内有 7 个.tcl 文件,如下图。



这些就是这个脚本的全貌。

3、Vivado Batch Mode Tool 使用介绍

1）将 Vivado_bat_mod_tool 文件夹放置在与你的 Vivado 工程的.xpr 同一级目录下，如下图。

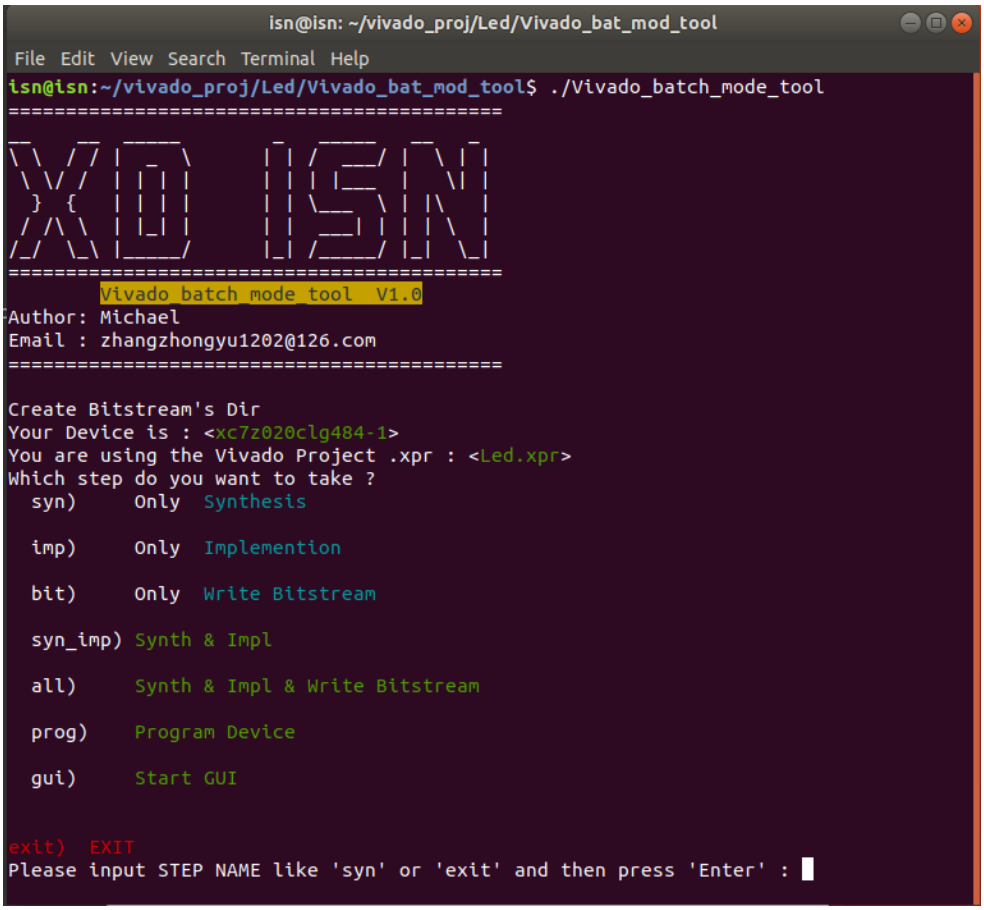


2）修改文件夹内的 Vivado_batch_mode_tool 这个文件的权限为允许作为可执行文件，可以使用 chmod 命令，也可以右键----Properties----Permission----下方勾选 Allow executing file as program。

3）在 Vivado_bat_mod_tool 文件夹这一级使用 Terminal，运行 Vivado_batch_mode_tool：

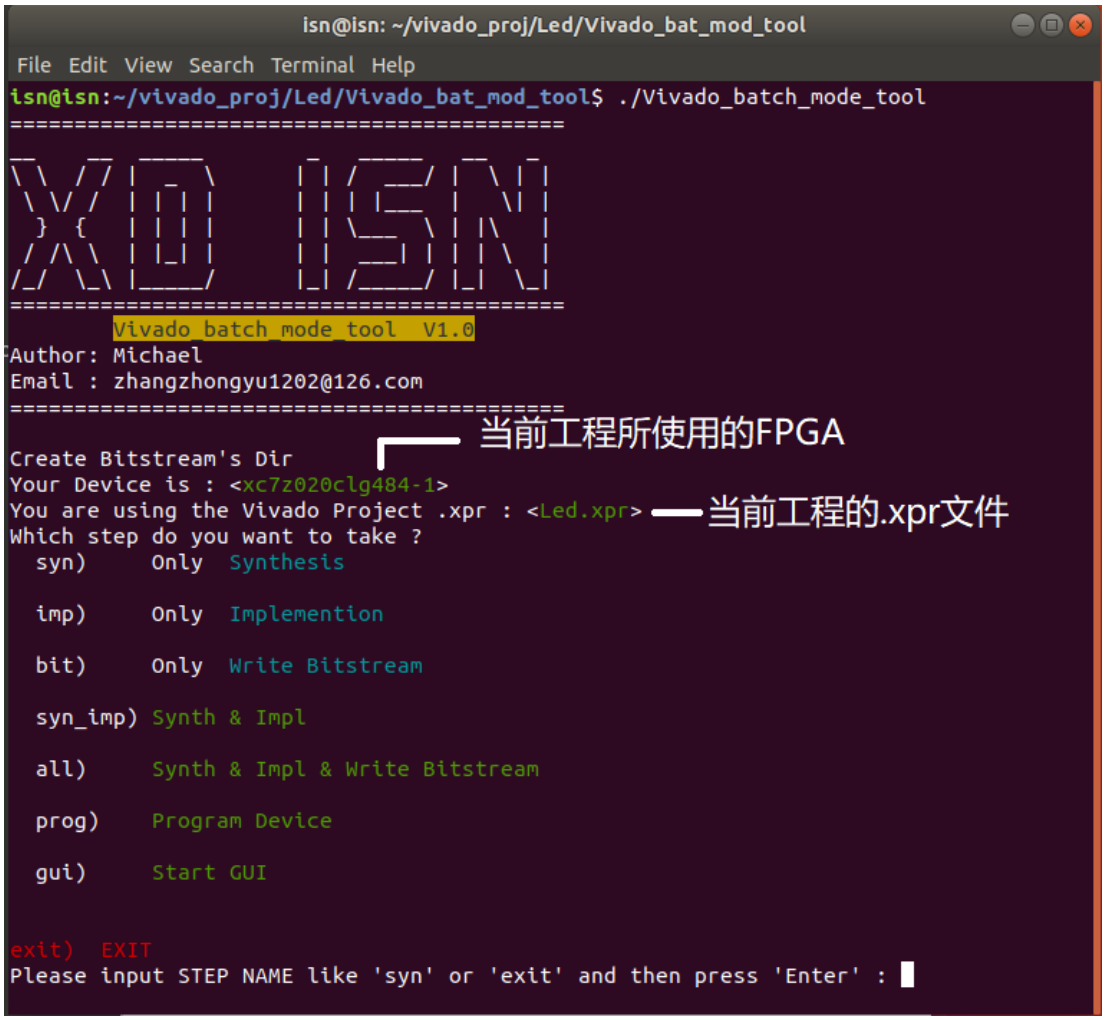
$ ./Vivado_batch_mode_tool

4）如果文件夹放置正确，则会检测到 Vivado 工程，显示如下的界面：



如果文件夹放置不正确，则会报错，如下图：

5）下面介绍各个选项的使用，请根据需要，输入相应的选项并回车即可。



syn：仅执行 Synthesis 操作，执行完会询问是否需要打开 Synthesis Design 的 GUI 界面，请输入 y/n 来决定是否打开。这里是为了当一些工程需要添加 debug core 的时候，可以在综合完成后，进入 GUI 去 set Debug，设置完成后再手动关闭 GUI 即可。

imp：仅执行 Implement 操作，执行完会询问是否需要打开 Implemention Design 的 GUI 界面，请输入 y/n 来决定是否打开。这里是为了有时需要在实现完成后检查布线图。

bit：仅执行 Write Bitstream 操作，会将生成的.bit 文件和.ltx 文件（如果有的话）放置在 Vivado_bat_mod_tool 文件夹中的 bitstream 文件夹内。

syn_imp：连续执行 Synthesis 和 Implement，执行完会询问是否需要打开 Implemention Design 的 GUI 界面，请输入 y/n 来决定是否打开。

all：连续执行 Synthesis、Implement 和 Write Bitstream。

prog：将 bitstream 内的[current_project].bit 下载入 FPGA。如果没有发现匹配的 Device 则会报错。注意，目前此脚本还仅能下载.bit 文件，如果有.ltx 文件请进入 GUI 去下载，毕竟抓信号有时候还是需要用 GUI 观察。

gui:使用 GUI 打开这个 Vivado 工程。

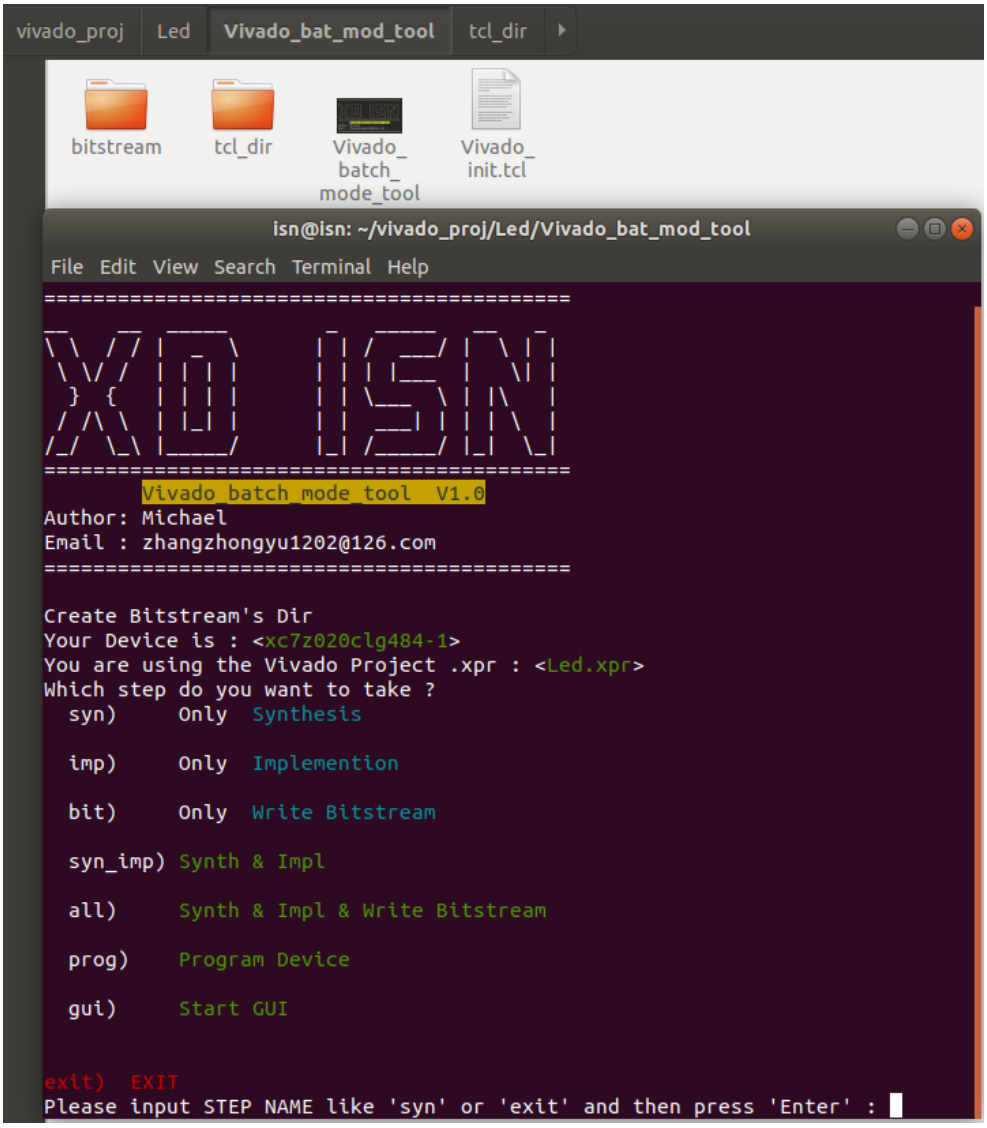
exit：退出脚本。

6）每一次执行完之后只需要按回车就可以返回主界面，不要反复开关脚本。

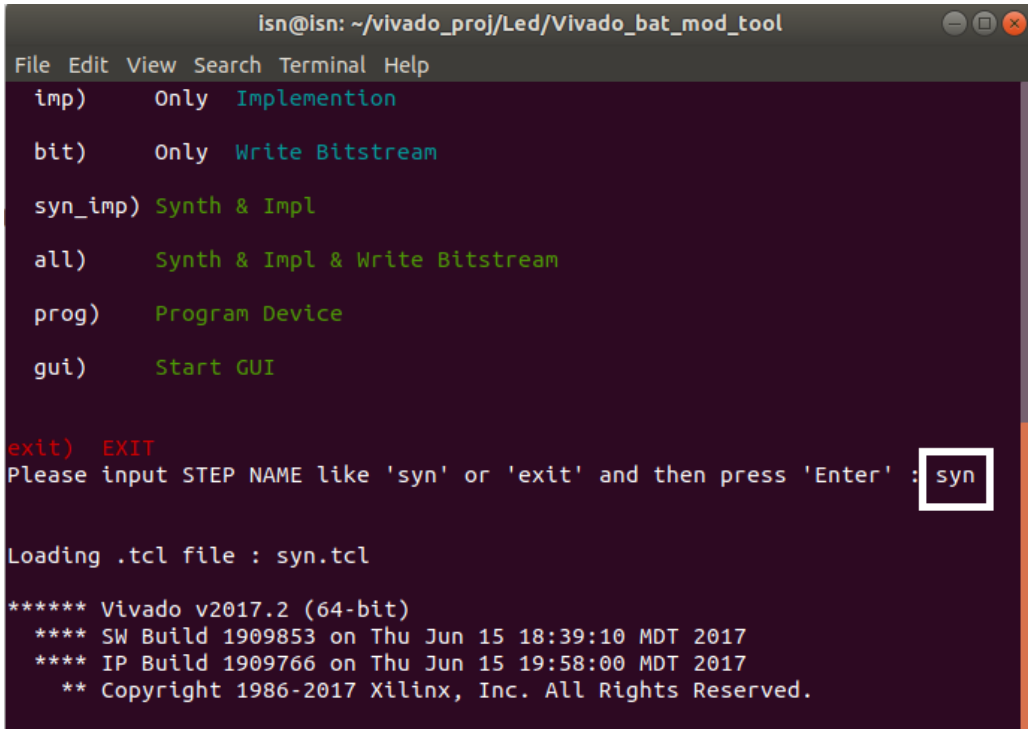
四、 Vivado Batch Mode Tool 使用实例

接下来，我将通过讲解一次完整的从 Synthesis 到 Program Device 的操作来为大家提供一个使用这个脚本的实例。

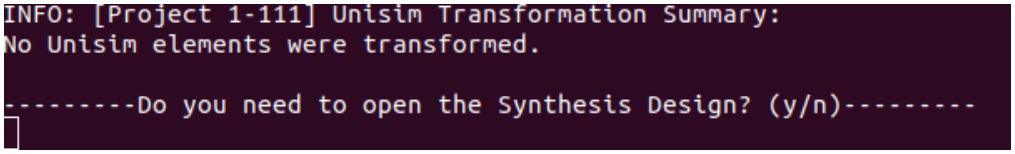
1）如前文描述一样放置正确路径并打开脚本，得到下图：



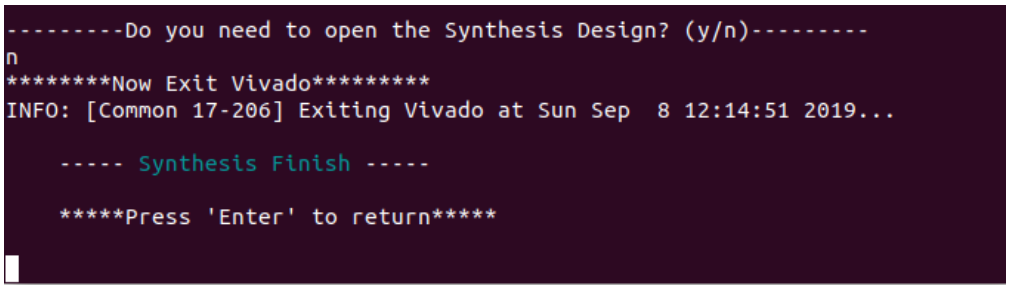
2）单独执行 Synthesis（当然也可以一步到位执行 all），输入 syn 并回车得到下图。



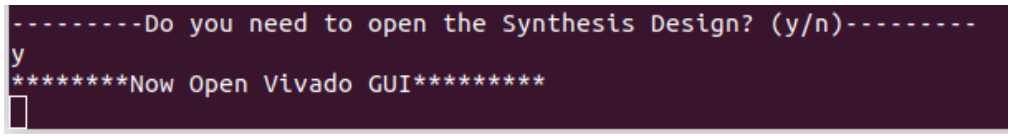
然后 Vivado 就会开始对你当前的工程进行 Synthesis，完成之后会有询问是否要打开 Synthesis Design，如下图。



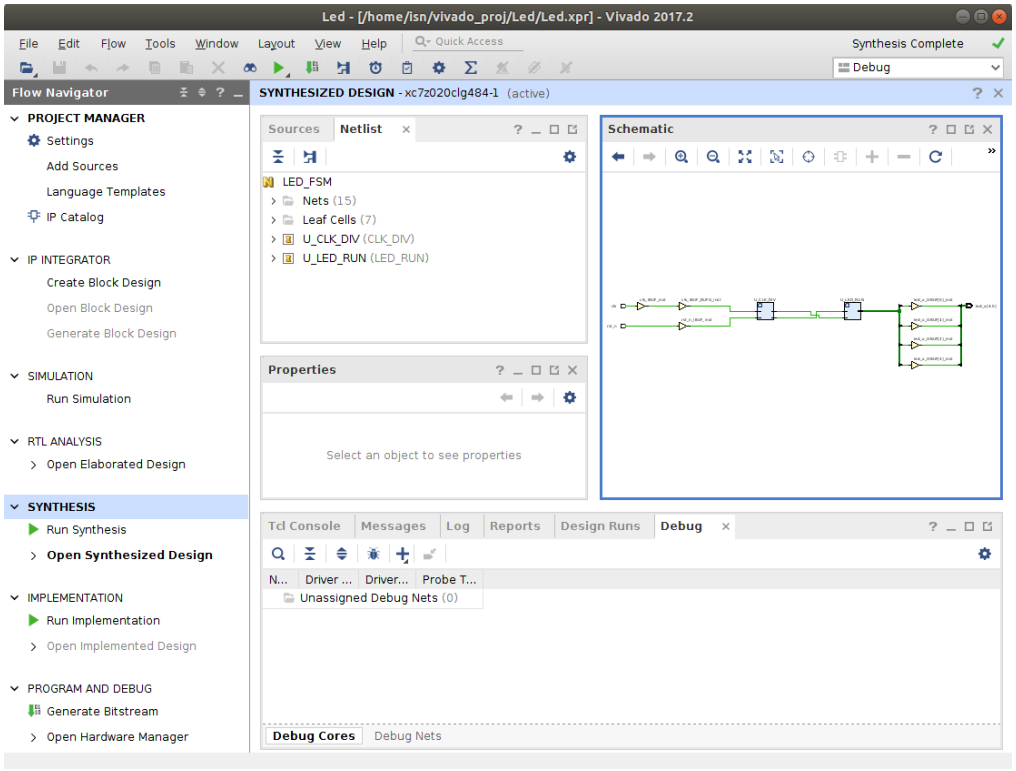
不需要就输入 n 然后回车，如下图。



需要打开 Synthesis Design 则输入 y 并回车，然后就会打开 Synthesis Design 的 GUI。



下图是打开的 Synthesis Design GUI 界面。



之后关掉 GUI 时也会显示

```
----- Synthesis Finish -----

*****Press 'Enter' to return*****
```

只需要按回车就可以返回主界面。

3）直接执行 all，从 Synthesis 跑到 Write Bitstream，输入 all 并回车，如下图。

```
Your Device is : <xc7z020clg484-1>
You are using the Vivado Project .xpr : <Led.xpr>
Which step do you want to take ?
  syn)      Only  Synthesis

  imp)      Only  Implemention

  bit)      Only  Write Bitstream

  syn_imp) Synth & Impl

  all)      Synth & Impl & Write Bitstream

  prog)     Program Device

  gui)      Start GUI


exit)  EXIT
Please input STEP NAME like 'syn' or 'exit' and then press 'Enter' : all


Loading .tcl file : runall.tcl

****** Vivado v2017.2 (64-bit)
  **** SW Build 1909853 on Thu Jun 15 18:39:10 MDT 2017
  **** IP Build 1909766 on Thu Jun 15 19:58:00 MDT 2017
    ** Copyright 1986-2017 Xilinx, Inc. All Rights Reserved.
```

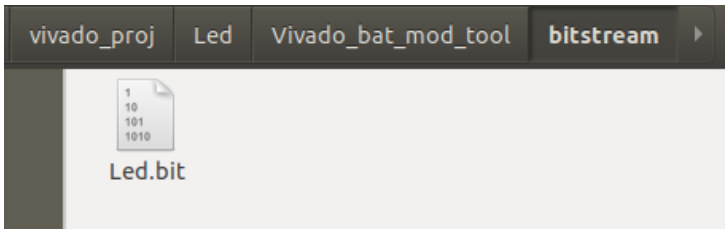
然后 Vivado 就会开始自动的先执行 Synthesis，再执行 Implemention，最后执行 Write Bitstream。完成后会有如下图的提示。

```
INFO: [Common 17-206] Exiting Vivado at Sun Sep  8 12:19:47 2019...

    ----- All Steps Finish -----

    *****Press 'Enter' to return*****
```

同时 bitstream 文件夹下会出现.bit 文件，有 debug core 时还会出现.ltx 文件。



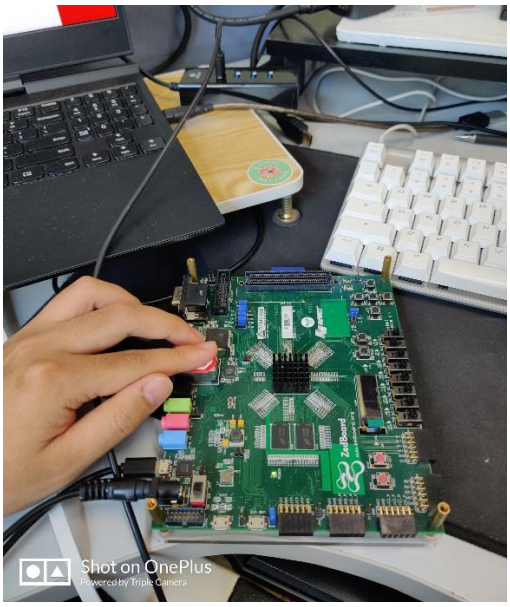
4）最后我要进行 Program Device 操作

此时我的 PC 上没有连接任何 FPGA 板，如果此时执行 prog，则会出现下图的报错。

```
ERROR: [Labtoolstcl 44-199] No matching targets found on connected servers:
 localhost
Resolution: If needed connect the desired target to a server and use comman
d refresh_hw_server. Then rerun the get_hw_targets command.
ERROR: [Common 17-39] 'get_hw_targets' failed due to earlier errors.

    while executing
"get_hw_targets"
    invoked from within
"foreach { hw_target } [get_hw_targets] {
    current_hw_target $hw_target
    open_hw_target
    foreach { hw_device } [get_hw_devices] {
    if { [st..."
    (file "./tcl_dir/prog.tcl" line 26)
INFO: [Common 17-206] Exiting Vivado at Sun Sep  8 12:24:13 2019...
    *****Press 'Enter' to return*****
```

我现在就去拿块板子插上。



好的板子插上了，我们现在再来 prog 一下。

```
Your Device is : <xc7z020clg484-1>
You are using the Vivado Project .xpr : <Led.xpr>
Which step do you want to take ?
  syn)      Only  Synthesis

  imp)      Only  Implemention

  bit)      Only  Write Bitstream

  syn_imp) Synth & Impl

  all)      Synth & Impl & Write Bitstream

  prog)     Program Device

  gui)      Start GUI


exit)  EXIT
Please input STEP NAME like 'syn' or 'exit' and then press 'Enter' : prog


Loading .tcl file : prog.tcl

****** Vivado v2017.2 (64-bit)
  **** SW Build 1909853 on Thu Jun 15 18:39:10 MDT 2017
  **** IP Build 1909766 on Thu Jun 15 19:58:00 MDT 2017
    ** Copyright 1986-2017 Xilinx, Inc. All Rights Reserved.
```

Program 完成后会有提示，如下图。

```
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digi
lent/210248585150
------Successfully Found Hardware Target with a xc7z020clg484-1 device-----
-
INFO: [Labtools 27-3164] End of startup status: HIGH
INFO: [Common 17-206] Exiting Vivado at Sun Sep  8 12:31:22 2019...
    *****Press 'Enter' to return*****
```

并且此时开发板上是这样的

没错它是个流水灯。

全程完成，操作少，速度快，易掌握。

五、 写在最后

希望看完了以上内容的你能够掌握这种 Vivado 的 Batch mode 用法，并能够去开发自己的脚本工具。

希望本工具能够起到抛砖引玉的作用，带给以前没有接触过使用脚本进行开发的同学们一个新的体验，然后打开一个新的大门，提高更多的能力。

目前这个 Vivado Batch Mode Tool V1.0 也是我关于 Vivado Batch mode 的第一个脚本，比较简陋的 shell 脚本，也希望后面我也加强学习，等到 V2.0 的时候给大家带来更功能强大的 Python 版本的脚本。

Michael 2019.09.08