第十一章 谁说 Git 只管软件？PCB 也不在话下

广大硬件工程师们你们是如何管理 PCB 工程的版本的呢？以我这么多年的经历所看到的，很多工程师和公司更在乎的是 PCB 及相关资料的留存和安全，在版本管理上似乎并不太在意，或者说没有什么好的办法。关于硬件资料的版本，更多是依赖于工程师自身的素养。比如统一整理到某种存储介质中，可以是一台设置了权限的共享计算机，或者一个 NAS 服务器，也可以是一个类似于 seafile 的网盘。而对于这些资料的追溯，则要靠完整的技术文档。这样的管理方式，看似严谨，硬件资料都在公司的掌控之中，但实际上最终难免落得个资料凌乱，版本混乱的下场。很多时候，你面对着众多的 PCB 版本，并不能确定哪一个是最终可用于生产的版本，因为往往并不是最新的版本就是可用的版本。OK，那我们向硬件引入 Git 这一版本管理利器，来看看通过它如何来管理硬件版本。

1、基于 Git 的研发管理体系

本章主要是讲如何使用 Git 对硬件相关资料进行版本管理。其实本质上，Git 只不过是一个工具，真正要把研发资料管理好，其实这背后有更深层的管理策略。这就像我们拥有一把雕刻刀，它锋利而顺手，但是用它能不能创造出伟大的作品，还是要看人的素质。所以，任何事物人才是决定性的因素。振南在基于 Git 的研发管理体系上有一些经验和拙见，拿出来与大家共勉，如图 11.1。

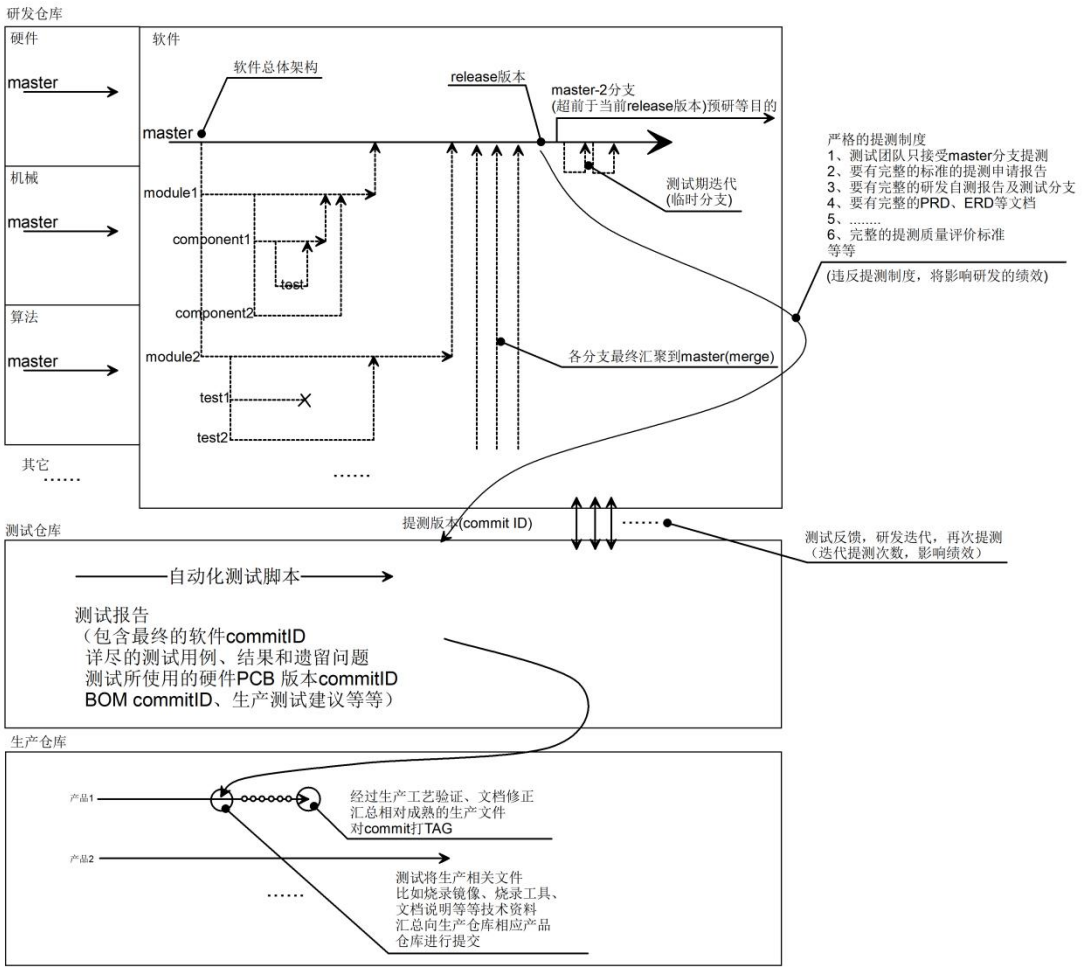

图 11.1 基于 Git 的研发管理体系

我在清华创业公司就职的几年里（2019~2022 年），作为大硬件部门的主管参与了大量研发流程和制度的制定，主要的目的就是为了约束整个硬件研发的规范，以及资料的良性迭代、固化和可追溯。CTO 因为是软件出身，同时又有大型软件公司的任职经历，所以他非常推崇 Git，并希望 Git 能够贯穿于整个研发管理之中。

2、基于 Git 的 PCB 版本管理

OK，上面振南点了一下研发管理体系的问题，但这不是本章的重点。我们把目光集中到 PCB 版本管理上来。

2.1 Git 的增量式管理

Git 对于文件的管理采用的是增量式管理，如图 11.2 所示。

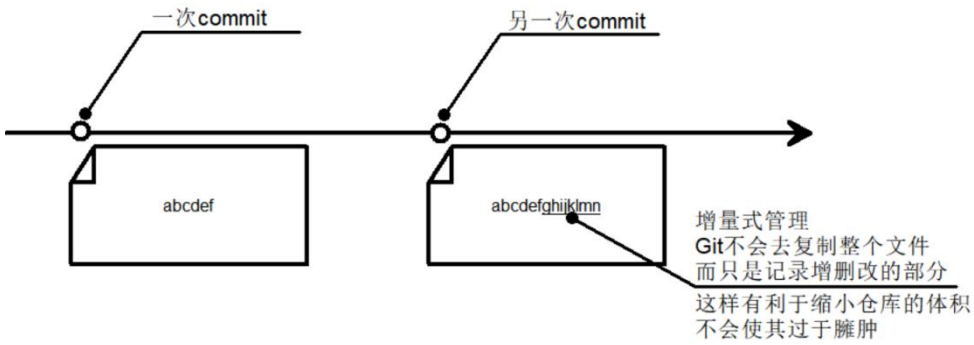

图 11.2 Git 的增量式文件管理

Git 在文本管理上的优势是显而易见的。但是像 PCB、Word、PDF 等这些文件，都是二进制的，不同版本的文件相同之处比较少，此时再用增量式管理，基本无异于文件拷贝。所以，用 Git 管理二进制文件，会让仓库体积比较大。而且，二进制文件是没法直接合并的。不过，这些并不妨碍我们使用 Git 来进行管理。因为 Git 针对于二进制或者大文件有自己的一套管理方法（比如压缩或者 LFS，大家可以去百度一下）。

2.2 AD 中的 Git

鉴于 Git 的广泛应用，很多开发环境都已经与 Git 无缝衔接，而并不需要涉及太多的命令行操作。AD（Altium Designer）作为最著名的 EDA 工具之一，自然是支持 Git 的。

2.2.1 本地化操作

振南以一个 PCB 工程为例来进行介绍，如图 11.3。

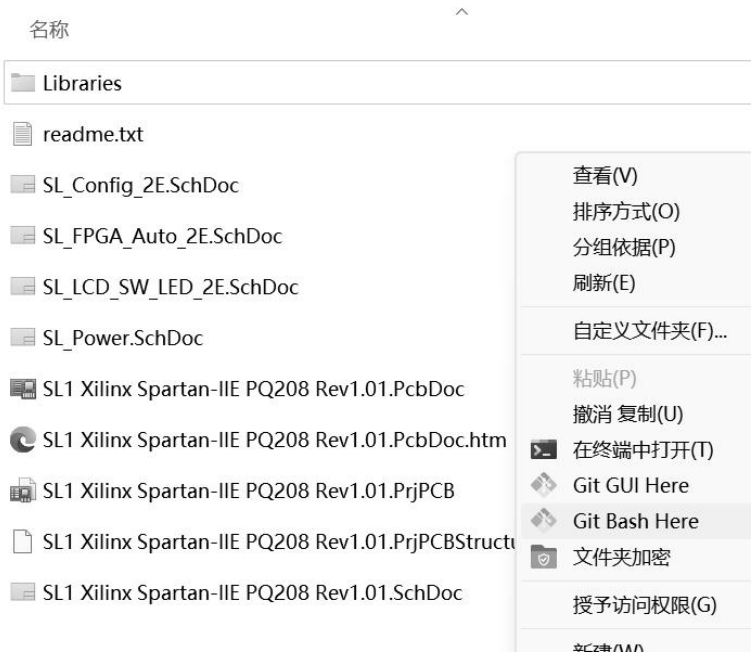

图 11.3 在工程目录下右键打开 Git Bash

首先创建一个本地仓库 git init，如图 11.4。

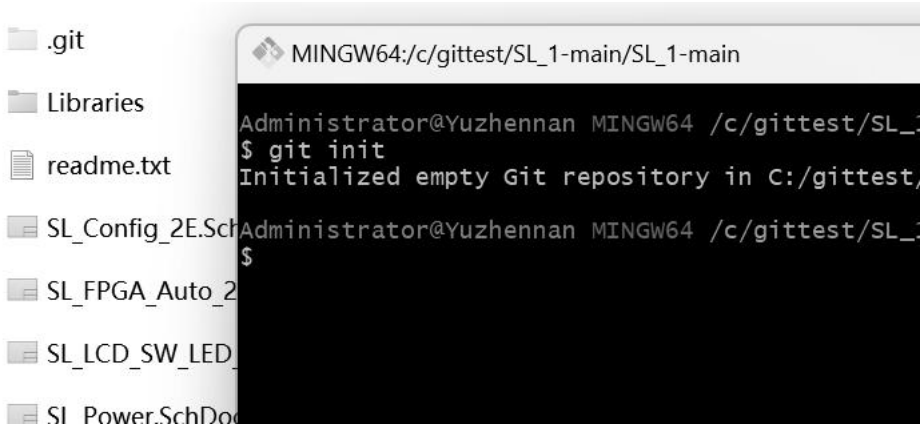

图 11.4 创建一个本地仓库

OK，命令行就到这里。接下来我们的操作基本上都在 AD 软件中来完成。右键 PCB 工程文件，将其添加到仓库中来。如图 11.5 所示。

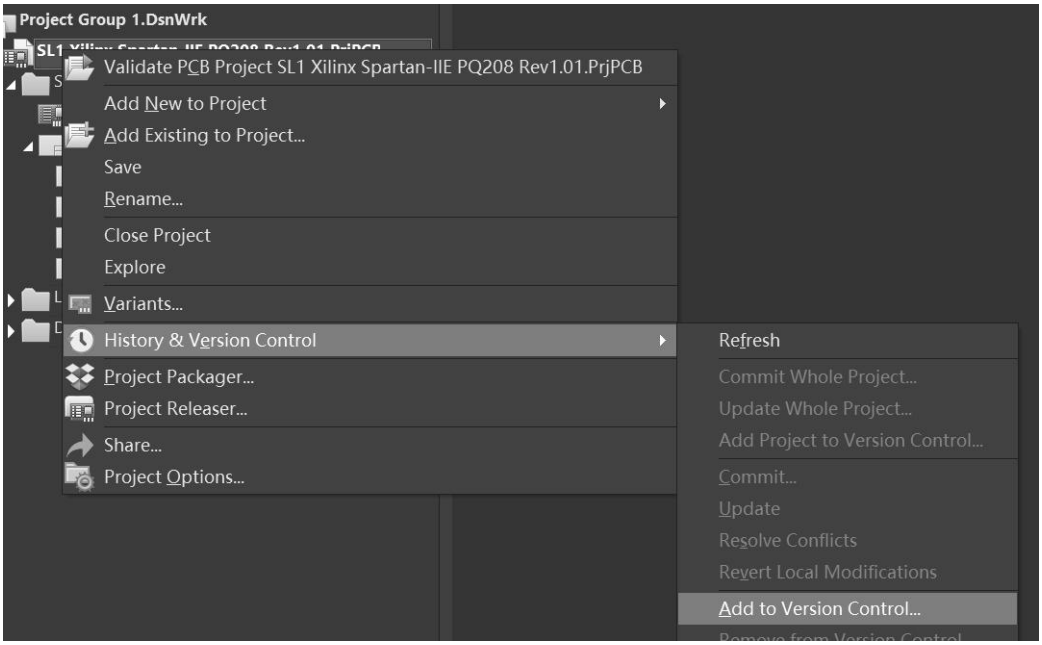

图 11.5 将 PCB 工程文件添加到仓库之中

接下来，我们尝试将整个 PCB 工程进行提交。如图 11.6。

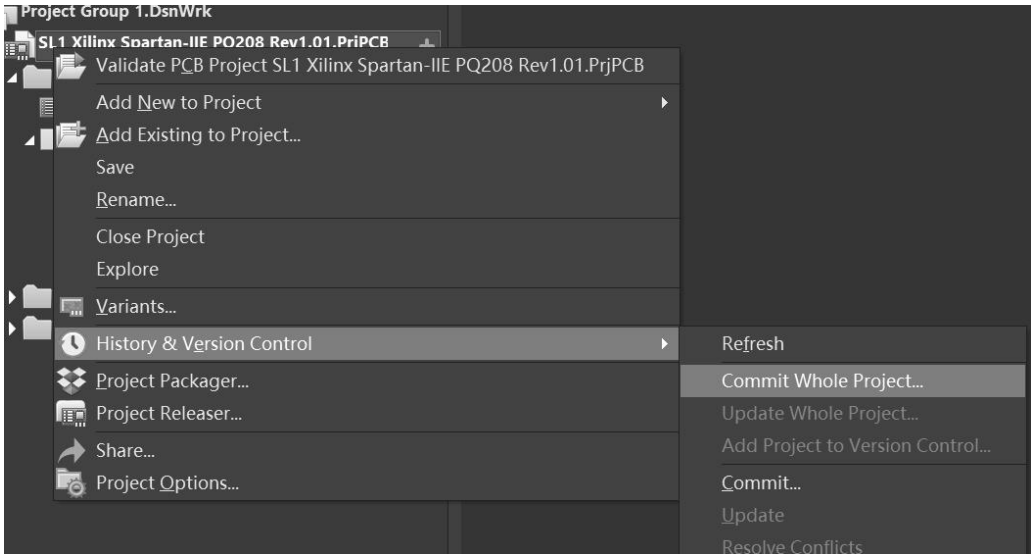

图 11.6 将整个 PCB 工程进行提交

在 Comment 中可以填入关于此次提交的一些说明，这类似于 git commit -m。如图 11.7。

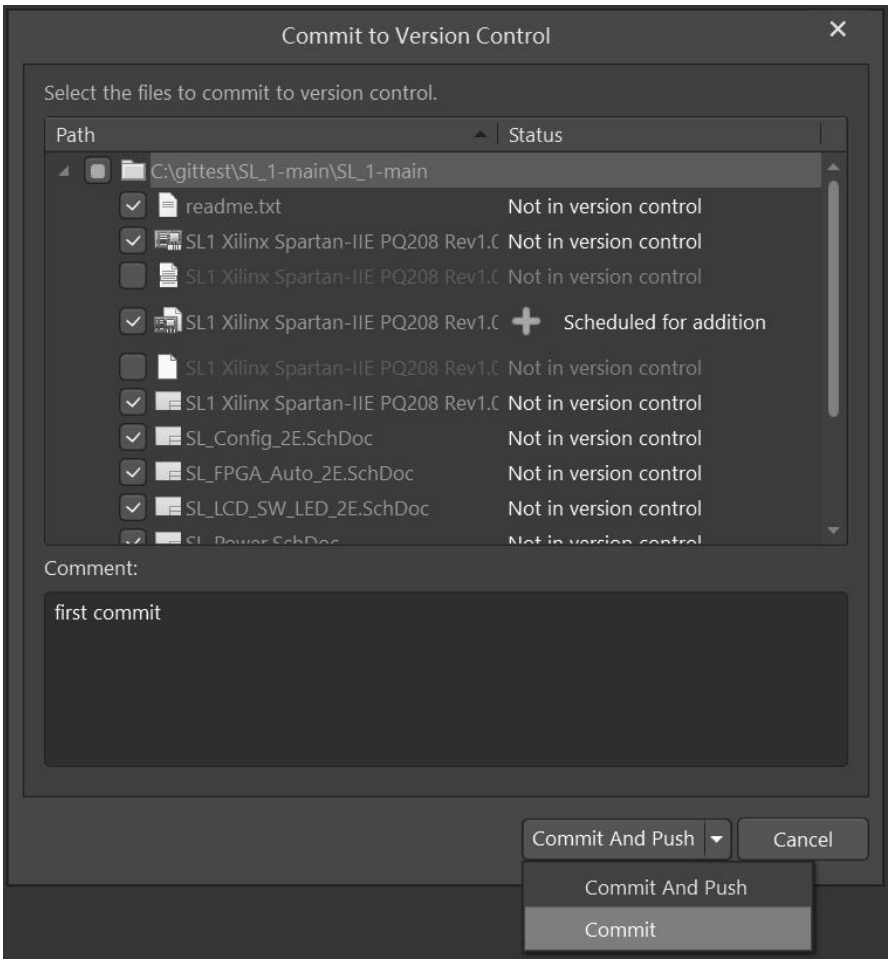

图 11.7 填写 Comment 后进行提交

随后我们可以看到工程视图中的文件右边出现了图标，如图 11.8 所示。

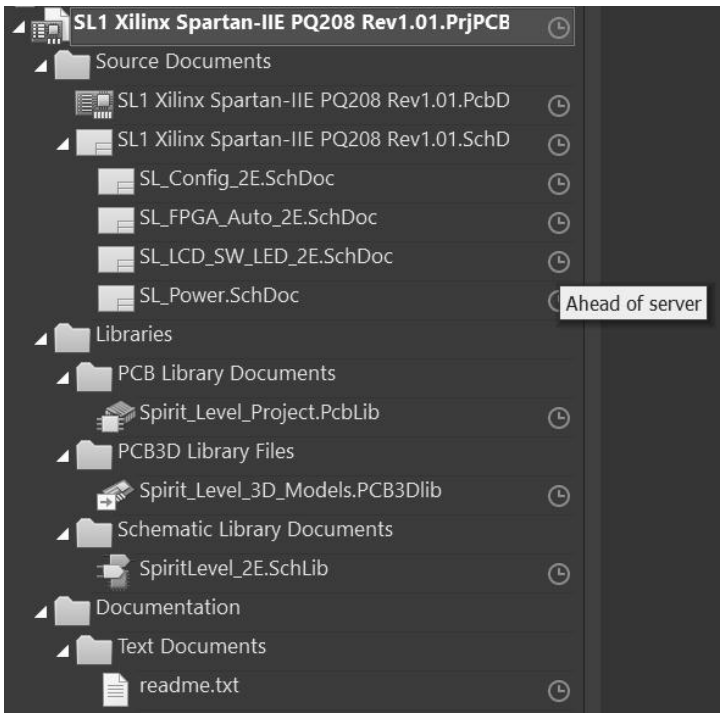

图 11.8 提交之后 PCB 工程相关文件右边出现了图标

这种图标的意思是“Ahead of server”，即等待推到远端仓库。这是因为刚才只进行了 commit 而没有 push，等后面我们设置了远端仓库的链接，就可以执行 push 了。

我们尝试对某个文件进行修改，如图 11.9。

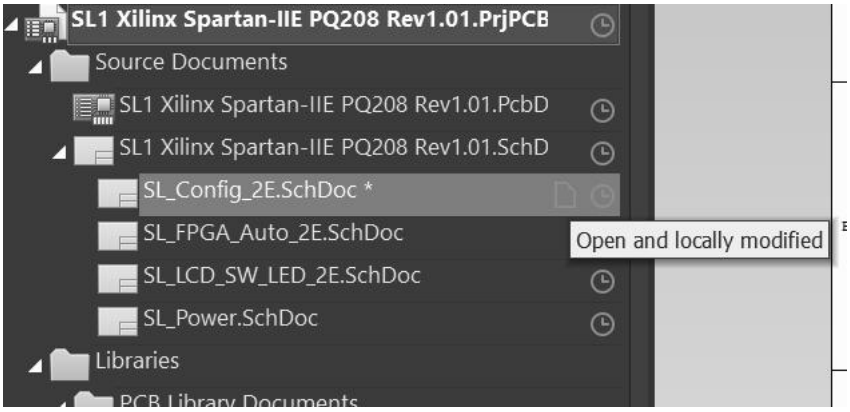

图 11.9 对某个原理图文件进行修改

此时这个文件右边将出现一个新的图标“Open and locally modified”，对其再一次进行提交，如图 11.10。

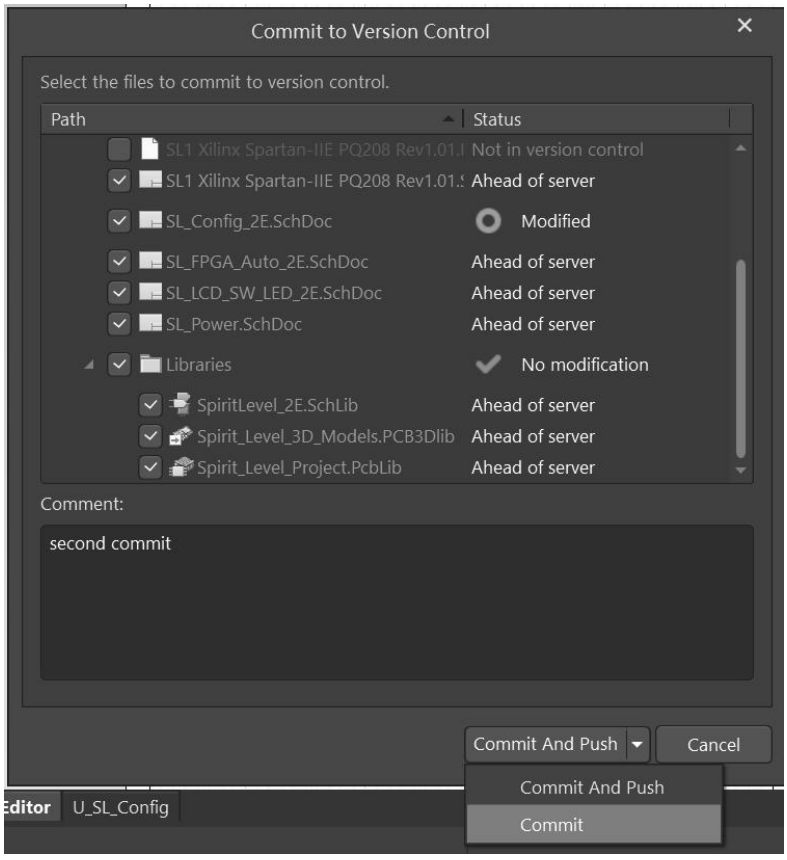

图 11.10 对 PCB 工程再一次进行提交

在 AD 中的这些操作，与 Git 是完全同步的，其实 AD 就是在背后调用 Git。我们可以用 git log 查看一下。如图 11.11。

```
Administrator@Yuzhennan MINGW64 /c/gittest/SL_1-main/SL_1-main (master)
$ git log
commit 9caa0a92d46b56e6dfe92fcb89289c41ef2dd3dd (HEAD -> master)
Author: yuzhennan <987582714@qq.com>
Date:   Sun Nov 27 17:20:49 2022 +0800

    second commit

commit 69e794d286a1b0cf1a44aa132139ed92abfbd265
Author: yuzhennan <987582714@qq.com>
Date:   Sun Nov 27 16:58:29 2022 +0800

    first commit

Administrator@Yuzhennan MINGW64 /c/gittest/SL_1-main/SL_1-main (master)
$
```

图 11.11 通过 git log 查看提交记录

2.2.2 远端操作

接下来，我们将本地仓库推到远端服务器。首先，先要在远端服务器上建立一个仓库（GitHub 上创建仓库前面章节已有介绍，这里不再赘述）。使用 Git 命令行向仓库添加远端仓库的链接，如图 11.12。

```
Administrator@Yuzhennan MINGW64 /c/gittest/SL_1-main/SL_1-main (master)
$ git remote add origin https://github.com/ZNelec/SL.git

Administrator@Yuzhennan MINGW64 /c/gittest/SL_1-main/SL_1-main (master)
$
```

图 11.11 通过 git log 查看提交记录

这里我们仍然使用 HTTPS 链接，一方面是方便，省去了生成 SSH-key 的麻烦；另一方面是 AD 的 CVS 系统目前只支持 HTTPS。随后，右键 push 即可将仓库推到 GitHub 上了。这个过程中，可能会提示输入用户名密码，输入即可。

2.3 PCB 工程的协作开发

我们使用 Git 一方面是为了管理版本，另一方面是让开发工作实现并行化和更好的团队多人协作。Git 解决了很多团队协作过程中经常出现的问题，遇到最多的就是多人同时开发同一分支。如图 11.12。

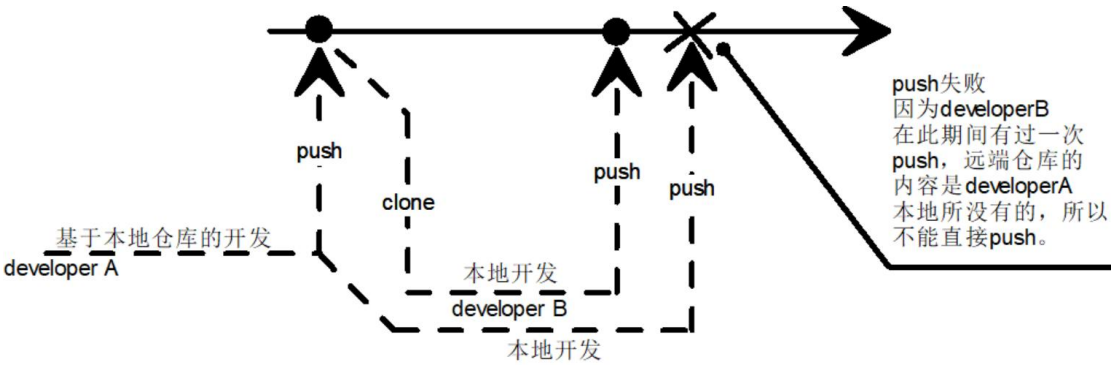

图 11.12 多人同时开发同一分支时 push 失败

解决的方法就是 A 应该先 git pull，将远端仓库同步到本地，并在本地完成合并和提交，然后再 push。具体过程，如图 11.13。

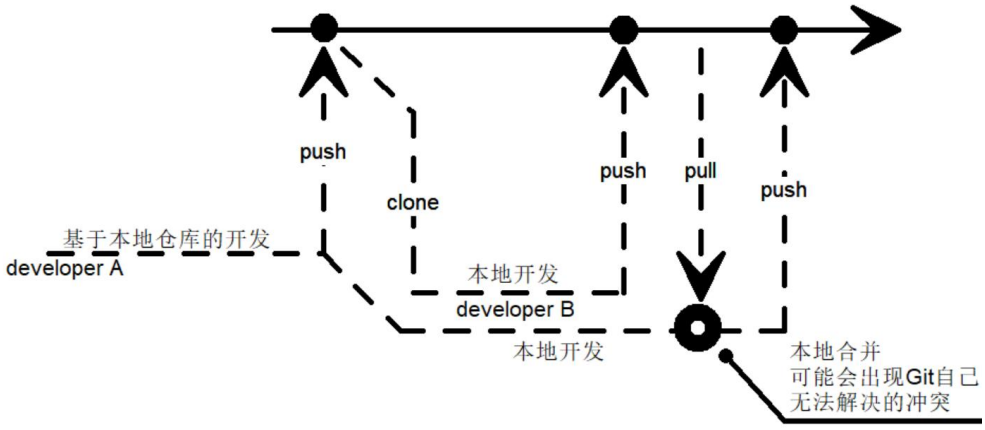

图 11.13 多人同时开发同一分支时 push 失败的解决方法

值得注意的是，在 pull 之后在本地进行合并时，是有可能出现无法自动解决的冲突的，这就需要我们手动来进行解决。根本问题在于，为什么会出现无法自动解决的冲突？这种情况绝大多数都是因为多人编辑了同一个文件，并在文件相同的位置上存在不同的内容，造成 Git 无法决定舍谁留谁。这种情况，一般需要参与开发的工程师共同商量，以人工方式来解决冲突。

如果多人在同时设计一张电路原理图或者 PCB 版图，那是否可以合并呢？答案是否定的，因为 PCB 文件是二进制的，而非字符。对于二进制的合并，将毫无意义。