数字逻辑项目文档：GenshinKitchen

黄政东
祝超
何俞均

1 团队分工

1.1 团队分工

- 贡献比 1:1:1
- 详细工作安排见下表。

黄政东	project 顶层模块的处理（manualTop, ScriptTop 以及 DemoTop）
	实现接收并用 LED 显示来自客户端的四个反馈信号
	所有模块的代码规范性检查
	实现在手动模式/自动 (脚本) 模式之间进行切换
	实现脚本单步调试
	部分文档攥写工作
祝超	project 手动模式的合法性检查
	project 脚本模式中 jump，wait 语句相关模块执行的处理
	各种需要的测试脚本的准备，高效的脚本设计
	loadingLamp 模块的编写
	部分文档攥写工作
何俞均	project 手动模式的按键到机器码部分（buttonDecoder 模块）
	project 脚本模式中 Action、Game State Instruction 部分的处理
	使用七段数码管显示信息
	错误脚本状态自动处理
	部分文档攥写工作

表 1: 详细工作安排

1.2 开发计划日程安排和实施情况

开发计划安排

- 11 周讨论 project 实现方式，大致模块安排以及粗略设计实现方式。

- 12 周完成 manual 模块的所有子模块，一人进行 manualTop 模块的编写及调试，另外两人进行脚本模块的编写。
- 13 周脚本模块编写完成，一人进行 scriptTop 模块的编写及调试，一人负责准备脚本，另一人负责完成一些 bonus 中的任务。
- 14 周基本完成所有模块，解决一些发现的 bug，并对一些对用户不太友好的操作进行更加人性化的调整。
- 15 周测试所有模块，继续解决遇到的 bug，准备答辩。
- 15 周答辩。

实际开发情况

- 11.30: 开会讨论设计大致模块。决定设计 DesignedTop，ManualTop，ScriptTop，TargetRegister。并先完成 DesignedTop，ManualTop，TargetRegister。
- 12.2: DesignedTop，TargetRegister 完成。设计 ManualTop 中包含的 TargetStateMachine，TargetStateEncoder，GameStateEncoder，OperationEncoder，ManualFliter。
- 12.10: ManualTop 完成设计 ScriptTop 中的 GameStateScriptHandler，ActionScriptHandler，JumpScriptHandler，WaitScriptHandler。
- 12.17: ScriptTop 完成准备 Bonus 中的 ScriptFixer 以及快速脚本。
- 12.23: ScriptTop 中出现 bug，没修好。
- 12.26: ScriptTop 修好，快速脚本完成。调试脚本模块。
- 12.27: 调试脚本模块。决定整理代码于 12.29 答辩。
- 12.28: 整理代码完成。

2 系统功能列表

2.1 手动模式

1. 通过按钮，玩家可以进行实现：

    - `game start/end`：分别对应了拨码 `0/1`。
    - `get`：右侧左边按钮。
    - `put`：右侧中间按钮。
    - `move`：右侧右边按钮。
    - `interact`：拨码 `2`。
    - `throw`：拨码 `3`。
    - `change target machine`：右侧上边按钮为顺时针移动一格，下边按钮为逆时针移动一格。

2. 对于开始和结束游戏：

    - 开始游戏：拨动拨码 0，可以开始游戏。玩家可以通过按钮进行各种操作。如果当前正处于脚本模式，则会直接进入手动模式。
    - 结束游戏：先拨动拨码 1，再回拨拨码 0，可以结束游戏。这是为了防止拨码接触不良以及玩游戏时误启动 `game end`，设计的“双保险”。

3. 可以自动过滤掉非法的操作：

    - 开发板能够阻止移动时 (玩家未在机器跟前时) 的非法交互。
    - 开发板能够阻止不合理的存取物品交互。
    - 开发板能够阻止不合理的投掷食材操作 (只可以投掷到桌子/垃圾桶)。
    - 开发板能保证操作信号是 `One-Hot` 编码的。

2.2 脚本模式

1. 通过向上拨动拨码 0 可以实现手动模式与脚本模式的切换。(向下拨动后可以切换至脚本模式，若拨码 0 在上方则强制为手动模式)
2. 通过输入脚本,可以实现 `get`,`put`,`interact`,`throw`,`wait`,`waituntil`, `jumpif`, `jumpifnot` 等操作。
3. 脚本模式中，4、5 拨码同时向上拨启动单步调试，随后下拨上拨一次拨码 5 执行下一条指令。

3 系统使用说明

使用说明已在功能列表中阐述。
在开发板上的实际按钮如下图所示:

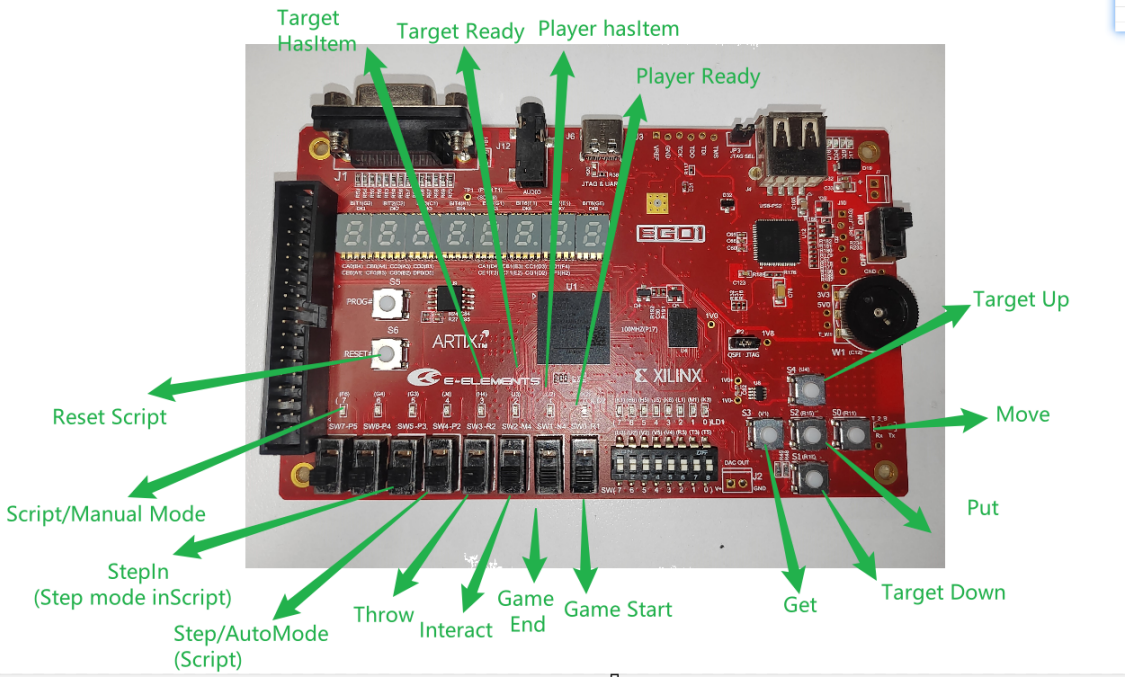

图 1: 图例

4 系统结构说明

本系统采用结构化的建模方式，依据需求文档将项目功能划分为手动模式与自动模式两大功能，分别设计对应的 Top 模块并统一接口，并使用 Designed Top 将两模块功能合并，整体的项目结构如下

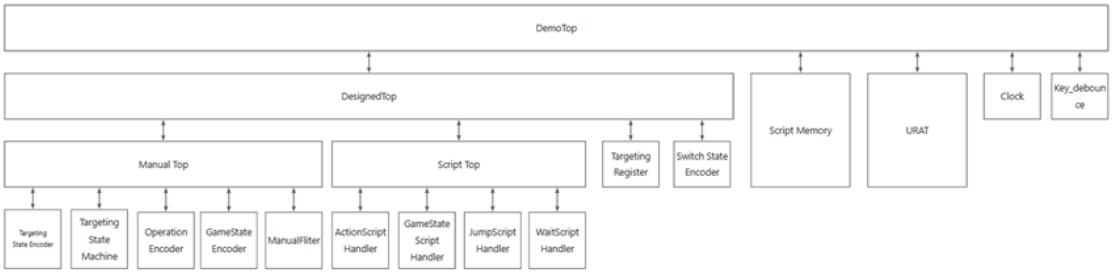

图 2: 顶层模块的结构示意

其中，对于手动模式功能的实现，我们将 Manual Top 模块设计为一个两状态状态机：状态一为轮询状态，状态二为发送状态，其状态切换关系如下

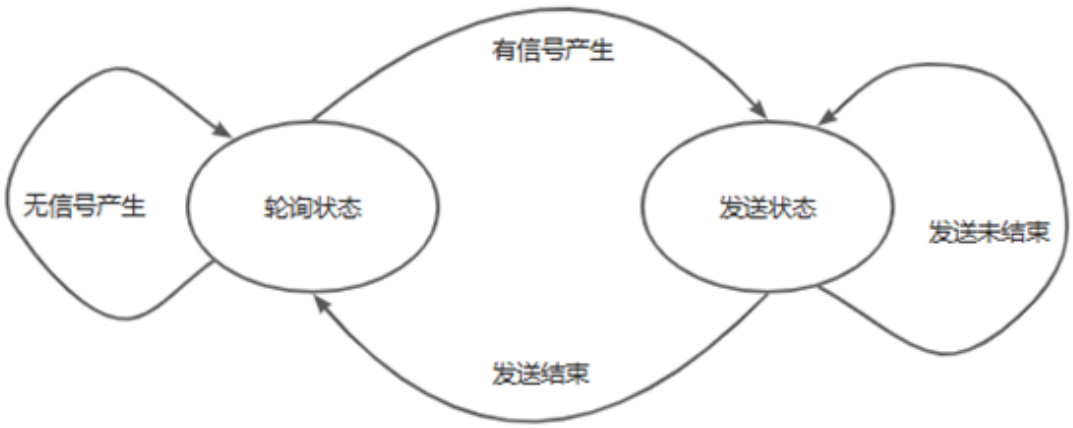

图 3: 手动模式状态机示意图

Manual Top 模块的结构示意如下

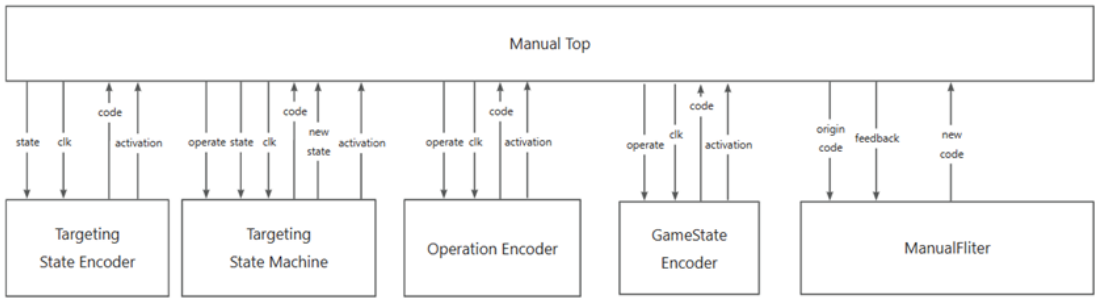

图 4: Manual Top 模块的结构示意

在初始阶段，Manual Top 模块将被置为状态一，此时该模块将会轮询其所例化的多个用于解析用户的不同操作的 Encoder 模块，如果用户执行了游戏操作，则对应的 Encoder 会将该操作依据通讯协议编码为对应指令传出，并产生一个时钟周期的激活脉冲信号，Manual 模块将会在轮询中接收到这个脉冲并切换至状态二。

由于需要发送较长时钟周期的指令信号才能确保该信号被客户端接收与处理，因此在发送状态时，Manual Top 模块将启动一个计数器用以向客户端发送指定时钟周期的指令信号，在计数结束后，该模块状态将会切换为状态一，以继续进行操作的轮询。

为实现对非法用户操作的过滤，Manual Top 模块例化了 Filter 模块用以过滤非法操作，Filter 模块将分析从客户端所接收的反馈信号以判断当前用户操作产生的指令信号是否合法，并对非法指令信号进行拦截，从而实现了对非法用户操作的过滤。

对于脚本模式功能的实现，我们将 Script Top 模块设计为三状态状态机：状态一为读取下一条指令，状态二为分析当前脚本，状态三为发送指令，其状态切换关系如下：

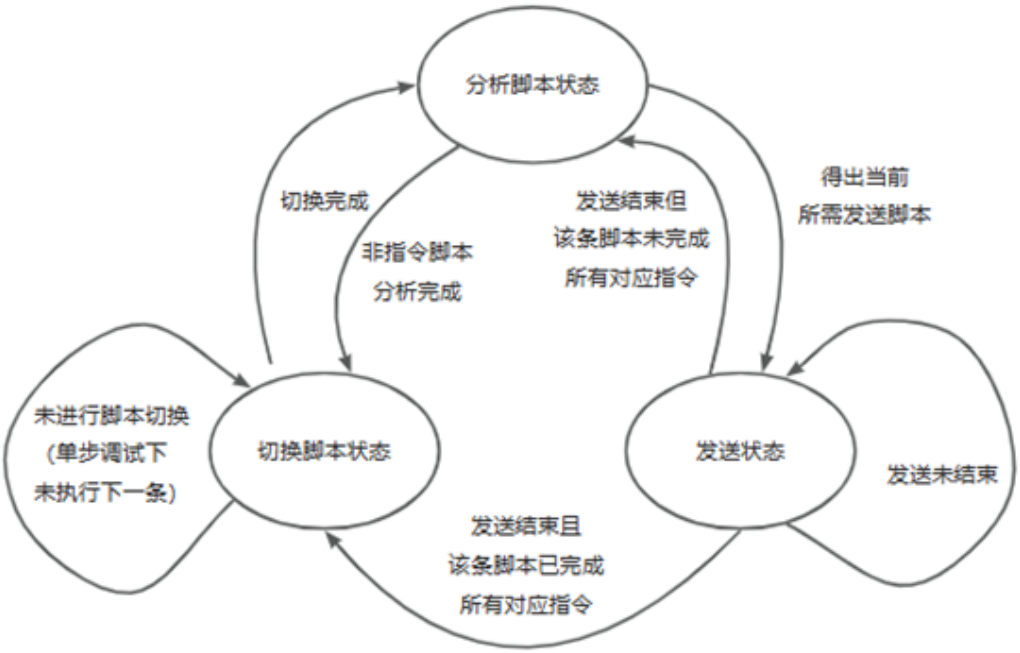

图 5: 自动模式状态机示意图

Script Top 模块的结构示意如下:

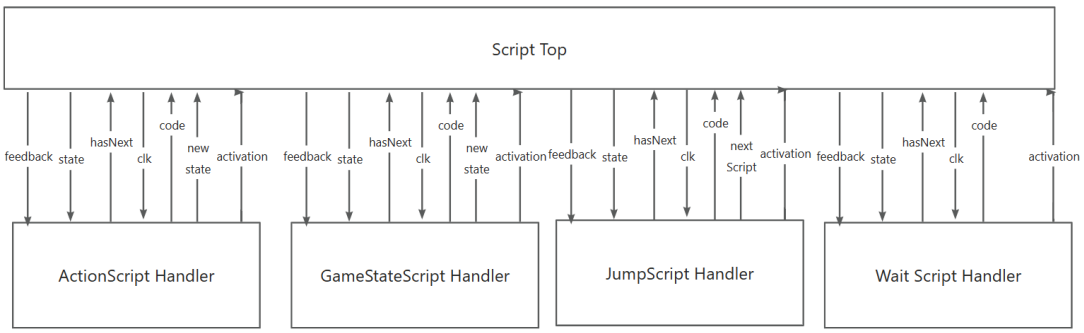

图 6: Script Top 模块的结构示意

在初始阶段，Script Top 模块将被置为状态一，此时该模块将会依据处

是否处在手动单步调试判断是否加载下一条脚本，如加载下一条脚本，则切换至状态二，否则维持当前状态，等待切换操作的发生。

对于状态二，该状态将依据 feedback 反馈与用户所处位置分析当前脚本所需执行的操作，如为指令操作则切换至发送状态，并且记录 hasNext 参数该脚本是否已经执行完成。如为非指令操作（如 jump 指令切换脚本地址）则返回状态一。

对于状态三，该状态将会发送指定时钟周期的传输信号，在发送结束后将依据上一条脚本是否执行完成切换状态。

基于上述设计，脚本模式实现了较为清晰的状态切换与需求划分，避免了在单一文件中分析所有脚本造成的状态切换复杂，具有一定的可扩展性。

Design Top 与 Demo Top 模块为上述模块的顶层模块，其模块关系如下：

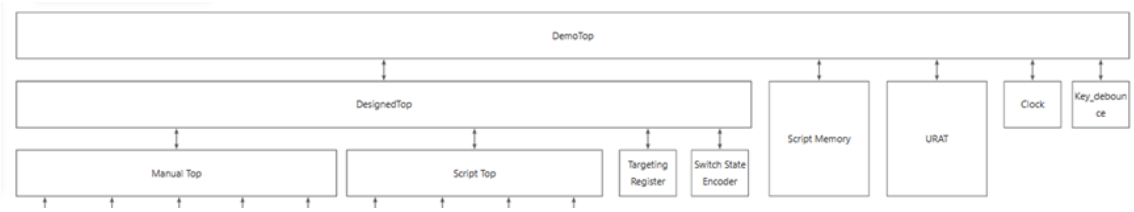

图 7: 顶层模块的结构示意

而对于 Design Top 模块，该模块例化了 Manual Top 与 Script Top 模块，该模块将使用 Target Register 存储目标机器的所处位置，并依据外部反馈信号进行手动模式与自动模式的切换，发送对应模块的反馈信号。Demo Top 模块则例化了 Designed Top 模块，使用所提供的 URAT 模块与 Script 模块实现脚本读取与客户端-开发板之间的交互，同时例化了时钟分频与消抖模块以提供合适且稳定的时钟与按键信号。

5 子模块功能说明

5.1 人工模式

- 游戏开始/结束
- 拾取物品
- 放置物品
- 移动
- 交互
- 扔
- 改变目标机器
- 过滤非法情况

5.2 脚本模式

- 手动模式与脚本模式的切换。
- `get`, `put`, `interact`, `throw`, `wait`, `waituntil`, `jump`, `jumpif` 等手动模式下操作。
- 脚本模式开始后任意时刻启动单步调试。

6 bonus 的实现说明

6.1 高效脚本设计

设计理念:

1. 考虑到一些及其操作是全自动的，因此操作者实际上可以利用这段时间去做一些别的事情，以节省时间。

2. 由于操作者的移动事实上是比较耗时的，因此在脚本中尽量避免了操作者长距离的跑动，若是需要运输物品这样的操作，如果能丢到附近的桌子上，那么操作者就不会亲自跑过去一趟。

执行完成时间：`12.6s` 左右。

6.2 手动单步调试与自动脚本模式任意切换

在 Bonus 部分的手动单步调试与自动脚本模式任意切换中，得益于 Script Top 模块较为清晰的状态设计，我们能够较为清晰方便地实现该功能，其实现为在切换脚本状态内部添加一个小状态机，用于判断是否为单步调试状态，如为单步调试状态则判断是否加载下一条指令以进行外部状态机的切换，如为自动运行状态则直接加载下一条指令，该实现避免了对外部状态机的直接改动，从而避免了对外部模块的代码改动，使该功能具备较好的稳定性与兼容性，其状态机设计如下：

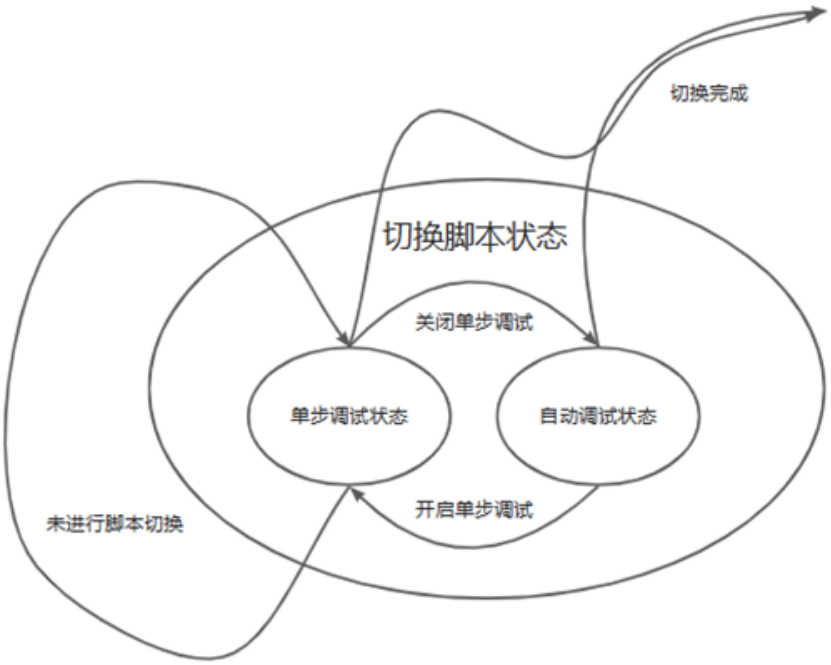

图 8: Manual Top 模块的结构示意

7 项目总结

在硬件开发方面，我们团队积累了丰富的经验。通过对不同的硬件模块进行研究和开发，我们深入了解了各种硬件的特性和使用方法，并能够根据需求进行相应的选择和调整。在实际的硬件开发过程中，我们熟练掌握了各种工具和技术，并能够快速解决遇到的问题。同时我们在调错中深入理解了硬件编程与软件编程的区别，培养了硬件编程的思维。

在团队合作方面，我们团队具备良好的沟通和协作能力。通过合理的任务分配和提供清晰的接口文档，我们能够有效地协调各个组员的工作，确保整个项目顺利进行。此外，我们定期召开团队会议，及时交流项目进展和存在的问题，并寻求共同解决方案。

在开发工作中，我们使用 `github` 平台进行版本控制和团队协作。通过合理的分支管理和代码审查，我们能够有效地协同完成项目的不同部分，并保证代码质量和可维护性。同时我们除了使用 `Vivado` 进行代码编辑，还安装了 `VScode` 中的插件对 `verilog` 进行编辑。

在测试工作中，我们采用了仿真测试的方法，对每个模块进行了全面的测试，并及时修复发现的问题。通过不断完善测试流程和测试用例，我们能够保证项目的质量和稳定性。同时我们在测试代码的过程中将变量绑到 `led` 灯中进行查看，这帮助我们发现了不少问题的成因。例如我们在编写 `Wait` 模块时，我们等待的时长一直不稳定，通过仿真测试，我们发现我们的一个变量的预期返回结果只会持续一个时钟周期，我们才修复了 `bug`；在编写脚本执行时，我们通过 `led` 灯发现 `ActionScriptHandler` 模块传入的 `feedback` 因为硬件原因位移了一位，使得 `Get` 指令执行不了。这两种测试方法给予了我们很大的帮助。

8 其他的想法和建议

1. 以保留往年数字逻辑的相关课设项目作为基础，并每年在其上进行适当的修改与创新，同时加强代码查重管理。这样可以保证课设项目难度的稳定性与项目质量，同时也降低了教学工作组的工作负担。

2. 关于新题目，可以考虑实现俄罗斯方块。该项目类似于本次厨房项目，

可以由助教负责编写 URAT 模块实现客户端与开发板的交互，学生需使用开发板分析客户端传输的信号，并依据用户操作实现对游戏进程的操控，包括游戏开始和结束的控制、方块的旋转与加速下落、不同类型方块的处理、消除方块等功能。此外，学生还可以通过 LED 与七段数码管反应当前游戏状态，实现游玩记录存储，或使用 VGA 画面显示以实现更丰富的界面。这样的项目具备一定的难度据分度，能够很好的运用同学们的课程所学，加强同学们对硬件语言的理解。