1 决赛提交

决赛提交包是基于预赛提交包的扩充。

1.1 提交作品格式

决赛提交的作品需包含：决赛分数报告、设计文档、各测试项生成的 bit 流文件、基于 axi 的 mycpu 源码和决赛展示内容。

提交目录参考决赛发布包（***nscscc2021_group_final_submission***）里的 ***submission/MOU_1_zhangsan***，有以下注意事项：

(1) 不要提交整个决赛发布包，只需要将 ***submission/MOU_1_zhangsan*** 按格式整理好后打包提交即可。

(2) 请将 ***MOU_1_zhangsan*** 文件夹按“**学校英文简写_队伍编号_队长名拼音**”的格式进行**更名**。

(3) 整理设计文档：

将 CPU 设计文档整理为 pdf 格式，复制到 ***submission/MOU_1_zhangsan/***目录里。

(4) 填写功能和性能得分：

将功能和性能得分填写到 ***submission/MOU_1_zhangsan/score_final.xlsx*** 文件里，请注意填写队名、学校名、队员和指导老师信息。**注意决赛性能分计算与预赛不同**。

(5) 整理决赛作品最后的 bit 流文件：

有 4 个 bit 流文件：功能测试 bit、记忆游戏 bit、性能测试 bit 和系统测试 bit，请分别放在 ***submission/MOU_1_zhangsan/bit/***目录里的对应子目录下。

请确保提交的 bit 流文件没有错误。

如果记忆游戏或系统测试未通过，则不需要提交对应的 bit 流文件。

(6) 整理决赛作品的 CPU 设计源码：

完成 AXI 接口 CPU 的同学，请将 CPU 设计源码放在 ***submission/MOU_1_zhangsan/src/mycpu/***目录中，同时将性能测试的 clk_pll IP 定制文件（***soc_axi_perf/rtl/xilinx_ip/clk_pll/clk_pll.xci***）复制到 ***submission/MOU_1_zhangsan/src/perf_clk_pll.xci***。

(7) 千万注意，CPU 设计源码中如果调用了 Xilinx IP（比如调用 Xilinx Block RAM IP），请将这些 IP 的定制文件*.xci 同样复制到 *submission/MOU_1_zhangsan/(sram_)src/mycpu/*目录中。也就是提交的 mycpu 源码应当包含除了大赛发布包提供的 SoC 相关文件外的你们所有新增的设计文件。

(8) **整理决赛的展示内容**：将其放在 ***submission/MOU_1_zhangsan/display/***目录中。

目录格式自行组织，**要求用于展示的 myCPU 设计与功能/性能测试里的 myCPU 完全一致**。

决赛赛提交的作品的目录格式如下（***submission/MOU_1_zhangsan***）：

|--score_final.xlsx	Excel 表格，包含功能测试、性能测试得分的计算
|--**design.pdf**	PDF 文件，为 myCPU 设计报告
|--display/	目录，决赛展示内容，要求 myCPU 与 src/目录里完全一致
|--sram_src/	目录，SRAM 目录（可选），忽略该目录即可

|--src/	目录，AXI 目录
| |--mycpu	目录，存放 AXI 相关工程**新增的源码**文件，如果调用了 Xilinx IP，本目录下只提交 IP 的 xci 文件。注意不要忘记自行调用的 Xilinx IP 的 xci 文件。
| |--perf_clk_pll.xci	复制性能测试的 clk_pll.xci，更名为 perf_clk_pll.xci。不要忘记这一步。

1.2 提交作品确认

在决赛发布包（***nscscc2021_group_final_submission***）中包含了 ***script*** 目录，该目录是我们的复核提交作品的半自动化测试环境，是在预赛发布的半自动化测试脚本的完善后的版本。

由于今年预赛复核评测不太顺利，同时今年决赛入围队伍增多，留给决赛复核的时间不太够。因此要求各队伍在提交作品时，使用 ***script*** 目录里的脚本对自己的提交作品进行确认，确认提交的作品可正确生成各 bit 文件。在我们复核时，如果发现某队伍提交的作品在该半自动化测试环境里无法正确生成各 bit 文件，则会将该队伍作品的复核工作排到最后：在最后时间来得及的情况下，我们会联系队伍确认提交的作品是否有错；如果最后时间不允许，则按复核成绩为零计算。

script 目录里的脚本具体使用方法如下：

(1) 确保 ***nscscc2021_group_final_submission/submission*** 目录下有你的提交作品，且 ***submission/MOU_1_zhangsan/src/***目录里的内容符合要求。

(2) 启动终端进入 ***nscscc2021_group_final_submission/***目录。

Linux 环境或 Windows 的 WSL 环境很好实现该步。

Windows 环境下也可以启动 CMD 终端，使用 cd 命令切换到 ***nscscc2021_group_qualifier_submission/***目录。

(3) 在终端中输入并执行命令“***vivado -mode batch -source script.tcl***”进行 Vivado 的批处理模式。

在 Linux 环境或 Windows 的 WSL 环境，如果 Vivado 安装目录已添加到环境变量中，则直接输入以上命令即可。

在 Windows 环境下，我们也可以找到 Vivado 的执行脚本，比如可能是“***C:\Xilinx\Vivado\2019.2\bin\vivado.bat***”，输入命令“***C:\Xilinx\Vivado\2019.2\bin\vivado -mode batch -source script.tcl***”也可以进入 Vivado 的批处理模式。

(4) 接下来，我们可以输入“***help***”命令查看所有支持的命令。

目前支持命令有：help、list、init、run、sim、download、exit。

(5) 首先输入命令“init all”完成你在 ***submission/***目录里放置的提交作品的工程初始化。

工程初始化会在 ***script/***目录里新建一个 ***project/***子目录，并会在 ***project/***目录中再新建一个你的作品的子目录，进而新建基于你的作品的功能测试、记忆游戏、性能测试和系统测试的 Vivado 工程。

(6) 然后输入“run all all”命令执行 ***project/***子目录里的各 Vivado 工程的综合并生成 bit 流文件。

该命令会依次生成功能测试、记忆游戏、性能测试和系统测试的 bit 流文件。

如果执行有错，请在 ***project/***子目录里打开对应的 vivado 工程，查看错误原因，确认提交的源码是否符合规范，修改后重新执行(5)、(6)步。

如果执行无误，则会在 ***result/***子目录里生成对应的 bit 流文件。

(7) 请将 ***result/***子目录里的各 bit 流文件依次下载到实验箱上，确认这些 bit 流文件可以正确运行。如果运行结果不符合预期，请检查提交的作品（脚本工程源码拷贝自 ***submission/MOU_1_zhangsan/src/***），修改后

重新执行(5)、(6)、(7)步。

完成以上测试后，就可以放心的提交作品了。

1.3 提交方式

整个提交的压缩包应当尽量小，提交方式：

（1） 请将 ***MOU_1_zhangsan*** 文件夹按“**学校英文简写_队伍编号_队长名拼音**”的格式进行**更名**，并压缩，压缩包格式为 ZIP 格式，压缩之后的名称应与文件夹名称一致，如“MOU_1_zhangsan.zip”，不能包含中文。（如果一个学校只有一个队伍，则自动编号为 1；如果有两个队伍，不知道编号，请相互协商分为 1、2 队）。

（2） 压缩包请直接以邮件附件形式发送到 ***service@nscscc.org***，邮件名为：【2021 决赛作品提交】【xx 大学】【x 队】【队长名】【日期】，如【2021 决赛作品提交】【某大学】【1 队】【张三】【20210816】。多次提交，以最后提交为准。

1.4 提交截止时间

决赛提交截止时间：2021 年 8 月 16 日 11:59:59（确切时间参见后续公布的预赛结果通知）。逾期不接收提交，视为放弃决赛。

1.5 决赛性能分计算变更说明

和去年一样，今年大赛的决赛的性能分占比 40%，决赛性能分和预赛性能分计算方式不同。

决赛性能分分为两个维度：处理器主频和程序执行周期数（利用 CP0 寄存器 Count 计数），各占 50%。

(1) 处理器主频：各队伍提交的作品中指定的 CPU 时钟的频率（要求在 vivado 的 implementation 后 WNS 为正值）。

(2) 程序执行周期数：利用 CPU 内部的 CP0 寄存器 Count 测量基准测试程序运行的时钟周期数。要求 CP0 Count 的累加频率是 CPU 时钟频率的一半（每两个 CPU 时钟累加 1），因此该计数值和 CPU 频率成正比，对于相同测试程序，该值可代表 CPU 执行基准测试程序的 CPI（每条指令的平均执行周期数，也就是 IPC 的倒数）。

因此决赛的性能测试程序上板时，需注意：最右侧 4 个拨码开关指定运行的基准测试程序，按下复位键开始运行，运行通过后，使用最左侧拨码开关控制数码管显示相应的计数结果：

(1) 最左侧拨码开关拨下，数码管显示 CPU 内部 CP0 Count 的计数结果。

(2) 最左侧拨码开关拨上，数码管显示 SoC 中 CONFREG 模块的 Timer 的计数结果（该值与预赛中性能计数方式的结果接近）。

因此决赛得分请记录在决赛发布包中的 ***submission/MOU_1_zhangsan/score_final.xlsx***。同时新的计算方式，**要求 myCPU 里实现 CP0 寄存器 Count**。

1.6 关于系统测试调试的重要说明

很多队伍的功能测试、记忆游戏和性能测试都能正确运行，但是系统测试无法正确运行的原因很有可能是：**lb 指令的地址是 uncached 属性时，将 lb 当做了 lw 对外发起了请求。**

Uncache 属性的 lb/lh 不能当作 lw 对外发请求，取回四字节，再选出需要的那 1 个或 2 个字节！它们必须在 AXI 接口上严格控制 araddr 和 arsize，使得 AXI 接口上与 lb/lh 严格对应。原因如下：

(1) Uncache 属性的 load 执行不允许推测执行，必须是确定性的执行。如果 Uncache 属性的 lb 当做 lw 执行，则对于不需要的另外 3 字节则属于推测执行（软件不需要读取那三个字节，但 CPU 却读回来丢弃掉了）。这是不允许的，uncache 属性的 load 不允许推测执行！

(2) 团体赛提供的系统测试环 SoC 中的串口设备属于字节设备，只支持字节宽度的访问。对于访问会使用 uncache 属性的 lb 指令，如果 uncache 属性的 lb 当做 lw 对外发起请求，则串口设备无法给予正确的响应。

另外提醒一下，地址空间的 Kseg1 段（虚地址 0xa000_0000~0xbfff_ffff）应该实现为永远是 uncache 属性。

1.7 决赛设计文档

决赛设计文档会提供给专家评阅，可能会作为专家评分标准之一，请自行组织内容。

1.8 注意事项

(1) 请决赛提交时千万注意格式，请在提交前完成第 1.2 节的内容。

(2) 请继续保持 myCPU 的主频没有负的 WNS。

(3) 决赛现场指令集答题，需要大家自行准备：电脑、实验箱、决赛提交包里的 soc_axi_func 环境(CPU 频率尽量设低，减少综合实现的时间)。

(4) 决赛提交的用于功能/性能测试的 myCPU、用于展示的 myCPU 和现场指令集答题时的 myCPU 要求完全一致。