



在SERVE云平台上 实现处理器敏捷开发与原型验证

王嵩岳

2021.10.9

目录

- SERVE云平台简介
- 云平台提供的开发验证流程
- 如何使用SERVE平台验证自己的设计
- Q&A

目录

- **SERVE云平台简介**
- 云平台提供的开发验证流程
- 如何使用SERVE平台验证自己的设计
- Q&A

芯片研发存在哪些“痛点”？

① 设计

- 设计与调试手段不敏捷
- 从头设计成本过高
- 开发工具笨重

② 验证

- 功能验证消耗大量算力
- 原型验证需要高成本的器件，成本贵

③ 生产

- 先进的工艺受制于人

如何降低芯片开发成本？

- 计算密集型任务

- 大量算力需求

① 设计

- 设计与调试手段不敏捷
- 从头设计成本过高
- 开发工具笨重

② 验证

- 功能验证消耗大量算力
- 原型验证需要高成本的器件，成本贵

- 共享资源型任务

- 板卡利用率并不高

• 先



云计算与云服务

- 计算密集型任务

- 大量算力需求

云计算

- 共享资源型任务

- 板卡利用率并不高

云服务

① 设计

- 设计与调试手段不敏捷
- 从头设计成本过高
- 开发工具笨重

② 验证

- 功能验证消耗大量算力
- 原型验证需要高成本的器件, 成本贵

• 先

SERVE可以提供什么

- 基于GitLab 代码托管与 CI/CD
- 持续开发流程（CI）：提供**通用计算和存储能力**
 - 开发者无需在本地配置笨重的开发工具（如EDA软件）
 - 开发者可直接在云上进行高算力需求的流程：如仿真、实现
- 持续原型验证（CD）：提供共享的**云FPGA**
 - 开发者无需购买昂贵的板卡
 - 提高了板卡资源的利用率
 - 先进的板卡：板上资源种类和数量充足

仿真、综合、实现

原型验证

□ 最新定制NetFirm（NF）板卡

- 使用Xilinx公司Zynq Ultrascale+ MPSoC系列中规格最高、逻辑容量最大的芯片（XCZU19EG）
- 外设类型及接口更加丰富，支持更加复杂的实验内容



对比指标	NF板卡	ZyForce板卡	指标: 提升幅度
逻辑规模	600K	103K	容量: ~6倍
DDR4内存	2 x 16GB	1 x 2GB	容量: 16倍
网络接口	4 x 100Gbps	1 x 1Gbps	速率: 400倍
存储	32GB 低速SD卡	1TB 高速固态硬盘	容量: 32倍

SERVE的使用场景

- 硬件敏捷开发设计
 - 国科大《计算机组成原理实验》
 - “逐梦杯” 芯片设计大赛
 - “一生一芯”
 - ...
- 软件设计
 - 国科大《操作系统研讨课》
 - ...

提供VIVADO, OpenROAD, yosys等
工具的开发流程

目录

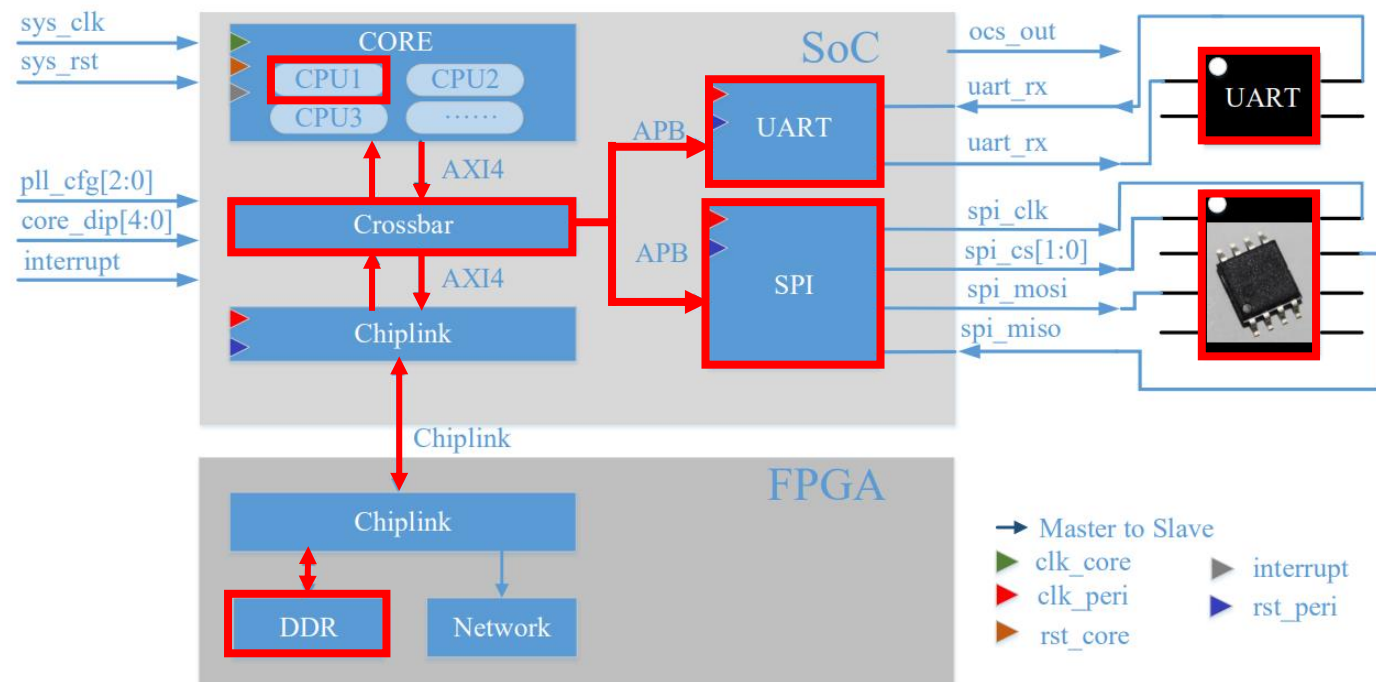
- SERVE云平台简介
- 云平台提供的开发验证流程
- 如何使用SERVE平台验证自己的设计
- Q&A

针对第三期一生一芯的开发验证框架

- **功能仿真+云FPGA上板验证**
- **兼具功能测试和SoC测试**
 - 云上仿真：功能测试
 - 支持 difftest 以验证功能正确性
 - 支持运行带 microbench 的 rt-thread 作为DUT的测试程序

针对第三期一生一芯的开发验证框架

- 功能仿真+云FPGA上板验证
- 兼具**功能测试**和**SoC测试**
 - FPGA验证：类似SoC的工程
 - 框架模拟了如下**红色**通路
 - 使用板载真实的**DDR内存**
 - 模拟了**SPI Flash**
 - 字符经由**UART控制器**打印



SERVE云提供的验证流程

功能仿真

原型验证

scalastyle

检查Scala语法

生成difttest仿真
程序

功能仿真
(difttest)

生成verilog工程与
文件

生成上板所需比特
流

云FPGA
上板验证

SERVE云提供的验证流程

功能仿真

原型验证

scalastyle

create_emu

检查Scala语法

生成difttest仿真
程序

功能仿真
(difttest)

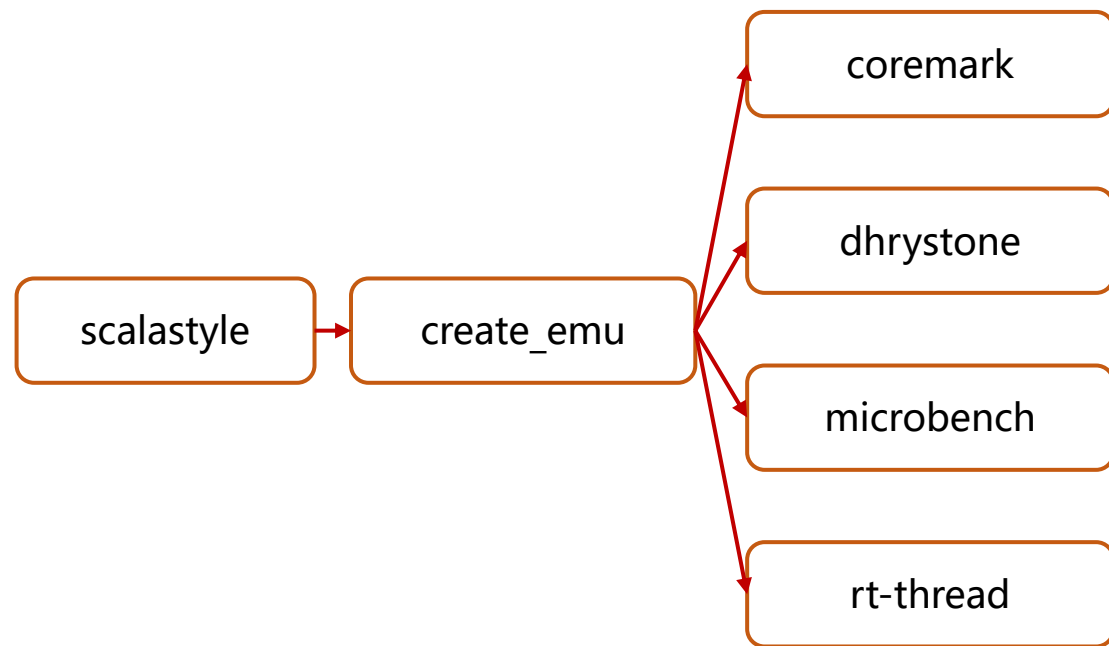
生成verilog工程与
文件

生成上板所需比特
流

云FPGA
上板验证

SERVE云提供的验证流程

功能仿真



原型验证

检查Scala语法

生成difttest仿真
程序

功能仿真
(difttest)

生成verilog工程与
文件

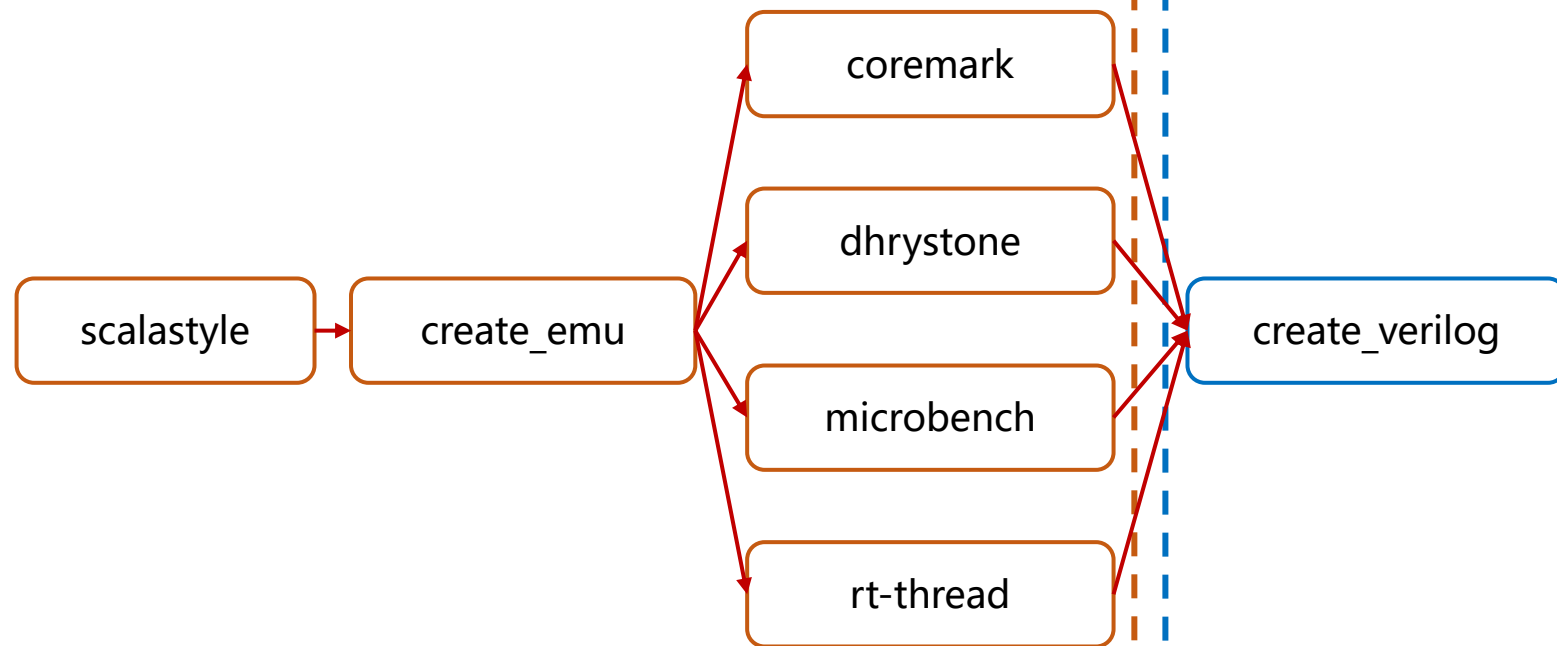
生成上板所需比特
流

云FPGA
上板验证

SERVE云提供的验证流程

功能仿真

原型验证



检查Scala语法

生成difttest仿真
程序

功能仿真
(difttest)

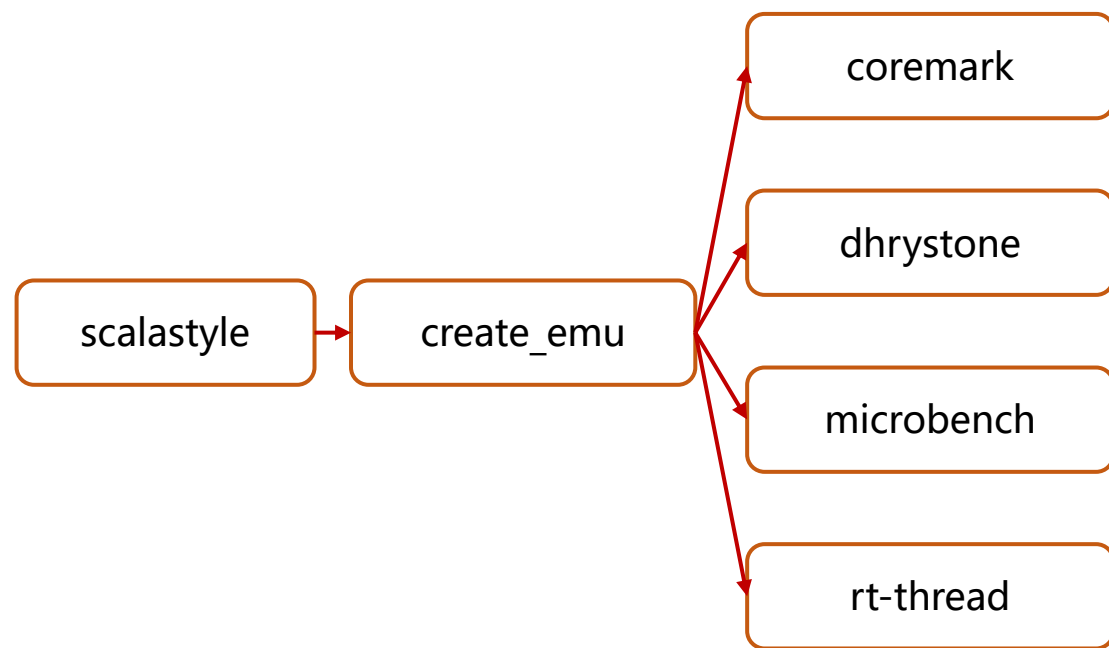
生成verilog工程与
文件

生成上板所需比特
流

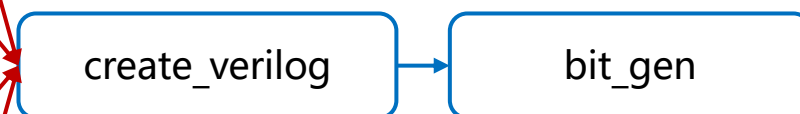
云FPGA
上板验证

SERVE云提供的验证流程

功能仿真



原型验证

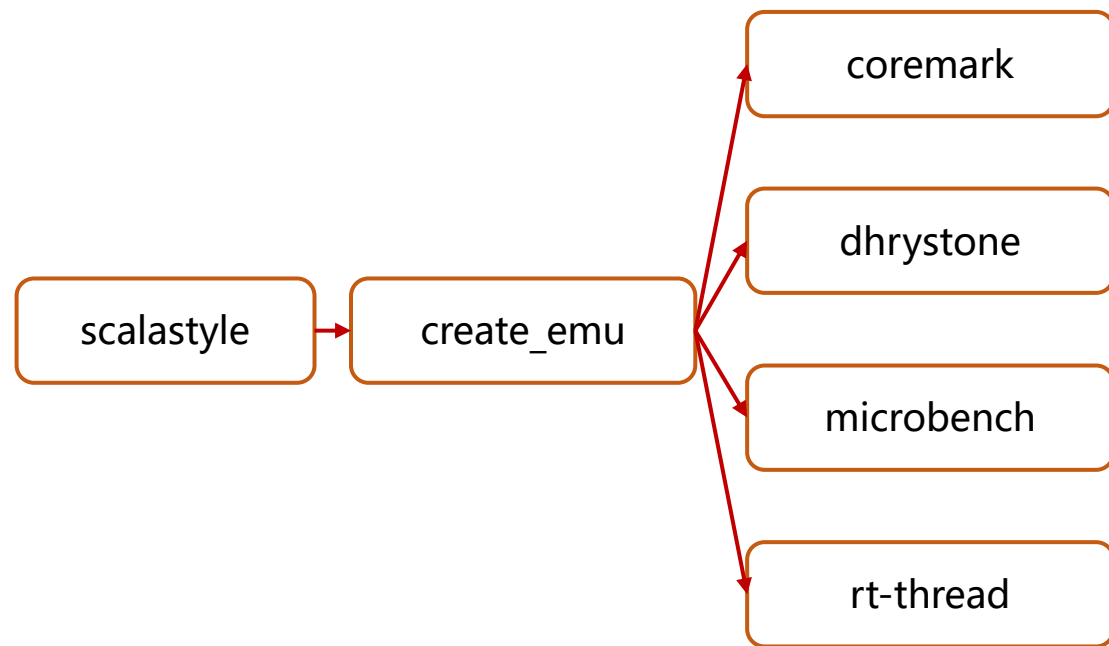


这一步会报出
Verilog 语法错误

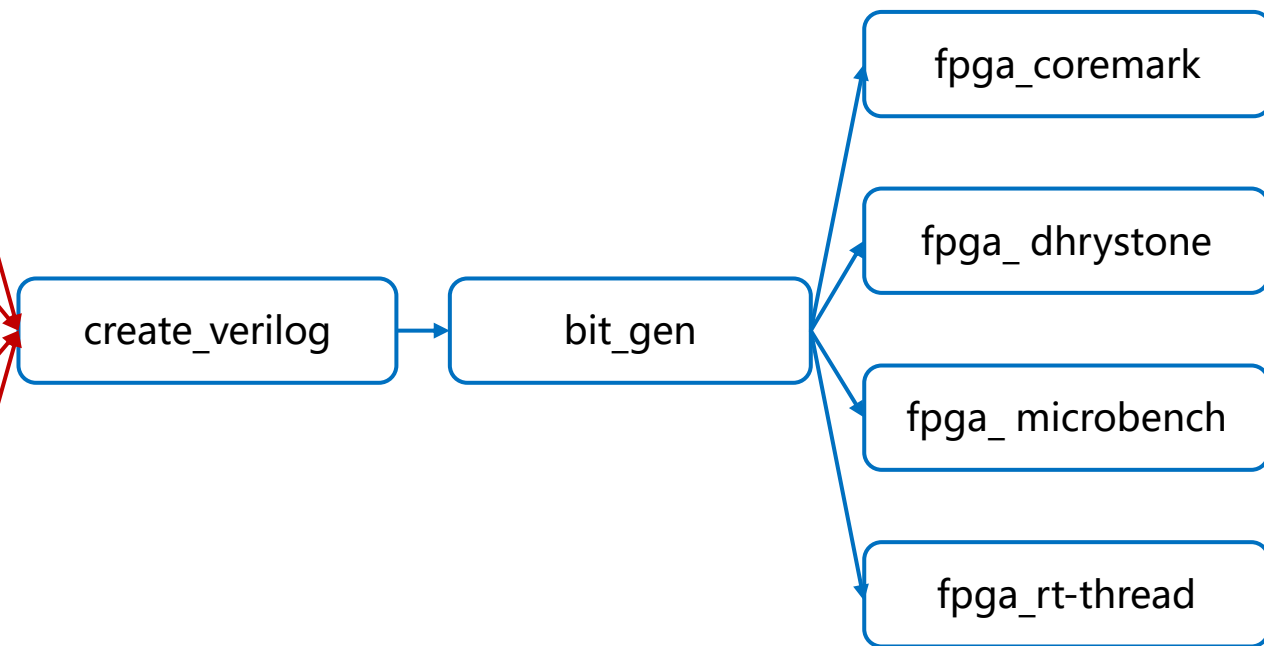


SERVE云提供的验证流程

功能仿真



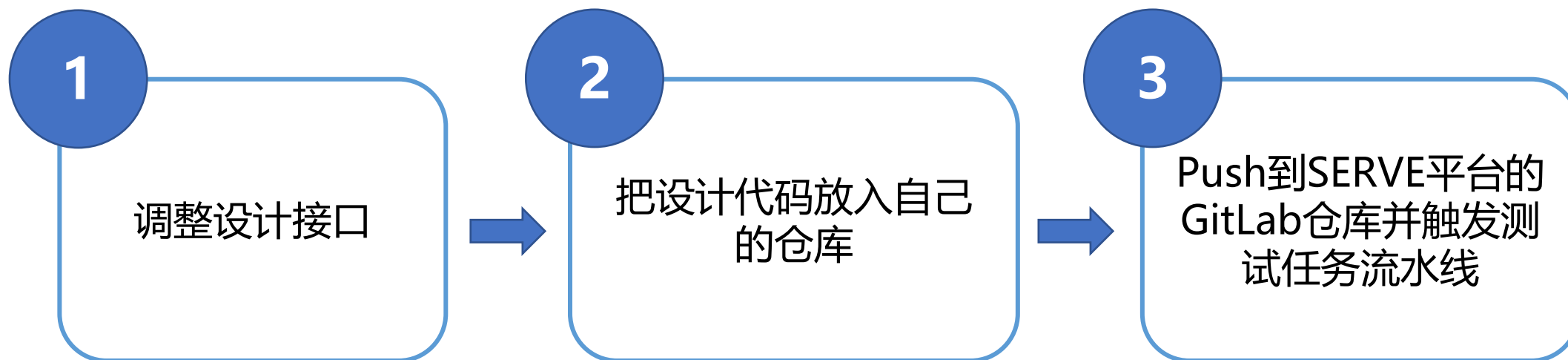
原型验证



目录

- SERVE云平台简介
- 云平台提供的开发验证流程
- 如何使用SERVE平台验证自己的设计
- Q&A

将你的CPU适配到SERVE测试环境



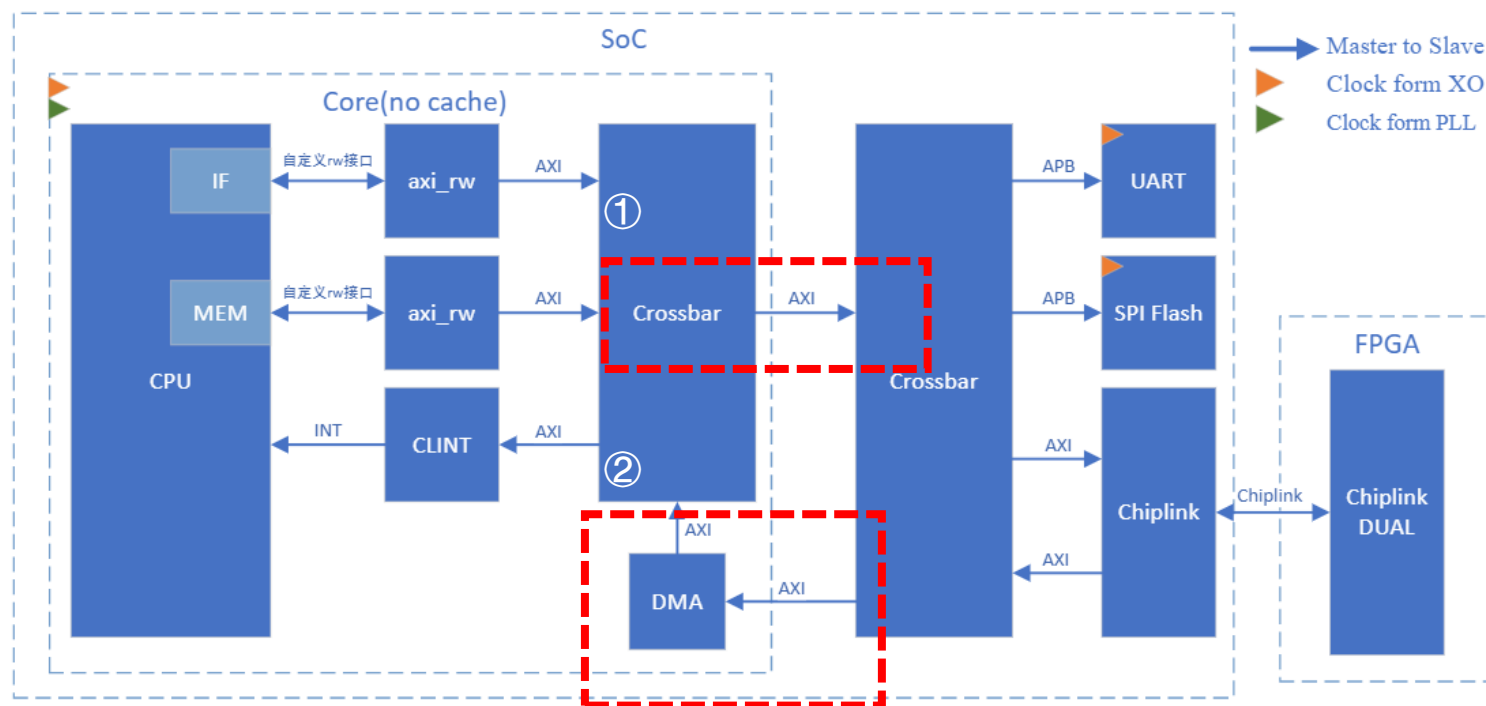
将你的CPU适配到SERVE测试环境



调整CPU设计接口

1. CPU接口概览：与一生一芯CPU规范最新接口一致

- ① AXI Master: CPU对外只有一个
- ② AXI Slave: 预留DMA接口（悬空，云平台暂不支持）



调整CPU设计接口

1. CPU接口概览

- ① AXI Master: CPU对外只有一个
- ② AXI Slave: 预留DMA接口（悬空，云平台暂不支持）
- ③ 外部中断暂不支持

调整CPU设计接口

2.1 仿真环境地址空间分配

应按如下地址空间调整设计，需要特别注意以下方面

- 复位PC: 0x8000_0000
- 实现Cache的同学，请将低于0x8000_0000的地址设为旁路（不可缓存）

地址范围	说明
0x4060_0000	UART
0x8000_0000	入口地址
0x8000_0000~+	Memory

调整CPU设计接口

2.2 云FPGA验证环境地址空间分配

应按如下地址空间调整设计，需要特别注意以下方面

- 复位PC: **0x3000_0000**

0x30000000处模拟了**SPI Flash**，存放两条指令：

```
30000000: auipc ra, 0x50000
```

```
30000004: jalr ra
```

作用：跳转到0x8000_0000

模拟了SoC框架内从SPI Flash长跳转到内存的过程

调整CPU设计接口

2.2 云FPGA验证环境地址空间分配

应按如下地址空间调整设计，需要特别注意以下方面

- 复位PC: **0x3000_0000**
- 实现Cache的同学，请将低于0x8000_0000的地址设为旁路（不可缓存）

地址范围	说明
0x2000_0000	UART
0x3000_0000	SPI Flash (入口地址)
0x8000_0000~+	Memory

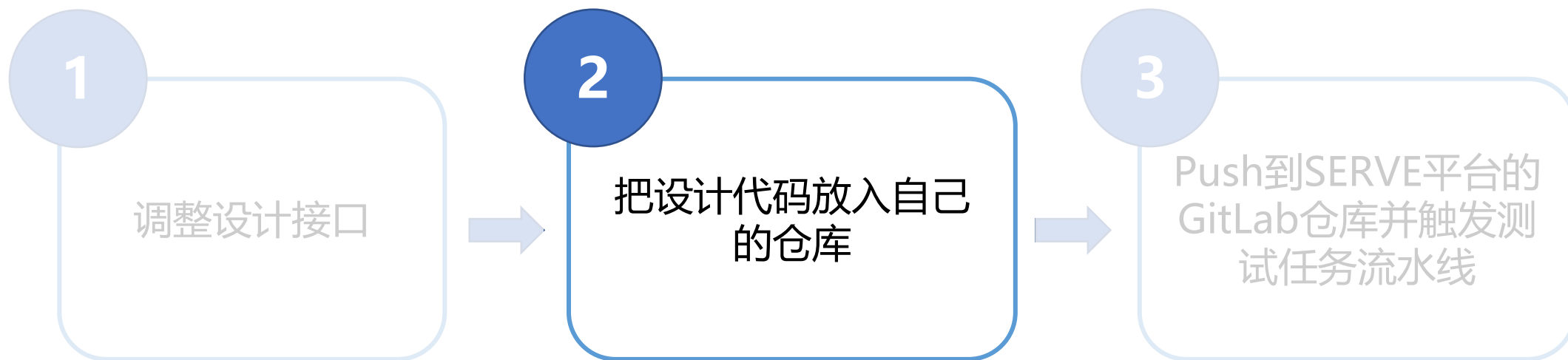
调整CPU设计接口

总结：地址空间分配

学生设计无需考虑外设地址，只需将低于0x8000_0000的地址设为旁路并在仿真和云FPGA测试时赋予不同的复位PC

	仿真地址空间	云FPGA验证的地址空间	备注
UART	0x4060_0000	0x2000_0000	旁路
SPI Flash	-	0x3000_0000	
内存	0x80000000+		
入口地址	0x8000_0000	0x3000_0000	

将你的CPU适配到SERVE测试环境



查看仓库并放入代码

- GitLab入口
 - <https://gitlab.agileserve.org.cn:8001>
 - 登录GitLab后，可看到rvcpu仓库
- 放入自己的设计
 - 将**Chisel**设计文件(.scala)放入**src**目录下
 - 将**Verilog**设计文件(.v)放入**vsrc**目录下

Name		Last commit
📁 fpga	上板vivado工程	Fixed some names.
📁 project		Update .gitlab-ci.yml
📁 scripts		Update and clean project
📁 src/main/scala	CPU设计代码	Fixed scala code.
📁 vsrc		Update and clean project
🔥 difftest @ 86302f80		Update difftest
🔥 .gitignore	云平台流程控制脚本	lsu: disable oo mem access
🔥 .gitlab-ci.yml		Fixed gitlab ci
🔥 .gitmodules		added gitlab-ci.yml file.
🔥 LICENSE		license: add copyright over
📄 Makefile		Update compile scripts.
📄 Makefile.include		Update compile scripts.
📄 README.md		fix the broken link to mil
🔗 build.sbt		bump chisel version to 3
📄 build.sc		difftest: remove built-in c
🖼️ debian_on_fpga.gif		misc: update debian on

注意：
 一生一芯在SERVE平台上的用户已经注册完成，用户可以直接到平台登录页面进行密码重置，随后即可用新密码登录平台。
 登录页面：https://gitlab.agileserve.org.cn:8001/users/sign_in
 密码重置：<https://gitlab.agileserve.org.cn:8001/users/password/new>

修改顶层以适配工程

- 将顶层模块名修改为为SimTop
- **与一生一芯OSCPU/oscpu-framework最新顶层一致**
 - https://github.com/OSCPU/oscpu-framework/blob/2021/projects/cpu_axi_diff/vsrc/SimTop.v



顶层模块SimTop

```
5 module SimTop(  
6     input                clock,  
7     input                reset,  
8  
9     input  [63:0]        io_logCtrl_log_begin,  
10    input  [63:0]        io_logCtrl_log_end,  
11    input  [63:0]        io_logCtrl_log_level,  
12    input                io_perfInfo_clean,  
13    input                io_perfInfo_dump,  
14  
15    output                io_uart_out_valid,  
16    output [7:0]          io_uart_out_ch,  
17    output                io_uart_in_valid,  
18    input  [7:0]          io_uart_in_ch,  
19  
20    input                `AXI_TOP_INTERFACE(aw_ready),  
21    output                `AXI_TOP_INTERFACE(aw_valid),  
22    output [`AXI_ADDR_WIDTH-1:0] `AXI_TOP_INTERFACE(aw_bits_addr),  
23    output [2:0]          `AXI_TOP_INTERFACE(aw_bits_prot),  
24    output [`AXI_ID_WIDTH-1:0] `AXI_TOP_INTERFACE(aw_bits_id),  
25    output [`AXI_USER_WIDTH-1:0] `AXI_TOP_INTERFACE(aw_bits_user),  
26    output [7:0]          `AXI_TOP_INTERFACE(aw_bits_len),  
27    output [2:0]          `AXI_TOP_INTERFACE(aw_bits_size),  
28    output [1:0]          `AXI_TOP_INTERFACE(aw_bits_burst),  
29    output                `AXI_TOP_INTERFACE(aw_bits_lock),  
30    output [3:0]          `AXI_TOP_INTERFACE(aw_bits_cache),  
31    output [3:0]          `AXI_TOP_INTERFACE(aw_bits_qos),  
32  
33    input                `AXI_TOP_INTERFACE(w_ready),  
34    output                `AXI_TOP_INTERFACE(w_valid),  
35    output [`AXI_DATA_WIDTH-1:0] `AXI_TOP_INTERFACE(w_bits_data)    [3:0],  
36    output [`AXI_DATA_WIDTH/8-1:0] `AXI_TOP_INTERFACE(w_bits_strb),  
37    output                `AXI_TOP_INTERFACE(w_bits_last),
```

```
39    output                `AXI_TOP_INTERFACE(b_ready),  
40    input                `AXI_TOP_INTERFACE(b_valid),  
41    input  [1:0]          `AXI_TOP_INTERFACE(b_bits_resp),  
42    input  [`AXI_ID_WIDTH-1:0] `AXI_TOP_INTERFACE(b_bits_id),  
43    input  [`AXI_USER_WIDTH-1:0] `AXI_TOP_INTERFACE(b_bits_user),  
44  
45    input                `AXI_TOP_INTERFACE(ar_ready),  
46    output                `AXI_TOP_INTERFACE(ar_valid),  
47    output [`AXI_ADDR_WIDTH-1:0] `AXI_TOP_INTERFACE(ar_bits_addr),  
48    output [2:0]          `AXI_TOP_INTERFACE(ar_bits_prot),  
49    output [`AXI_ID_WIDTH-1:0] `AXI_TOP_INTERFACE(ar_bits_id),  
50    output [`AXI_USER_WIDTH-1:0] `AXI_TOP_INTERFACE(ar_bits_user),  
51    output [7:0]          `AXI_TOP_INTERFACE(ar_bits_len),  
52    output [2:0]          `AXI_TOP_INTERFACE(ar_bits_size),  
53    output [1:0]          `AXI_TOP_INTERFACE(ar_bits_burst),  
54    output                `AXI_TOP_INTERFACE(ar_bits_lock),  
55    output [3:0]          `AXI_TOP_INTERFACE(ar_bits_cache),  
56    output [3:0]          `AXI_TOP_INTERFACE(ar_bits_qos),  
57  
58    output                `AXI_TOP_INTERFACE(r_ready),  
59    input                `AXI_TOP_INTERFACE(r_valid),  
60    input  [1:0]          `AXI_TOP_INTERFACE(r_bits_resp),  
61    input  [`AXI_DATA_WIDTH-1:0] `AXI_TOP_INTERFACE(r_bits_data)    [3:0],  
62    input                `AXI_TOP_INTERFACE(r_bits_last),  
63    input  [`AXI_ID_WIDTH-1:0] `AXI_TOP_INTERFACE(r_bits_id),  
64    input  [`AXI_USER_WIDTH-1:0] `AXI_TOP_INTERFACE(r_bits_user)  
65 );
```

调整自设计CPU时钟频率

- 若希望修改FPGA验证时自设计CPU的频率，可修改Makefile.include
 - 默认值是100MHz
 - 与仿真无关，仅影响设计综合实现后上FPGA测试的时钟

 **Makefile.include**  63 Bytes

```
1 # Set the frequency of tested module on board.  
2 DUT_FREQ ?= 100
```

将你的CPU适配到SERVE测试环境





本地提交后打开GitLab网页检查结果

• 本地

- 修改代码完毕后，通过git add、git commit和git push命令提交

• 云端

- 打开自己的项目
- 点击左侧CI/CD
- 查看流水线运行结果

刘士祺 > NutShell > Pipelines

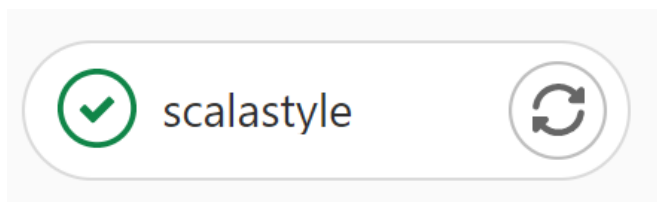
All 183 Finished Branches Tags

Filter pipelines

Status	Pipeline	Triggerer	Commit	Stages
running	#37827 latest		difftest -> bf1ad5d9 Update AXI4UART.sc...	! [stage icons]
running	#37826		difftest -> f8f73449 Update AXI4UART.sc...	[stage icons]
failed	#37825		difftest -> 573f324d Update AXI4UART.sc...	! [stage icons]
passed	#37824		difftest -> 4f4f4e8d Update AXI4UART.sc...	[stage icons]

Scala语法与代码风格检查

- 可点击查看检查的log



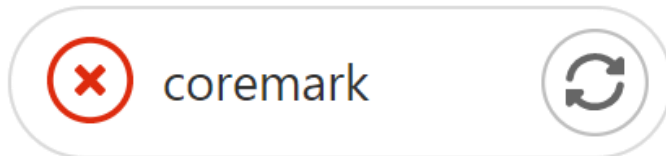
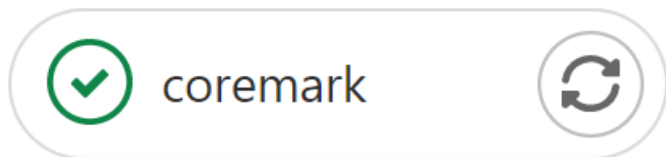
```
2633 [warn] /builds/liushiqi/NutShell/src/main/scala/utils/SRAMTemplate.scala:51:6: Public method must have explicit type
2634 [warn] /builds/liushiqi/NutShell/src/main/scala/utils/SRAMTemplate.scala:61:6: Public method must have explicit type
2635 [warn] /builds/liushiqi/NutShell/src/main/scala/utils/StopWatch.scala:1: Header does not match expected text
2636 [warn] /builds/liushiqi/NutShell/src/main/scala/utils/StopWatch.scala:4:1: Whitespace at end of line
2637 [warn] /builds/liushiqi/NutShell/src/main/scala/utils/StopWatch.scala:6:86: Whitespace at end of line
2638 [warn] /builds/liushiqi/NutShell/src/main/scala/utils/StopWatch.scala:8:51: Whitespace at end of line
2639 [warn] /builds/liushiqi/NutShell/src/main/scala/utils/StopWatch.scala:9:1: Whitespace at end of line
2640 [warn] /builds/liushiqi/NutShell/src/main/scala/utils/StopWatch.scala:10:87: Whitespace at end of line
2641 [warn] /builds/liushiqi/NutShell/src/main/scala/utils/StopWatch.scala:11:87: Whitespace at end of line
2642 [warn] /builds/liushiqi/NutShell/src/main/scala/utils/StopWatch.scala:12:31: Whitespace at end of line
2643 [warn] /builds/liushiqi/NutShell/src/main/scala/utils/StopWatch.scala:14:40: Whitespace at end of line
2644 [warn] /builds/liushiqi/NutShell/src/main/scala/utils/StopWatch.scala:23:6: Public method must have explicit type
2645 [info] scalastyle Processed 77 file(s)
2646 [info] scalastyle Found 0 errors
2647 [info] scalastyle Found 2606 warnings
2648 [info] scalastyle Found 0 infos
2649 [info] scalastyle Finished in 32 ms
2650 [success] created output: /builds/liushiqi/NutShell/target
2651 [warn] warnings exist
2652 [success] Total time: 6 s, completed Aug 23, 2021, 6:48:22 AM
✓ 2654 Cleaning up file based variables
2656 Job succeeded
```

样例结果, 仅供展示

00:02

行为仿真

- 正确性判断：观察运行结果是勾/叉
- 可点击查看运行difttest的log







```
57 Using simulated 8192MB RAM
58 Using /usr/src/NEMU/build/riscv64-nemu-interpreter-so for difftest
59 The first instruction of core 0 has committed. Difftest enabled.
60 Running CoreMark for 10 iterations
61 2K performance run parameters for coremark.
62 CoreMark Size : 666
63 Total time (ms) : 0
64 Iterations : 10
65 Compiler version : GCC9.3.0
66 seedcrc : 0xe9f5
67 [0]crclist : 0xe714
68 [0]crcmatrix : 0x1fd7
69 [0]crcstate : 0x8e3a
70 [0]crcfinal : 0xfcaf
71 Finised in 0 ms.
72 =====
73 CoreMark Iterations/Sec = 1
74 Core 0: HIT GOOD TRAP at pc = 0x800023d0
75 total guest instructions = 3706505
76 instrCnt = 3706505, cycleCnt = 6809605, IPC = 0.544305
```



样例结果，仅供展示

上板验证

Fpga_eval



 fpga_coremark 

 fpga_dhrysto... 

 fpga_microb... 

- 正确性判断：观察运行结果是勾/叉
- 可点击查看上板打印的log

上板验证

- 正确性判断：观察运行结果是  
- 可点击查看上板打印的log
 - 同时接入计时器，记录运行各单项测试的时间并计算分数（分数越高性能越强）
 - 正确运行结束打印 **Hit good trap**，错误打印Hit bad trap

```
32 $ bash -c "scripts/fpga_run.sh microbench"
33 ===== Running MicroBench [input *ref*] =====
34 [qsort] Quick sort: * Passed.
35   min time: 1201 ms [425]
36 [queen] Queen placement: * Passed.
37   min time: 1588 ms [296]
38 [bf] interpreter: * Passed.
39   min time: 11393 ms [207]
40 [fib] Fibonacci number: * Passed.
41   min time: 25976 ms [109]
42 [sieve] Eratosthenes sieve: * Passed.
43   min time: 32691 ms [120]
44 [15pz] A* 15-puzzle search: * Passed.
45   min time: 3480 ms [128]
46 [dinic] Dinic's maxflow algorithm: * Passed.
47   min time: 7355 ms [147]
48 [lzip] Lzip compression: * Passed.
49   min time: 4109 ms [184]
50 [ssort] Suffix sort: * Passed.
51   min time: 5550 ms [81]
52 [md5] MD5 digest: * Passed.
53   min time: 14320 ms [120]
54 =====
55 MicroBench PASS      181 Marks
56                      vs. 100000 Marks (i7-7700K @ 4.20GHz)
57 Total time: 122814 ms
58 Exit with code = 0
59 Hit good trap
```

样例结果，仅供展示



SERVE