一生一芯三期
SOC架构设计与对接规范

报 告 人：龙康杰、陈璐

指导老师：刘彤

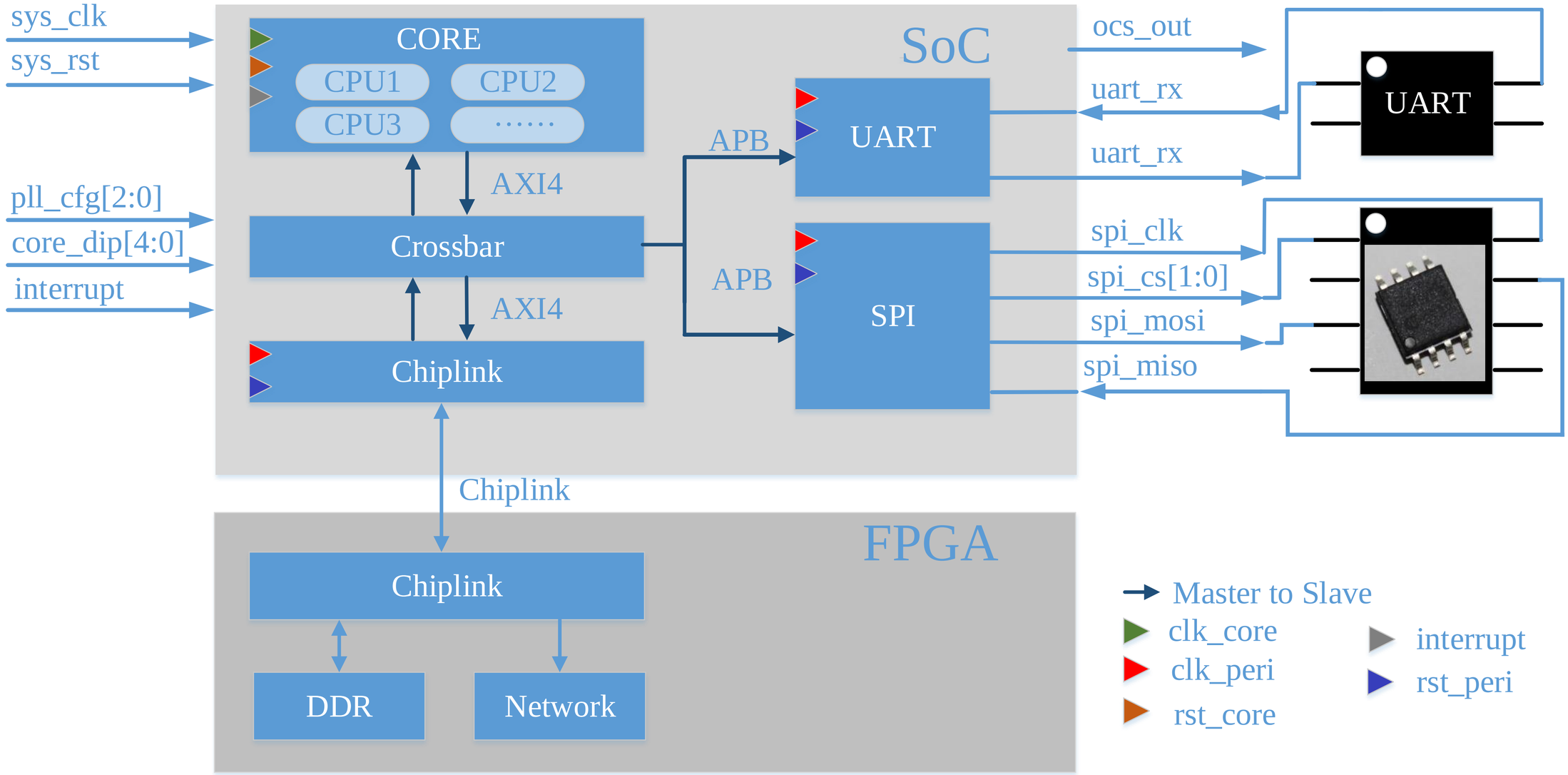
soc team：张文迪、谢王照祺、陈超、吴泽辉、刘钦超、聂晨

前端团队援助：陈璐、余子濠、唐浩晋

所属部门：PCNL-开源芯片

时　　间：2021/09/01

◆SOC总线框架


sys_clk
sys_rst
pll_cfg[2:0]
core_dip[4:0]
interrupt
CORE
CPU1
CPU2
CPU3
……
AXI4
Crossbar
AXI4
Chiplink
SoC
APB
APB
UART
SPI
ocs_out
uart_rx
uart_rx
spi_clk
spi_cs[1:0]
spi_mosi
spi_miso
UART
Chiplink
FPGA
Chiplink
DDR
Network
Master to Slave
clk_core
clk_peri
rst_core
interrupt
rst_peri

◆处理器

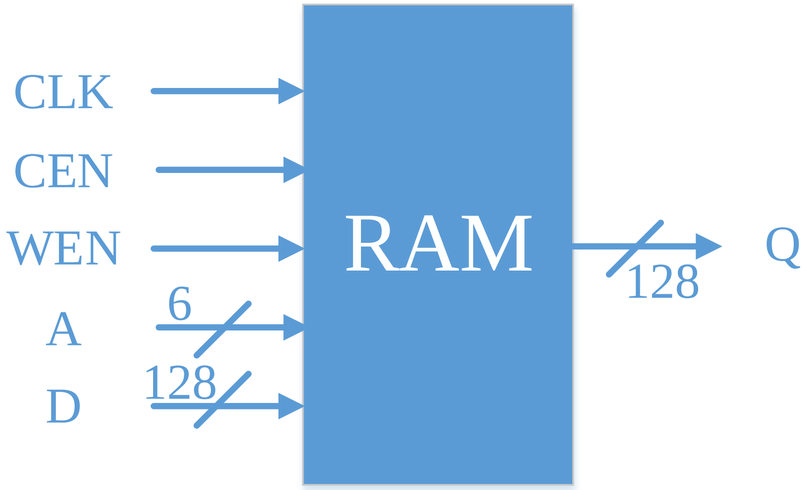
命名规范，见”ysyx3接口规范.xlsx”

cpu name		ysyx_编号（例如ysyx_210001）			
module name		ysyx_编号_module名			
时钟	input	clock			
复位(高电平有效)	input	reset			
中断	input	io_interrupt			
AXI总线					
master			slave		
input		io_master_awready	output		io_slave_awready
output		io_master_awvalid	input		io_slave_awvalid
output	[31:0]	io_master_awaddr	input	[31:0]	io_slave_awaddr
output	[3:0]	io_master_awid	input	[3:0]	io_slave_awid
output	[7:0]	io_master_awlen	input	[7:0]	io_slave_awlen
output	[2:0]	io_master_awsize	input	[2:0]	io_slave_awsize
output	[1:0]	io_master_awburst	input	[1:0]	io_slave_awburst
input		io_master_wready	output		io_slave_wready
output		io_master_wvalid	input		io_slave_wvalid
output	[63:0]	io_master_wdata	input	[63:0]	io_slave_wdata
output	[7:0]	io_master_wstrb	input	[7:0]	io_slave_wstrb
output		io_master_wlast	input		io_slave_wlast
output		io_master_bready	input		io_slave_bready
input		io_master_bvalid	output		io_slave_bvalid
input	[1:0]	io_master_bresp	output	[1:0]	io_slave_bresp
input	[3:0]	io_master_bid	output	[3:0]	io_slave_bid
input		io_master_arready	output		io_slave_arready
output		io_master_arvalid	input		io_slave_arvalid
output	[31:0]	io_master_araddr	input	[31:0]	io_slave_araddr
output	[3:0]	io_master_arid	input	[3:0]	io_slave_arid
output	[7:0]	io_master_arlen	input	[7:0]	io_slave_arlen
output	[2:0]	io_master_arsize	input	[2:0]	io_slave_arsize
output	[1:0]	io_master_arburst	input	[1:0]	io_slave_arburst
output		io_master_rready	input		io_slave_rready
input		io_master_rvalid	output		io_slave_rvalid
input	[1:0]	io_master_rresp	output	[1:0]	io_slave_rresp
input	[63:0]	io_master_rdata	output	[63:0]	io_slave_rdata
input		io_master_rlast	output		io_slave_rlast
input	[3:0]	io_master_rid	output	[3:0]	io_slave_rid

◆RAM替换

在SMIC110工艺下，1个D触发器需要**26个**晶体管，
1bit RAM只需要**6个**晶体管
还未计算控制逻辑（如地址译码逻辑，数据选择逻辑）
替换RAM后能大幅减小面积，改善后端物理设计结果

RAM模块的框图及引脚含义如下：



时钟（CLK）
使能端 (CEN)，低电平有效
写使能端 (WEN)，低电平有效
地址 (A[0:i])
数据输入(D[0:n])
数据输出(Q[0:n])。

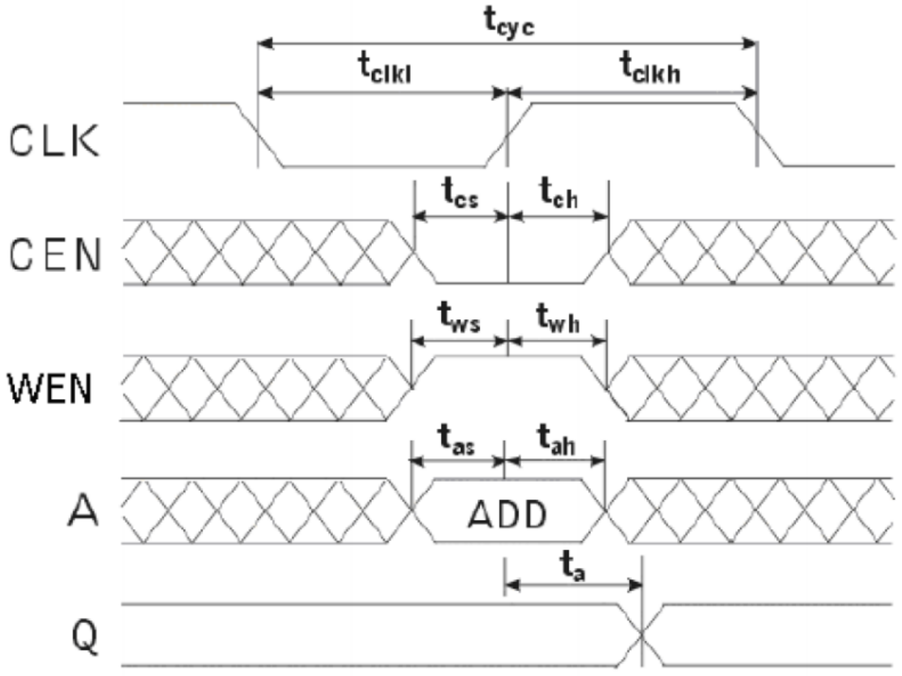
```
module S011HD1P_X32Y2D128 (
        Q,
        CLK,
        CEN,
        WEN,
        A,
        D);

    parameter Bits = 128;
    parameter Word_Depth = 64;
    parameter Add_Width = 6;

    output [Bits-1:0]       Q;
    input       CLK;
    input       CEN;
    input       WEN;
    input [Add_Width-1:0]  A;
    input [Bits-1:0]   D;
```

我们提供一个行为模型，
大家自行替换RAM

◆RAM和cache

- 仅用L1 cache，不允许使用L2 cache
- ICache 和DCache 的data array大小分别为4KB，合计8KB的RAM
- **仅支持单口RAM**
- Tag array不需要替换
- RAM原语的规格是128b*64，其中128b为宽度，64为深度
- RAM访问是同步的，由时钟的上升沿触发。
- 使能端 (CEN)、写使能端 (WEN) 、地址 (A[0:i]) 和数据输入(D[0:n]) 信号被锁存在时钟的上升沿

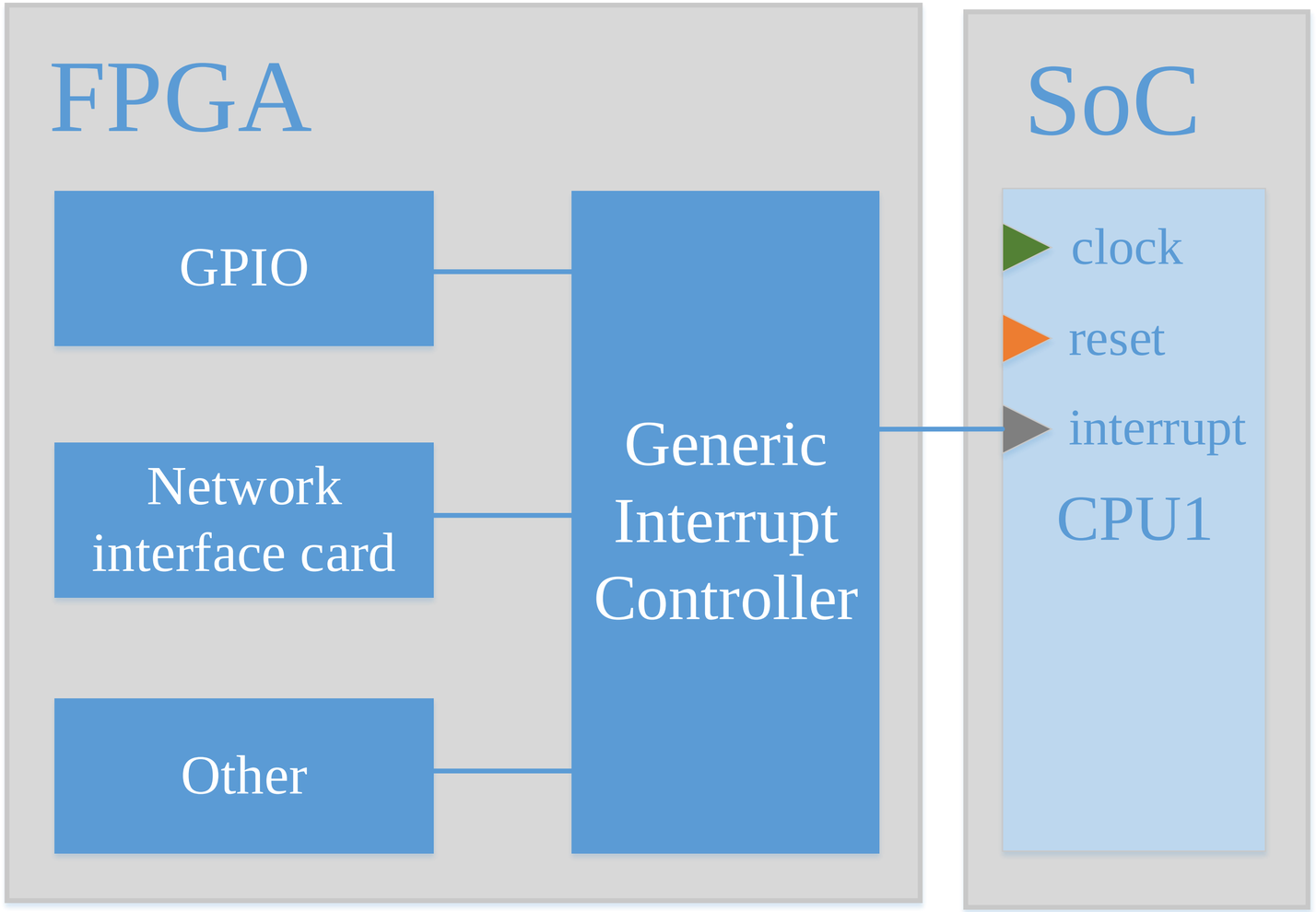
时序图



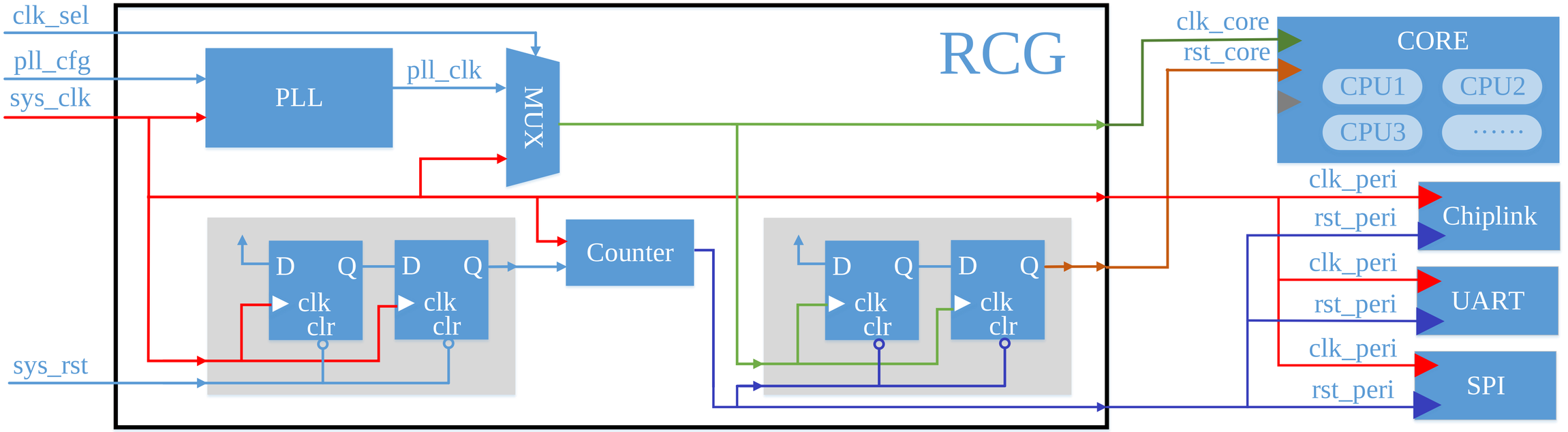
数据输出延迟为一周期

◆设备和中断的扩展

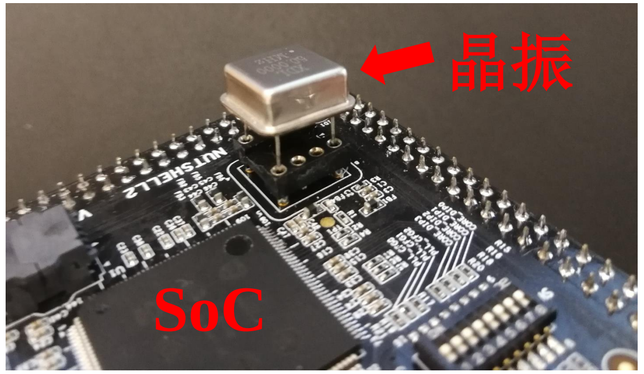
注意：拿到板卡后，学生需要自行在FPGA上实例化外设IP



◆时钟复位



- 晶振：20MHz, 50MHz, 100MHz
- pll_clk是可配置的
- MUX是手动切换clk_core, 提高容错率
- 异步复位同步撤离电路（RCG）：防止复位信号撤除时产生亚稳态信号



◆异步复位寄存器说明

1.确认异步复位寄存器是否同步撤离(防止出现亚稳态)。rtl仿真不存在亚稳态问题，但是综合后的网表仿真可能会出现亚稳态问题。

2.如果不清楚异步复位同步撤离的原理不建议使用。

3.10月7后提交初版代码后会返回报告，报告中会列出是否使用复位寄存器。（DC后可以检查，stdcell）

```
// 同步复位
always @(posedge clk)
    begin
        if(!rst) dout<=0;
    else dout<=din;
end
```

```
// 异步复位
always @(posedge clk or negedge rst_n)
begin
    if(!rst_n) a <= 1'b0;
    else a <= b;
end
```

◆外设介绍

- **UART (Universal Asynchronous Receiver/Transmitter)16550, and Registers list**

Name	Address	Width	Access	Description
Receiver Buffer	0	8	R	Receiver FIFO output
Transmitter Holding Register (THR)	0	8	W	Transmit FIFO input
Interrupt Enable	1	8	RW	Enable/Mask interrupts generated by the UART
Interrupt Identification	2	8	R	Get interrupt information
FIFO Control	2	8	W	Control FIFO options
Line Control Register	3	8	RW	Control connection
Modem Control	4	8	W	Controls modem
Line Status	5	8	R	Status information
Modem Status	6	8	R	Modem Status

Name	Address	Width	Access	Description
Divisor Latch Byte 1 (LSB)	0	8	RW	The LSB of the divisor latch
Divisor Latch Byte 2	1	8	RW	The MSB of the divisor latch

- **初始化 UART**

1. **关闭中断**
2. **配置LC寄存器，以使能 divisor latches**
3. **配置波特率**
4. **释放divisor latches**
5. **数据传输**

◆外设介绍

- **SPI(Serial Peripheral interface)控制器及其寄存器空间、和XIP模式**

Name	Address	Width	Access	Description
Rx0	0x00	32	R	Data receive register 0
Rx1	0x04	32	R	Data receive register 1
Rx2	0x08	32	R	Data receive register 2
Rx3	0x0c	32	R	Data receive register 3
Tx0	0x00	32	R/W	Data transmit register 0
Tx1	0x04	32	R/W	Data transmit register 1
Tx2	0x08	32	R/W	Data transmit register 2
Tx3	0x0c	32	R/W	Data transmit register 3
CTRL	0x10	32	R/W	Control and status register
DIVIDER	0x14	32	R/W	Clock divider register
SS	0x18	32	R/W	Slave select register

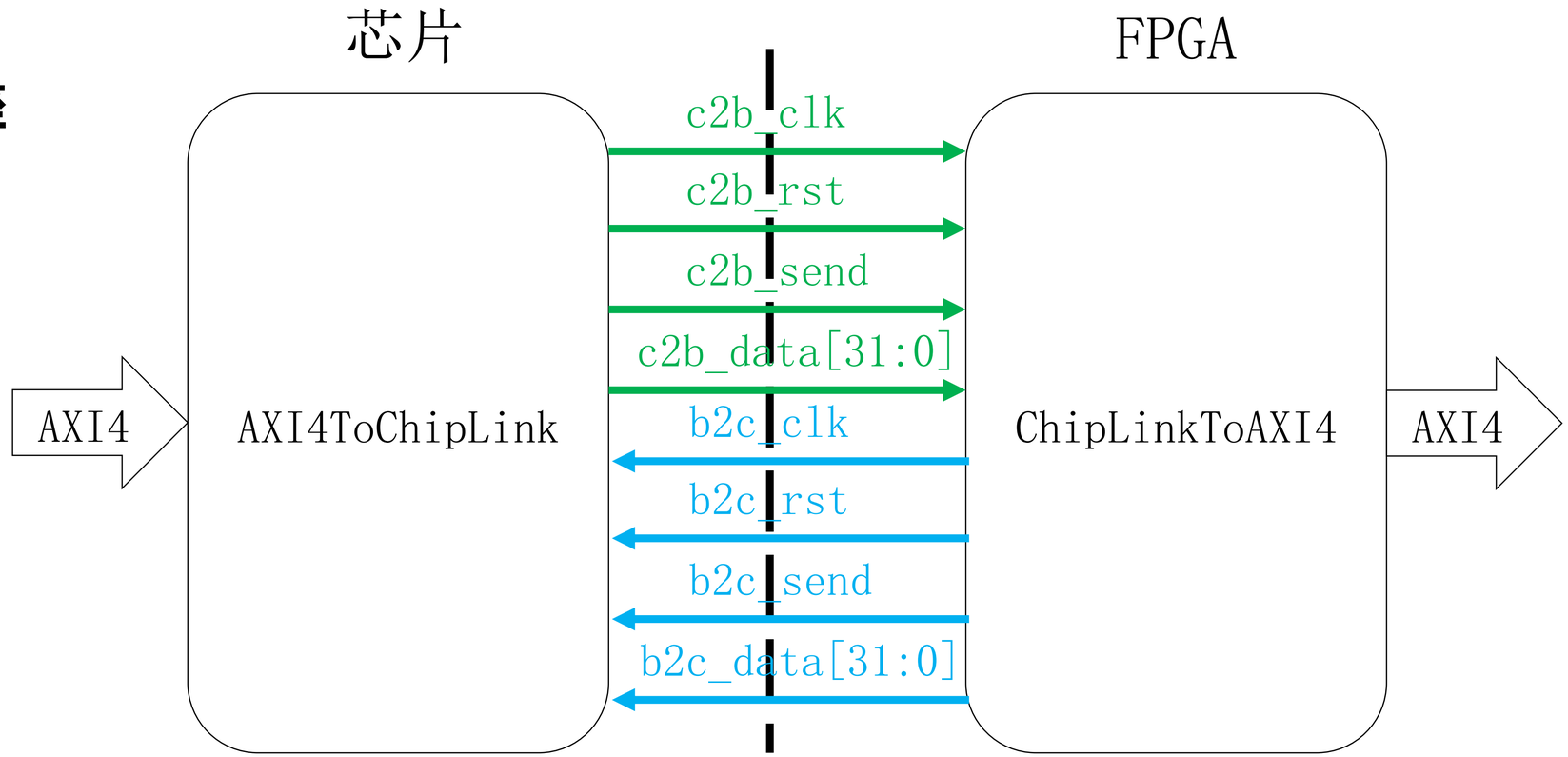
◆总线介绍

- **AXI4，APB，ChipLink**
  - **AXI4可以参考之前的讲座**

Signal	Source
PCLK	Clock source
PRESETn	System bus equivalent
PADDR	APB bridge
PSELx	APB bridge
PENABLE	APB bridge
PWRITE	APB bridge
PWDATA	APB bridge
PREADY	Slave interface
PRDATA	Slave interface
PSLVERR	Slave interface

APB协议

https://web.eecs.umich.edu/~prabal/teaching/eecs373-f12/readings/ARM_AMBA3_APB.pdf



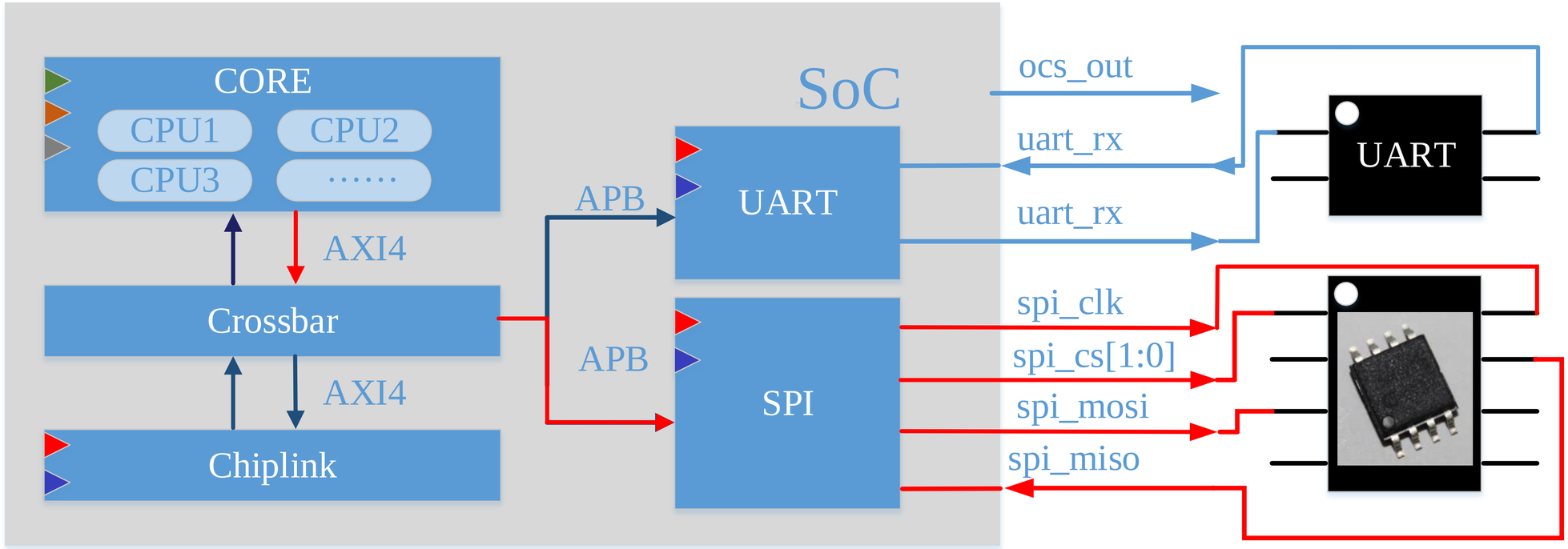
ChipLink工作原理：对AXI请求进行分片传输+重新组合

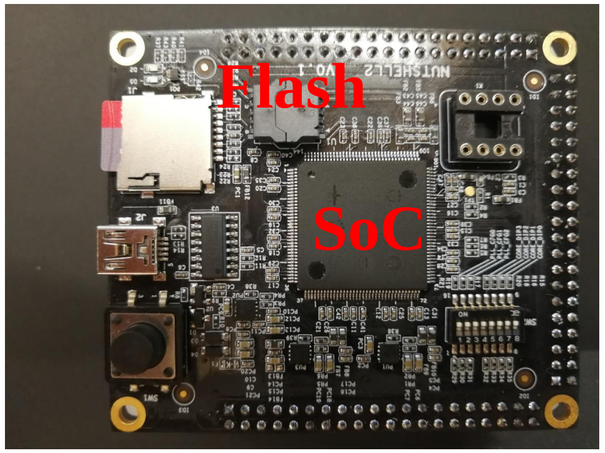
注意:

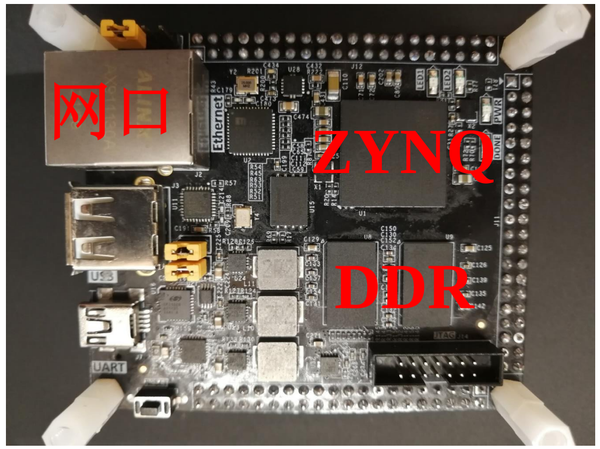
- ChipLink有sifive的开源实现，但没有文档
- ChipLink的带宽很低，200MHz时约60MB/s

选ChipLink原因：**易验证**，在**保证正确性**的前提下提供丰富的外设

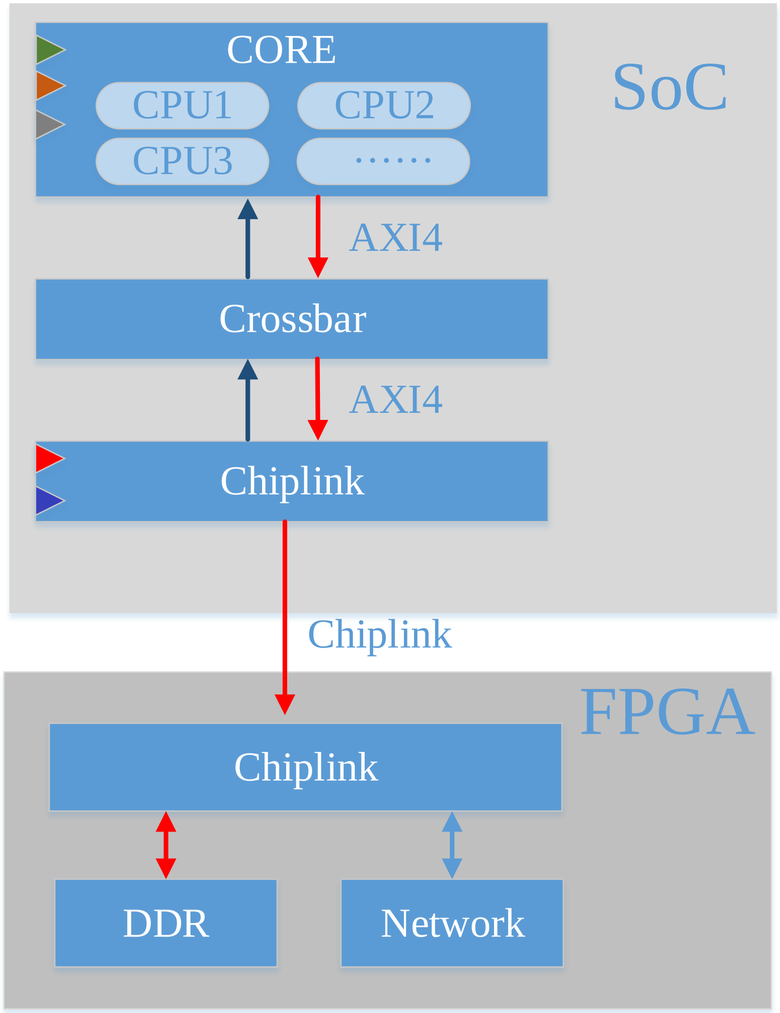
◆取指通路

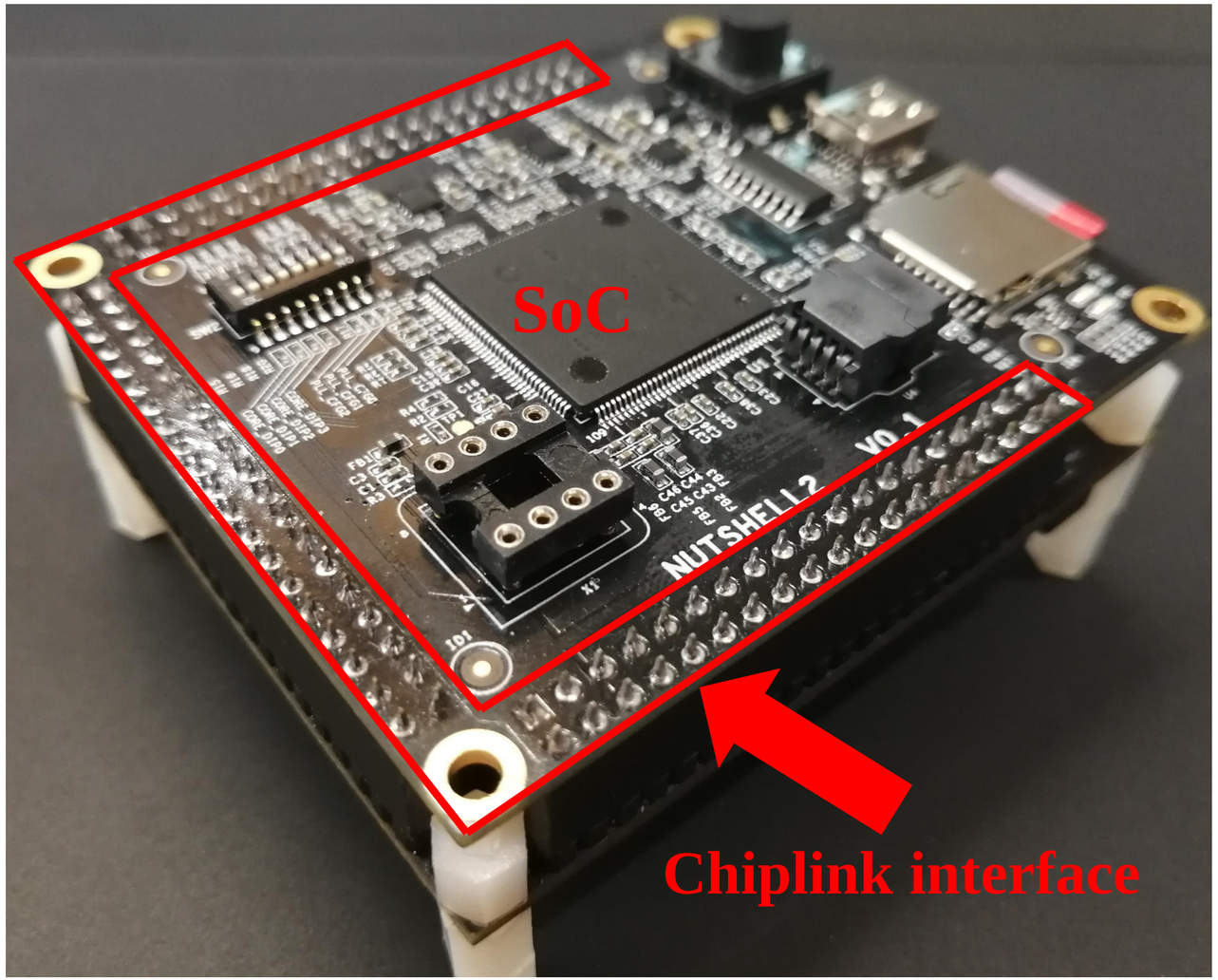

SoC
CORE
CPU1
CPU2
CPU3
……
AXI4
Crossbar
AXI4
Chiplink
APB
APB
UART
SPI
ocs_out
uart_rx
uart_rx
spi_clk
spi_cs[1:0]
spi_mosi
spi_miso
UART


Flash
SoC

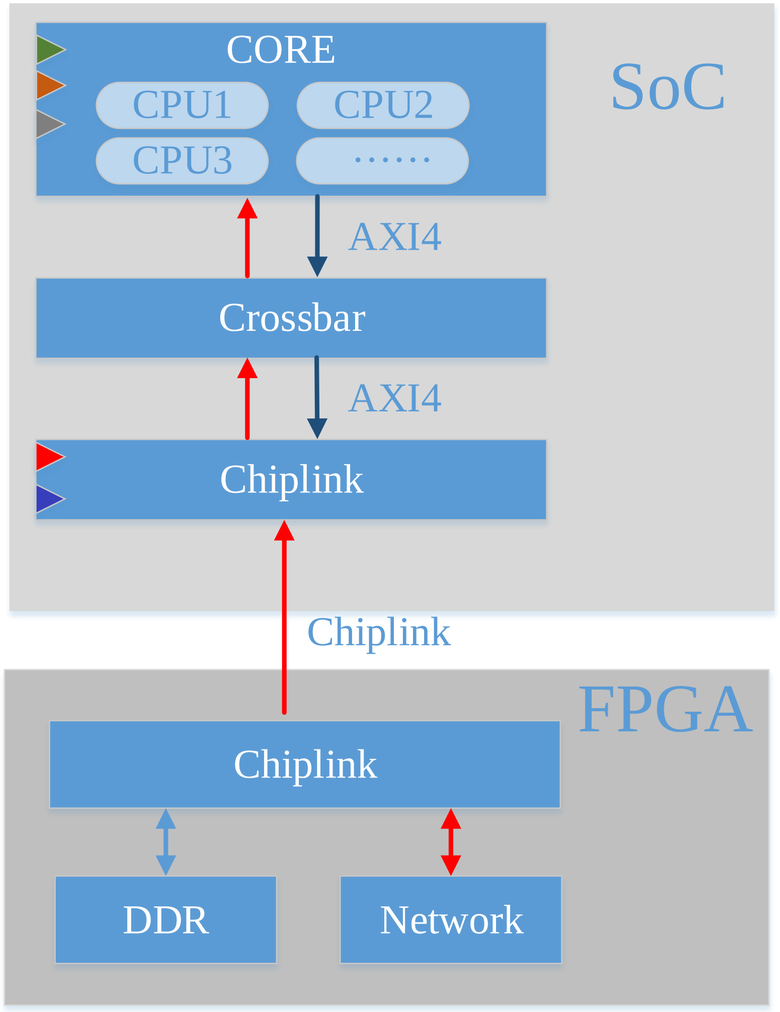

网口
ZYNQ
DDR

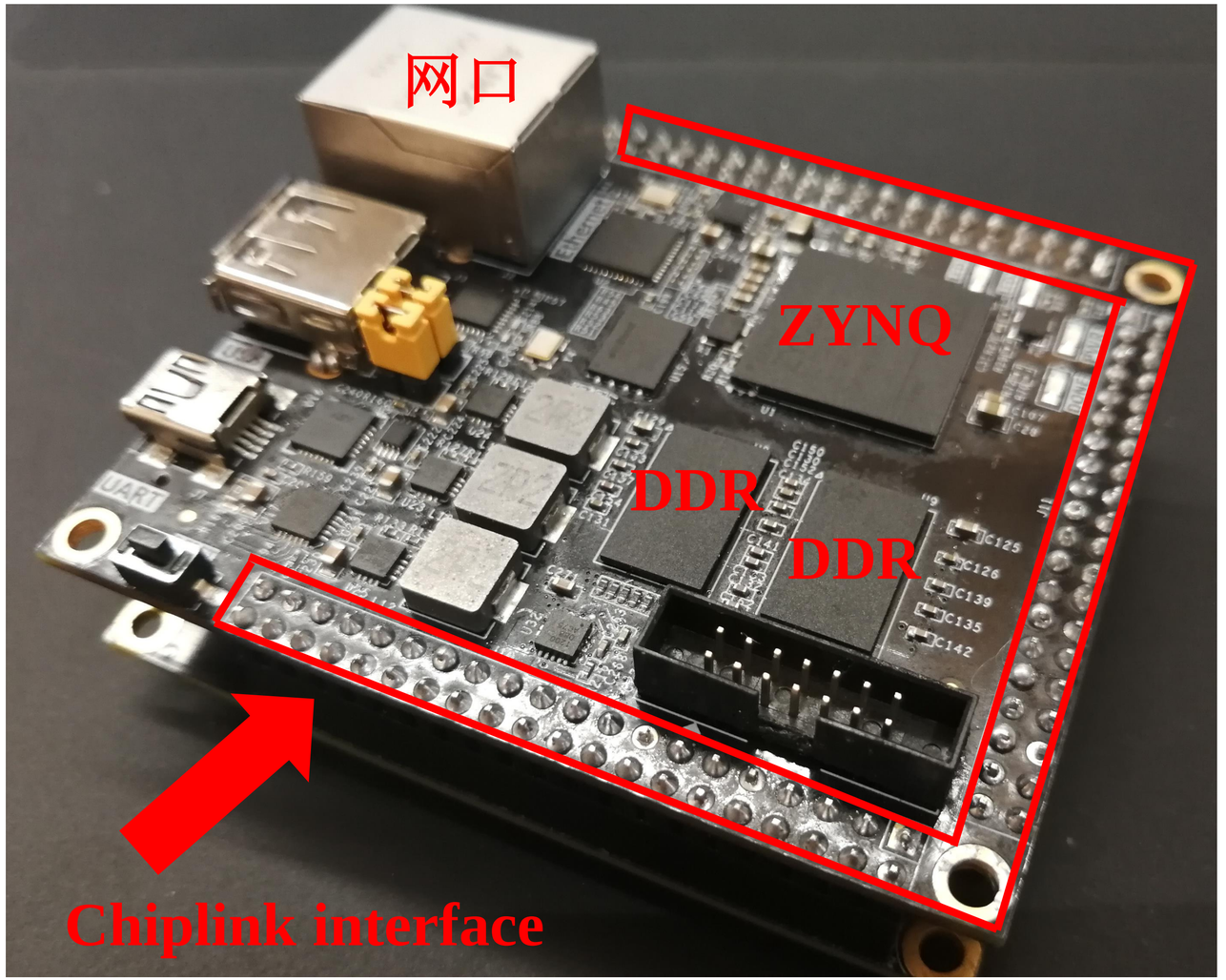
◆访存通路


CORE
CPU1
CPU2
CPU3
……
SoC
AXI4
Crossbar
AXI4
Chiplink
Chiplink
FPGA
Chiplink
DDR
Network


SoC
Chiplink interface

◆DMA通路


CORE
CPU1
CPU2
CPU3
……
SoC
AXI4
Crossbar
AXI4
Chiplink
Chiplink
FPGA
Chiplink
DDR
Network


网口
ZYNQ
DDR
DDR
Chiplink interface

◆地址空间分配

MMIO	0x8000_0000以下		
	reserve	0x0000_0000~0x01ff_ffff	
	CLINT (CPU内)	0x0200_0000~0x0200_ffff	与QEMU一致
	reserve	0x0201_0000~0x0fff_ffff	
	UART16550	0x1000_0000~0x1000_0fff	与QEMU一致
	SPI controller	0x1000_1000~0x1000_1fff	
	reserve	0x1000_2000~0x2fff_ffff	
	SPI-flash XIP mode	0x3000_0000~0x3fff_ffff	
	ChipLink MMIO	0x4000_0000~0x7fff_ffff	
MEM	0x8000_0000以上		与QEMU一致

- 复位PC值是0x3000_0000，第一条指令从flash中取出
- CLINT模块位于CPU内部，SoC不提供，需要大家自行实现（参考之前的报告）
- 若需要接入其它设备（如PLIC），请在处理器内部接入，并将地址分配到预留空间中
- 部分外设与QEMU保持一致，可直接运行相同的RT-Thread

◆第一个程序的加载

做Cache的同学需要关注

第一程序（XIP flash）	第二程序（memory）	ICache	DCache	Cache一致性	真实芯片	仿真	仿真速度	备注
测试程序	无	不使用	使用	不需要	yes	yes	慢	嵌入式系统工作方式
loader	测试程序	使用	使用	需要	yes	yes	中等	一般系统工作方式
跳转指令	测试程序	使用	使用	需要	no	yes	快	SoC团队仿真方式
无	测试程序	使用	使用	需要	no	yes	快	目前大家仿真方式

注意：

- Flash的XIP地址空间不能写
- 所有外设的寄存器都不大于4字节，因此不能向外设发送8字节的请求
- 不能向外设发送burst请求
- 我们提供的测试程序不会有上述行为，若发现错误，请检查RTL实现的正确性

◆XIP flash模式的RT-Thread初始化工作

- flash不支持写入，在初始化时需要将.data节从flash复制到memory中

```
# 80004750 <rt_object_container>
```

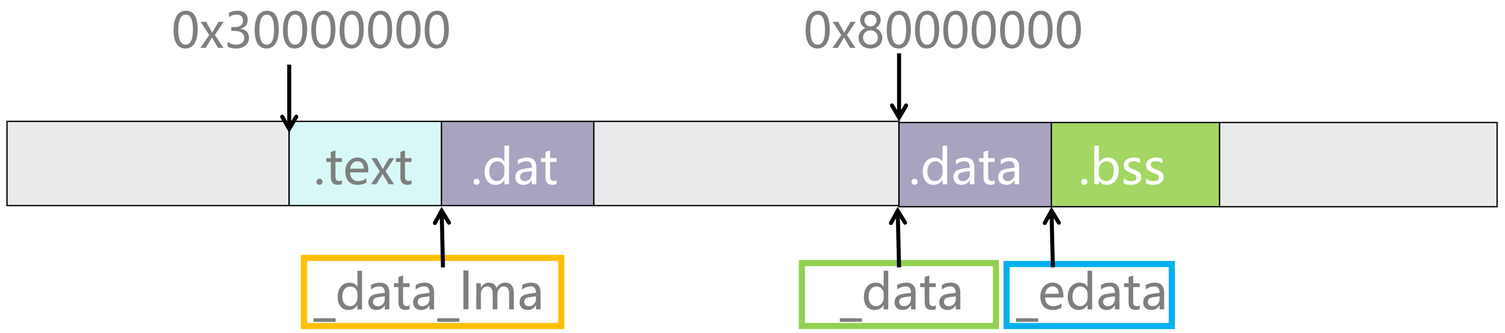
```
300054e4:   27070713   addi    a4,a4,624
300054e8:   fec42783   lw  a5,-20(s0)
300054ec:   00579793   slli    a5,a5,0x5
300054f0:   00f707b3   add a5,a4,a5
300054f4:   0007a703   lw  a4,0(a5)
```

```
    /* Load data section */
    la a0, _data_lma
    la a1, _data
    la a2, _edata
    bgeu a1, a2, 2f
1:
    lw t0, (a0)
    sw t0, (a1)
    addi a0, a0, 4
    addi a1, a1, 4
    bltu a1, a2, 1b
2:

    /* Clear bss section */
    la a0, __bss_start
    la a1, _end
    bgeu a0, a1, 2f
1:
    sw zero, (a0)
    addi a0, a0, 4
    bltu a0, a1, 1b
2:
```

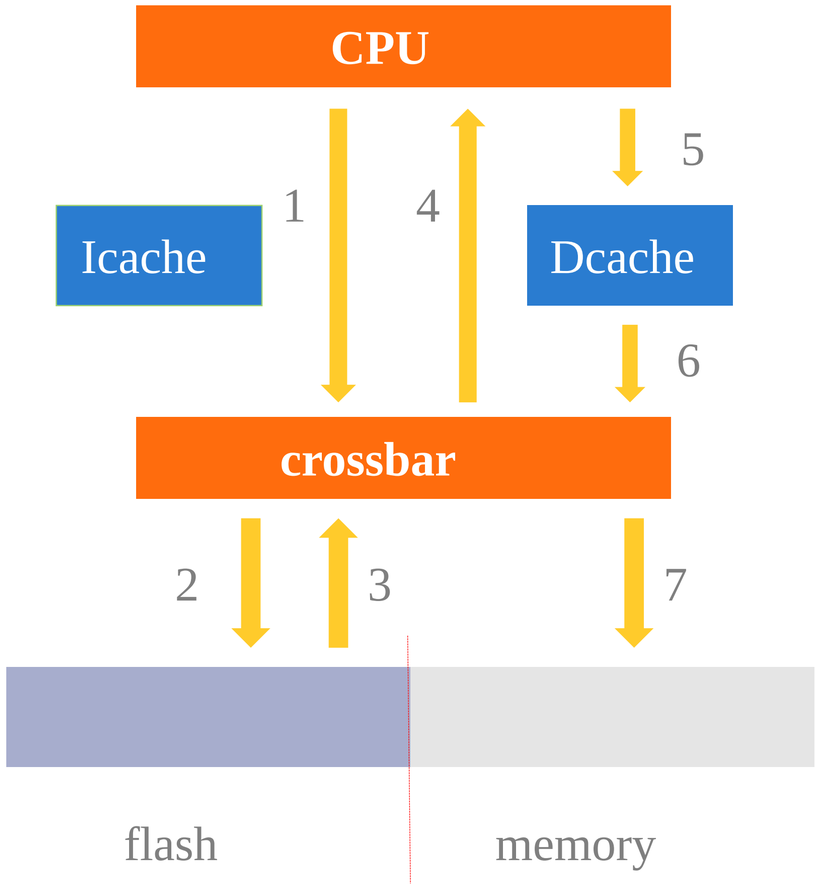
memcpy(_data, _data_lma, _edata-_data)

memset(_bss_start, 0, _end-_bss_start)

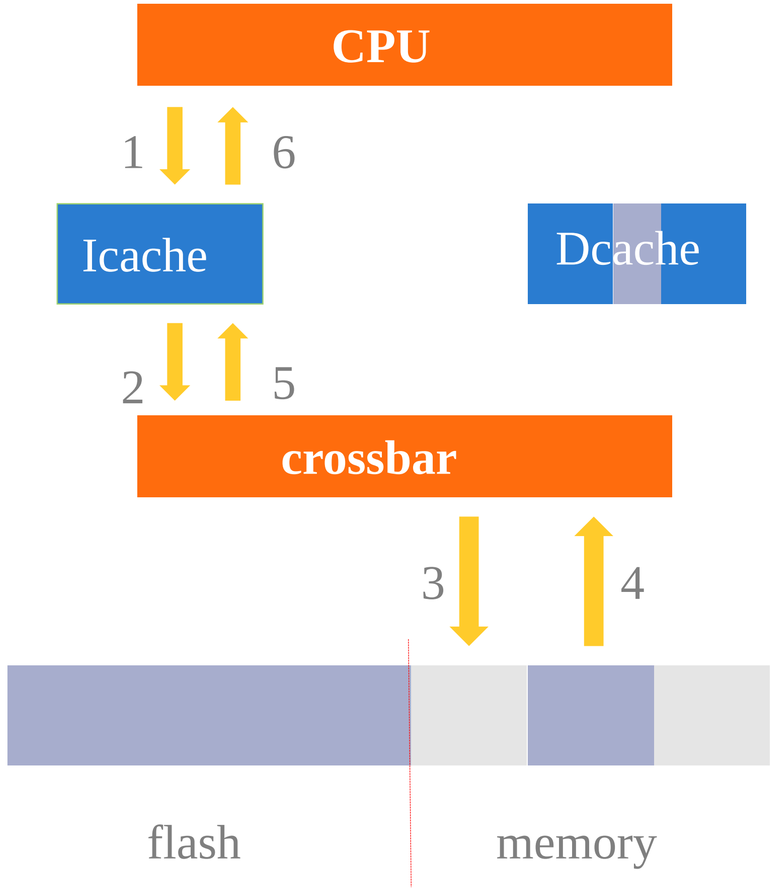
```
.dalign         :
{
  . = ALIGN(4);
  PROVIDE( _data = . );
} >ram AT>flash :ram_init

. = ALIGN(4);
.data           :
{
  /* section information for modules */
  *(.rdata)
  *(.rodata .rodata.*)
  *(.data .data.*)
  . = ALIGN(8);
  PROVIDE( __global_pointer$ = . + 0x800 );
  *(.sdata .sdata.*)
} >ram AT>flash :ram_init

. = ALIGN(4);
PROVIDE( _edata = . );
PROVIDE( __bss_start = . );
.bss            :
{
  *(.sbss*)
  *(.bss .bss.*)
  *(COMMON)
  . = ALIGN(4);
} >ram AT>ram :ram

. = ALIGN(8);
```



◆Cache一致性问题



loader执行时从flash将代码以数据拷贝的方式拷到内存，这部分代码可能仍然位于dcache



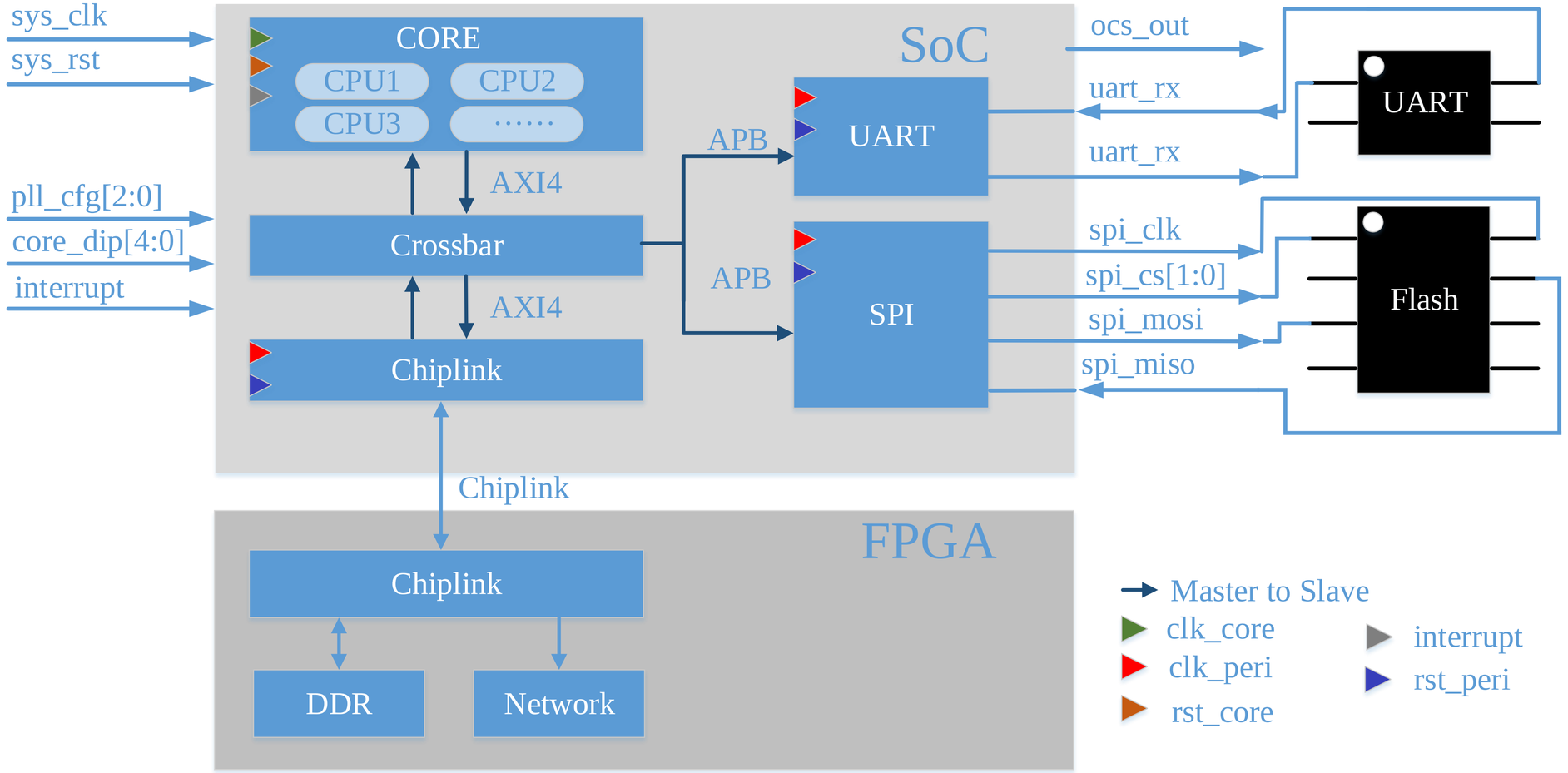
测试程序首次执行时，发生icache miss，从memory取指，但此时指令可能位于dcache，因此无法取得正确指令

◆可在verilator中仿真的SoC

https://github.com/OSCPU/ysyxSoC

与流片SoC区别：

- 没有跨时钟域和异步桥
- 没有PLL



◆龙芯杯作品接入一生一芯

需要额外开展如下的移植工作：

1. 重写IP：把处理器内部调用vivado的ip核进行手工重写，它们是不能流片的
2. 确认Cache中RAM实现：Cache的data array不能使用distributive ram（读延迟为0），但可以使用block ram（读延迟为1），因为流片使用的标准单元库与FPGA有所区别，没有distributive ram的原语，它们会被综合成触发器
3. 添加外设地址转换桥：在处理器中添加一个地址转换模块，把mips看到的外设地址映射到SoC上真正的外设地址，例如SoC上uart的地址是0x10000000，mips看到的uart地址是0xb0000000，那么在地址转换模块中减去两者的偏移量即可
4. 修改总线协议：SoC接口是AXI4协议，如果处理器使用的总线协议是AXI3，则需要修改成AXI4
5. 手写多选一crossbar：手写一个crossbar把AXI4通道合并成1个，接入一生一芯的SoC
6. 编译运行RT-Thread：编译一个mips版本的rt-thread操作系统，在仿真环境中跑起来,它不需要虚存的支持

◆SoC集成任务CheckList

命名规范

- CPU代码合并到一个.v文件，文件名为ysyx_自己学号后六位，如ysyx_210888.v
  - 在Linux上可通过cat命令实现
- 修改CPU顶层命名格式为ysyx_自己学号后六位，如ysyx_210888
- 按照”ysyx3接口规范.xlsx”修改CPU顶层端口名
- CPU内module命名格式为ysyx_自己学号后六位_module name
  - 如module ALU 改为module ysyx_210888_ ALU
  - Chisel福利：我们提供一个firrtl transform来自动添加模块名前缀
    - 见ysyxSoC项目中的README
- 运行命名规范自查脚本

◆SoC集成任务CheckList（续）

CPU内部修改

- 若实现了cache，需要对data array进行RAM替换
- 若用Verilog开发，确认异步复位寄存器是否已去除或同步释放
  - Chisel福利：Chisel默认生成同步复位寄存器

流程要求，尽最大可能降低流片风险

Verilator仿真

- 接入ysyxSoC项目
- 为Verilator添加-Wall编译选项，清除报告的所有Warning
  - Chisel福利：Chisel生成的Verilog基本上符合规范
- 运行ysyxSoC项目提供的测试程序：hello、memtest、RT-Thread，每个程序分别有flash版本和loader版本

Yosys综合（具体操作在后续开源EDA报告中介绍）

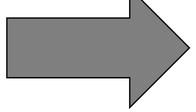
- 通过Yosys综合CPU顶层，清除报告的所有Warning

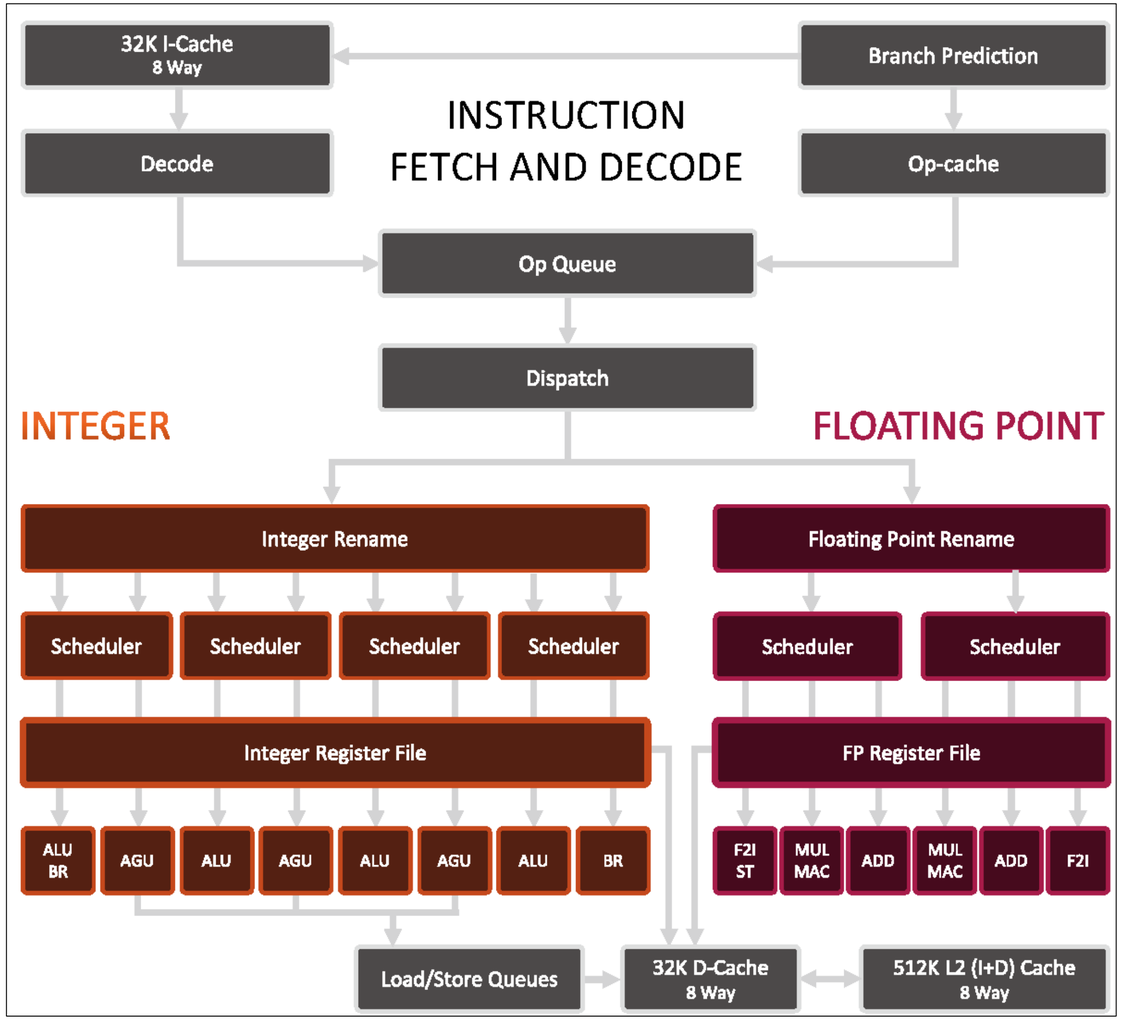
◆SoC集成任务CheckList（续）

提交

- 一份CPU的.v文件
- 一份带数据流向的处理器架构图
  - **IC后端要进行FloorPlan**

参考示例：HOTCHIPS-2021 AMD Next Generation “Zen 3” Core





◆代码提交流程

2021年9月15日正式开始提交代码

提交代码之前，一定要完成CheckList中的所有内容

soc team不会修改提交的任何代码，
大家要对自己编写的代码负责！！！

然后按顺序进行以下步骤：

早提交 早发现 早修改

1、向本组组长申请提交code
2、进行答辩（助教+老师审核）
3、正式git提交code（定期开启提交通道）
4、前端负责人把cpu代码统一发给soc team
5、soc->集成+验证

soc team提供资料：本次报告PDF和视频、ysyx3接口规范.xlsx、规范检测脚本、AXI手册

谢谢

THANK YOU

◆特别声明

https://github.com/OSCPU/ysyxSoC

该文档中的信息都已经更新到以上链接readme中，请以该链接中的readme信息为准。此文档不再更新，如有疑问，请及时和助教联系。