## SoC设计

图1表示整体SoC架构设计，处理器核有两组访问接口：IBUS指令读取接口，DBUS存储即外设访问接口，执行load/store指令。两组接口均为AHB总线master接口。从设备包括：1）ITCM指令存储单元，IBUS接口只访问ITCM模块进行取指；2）DTCM数据存储单元，主要是load、store指令的存储访问区间；3）UART、TIMER外设，该模块也是通过DBUS接口进行相关寄存器配置。

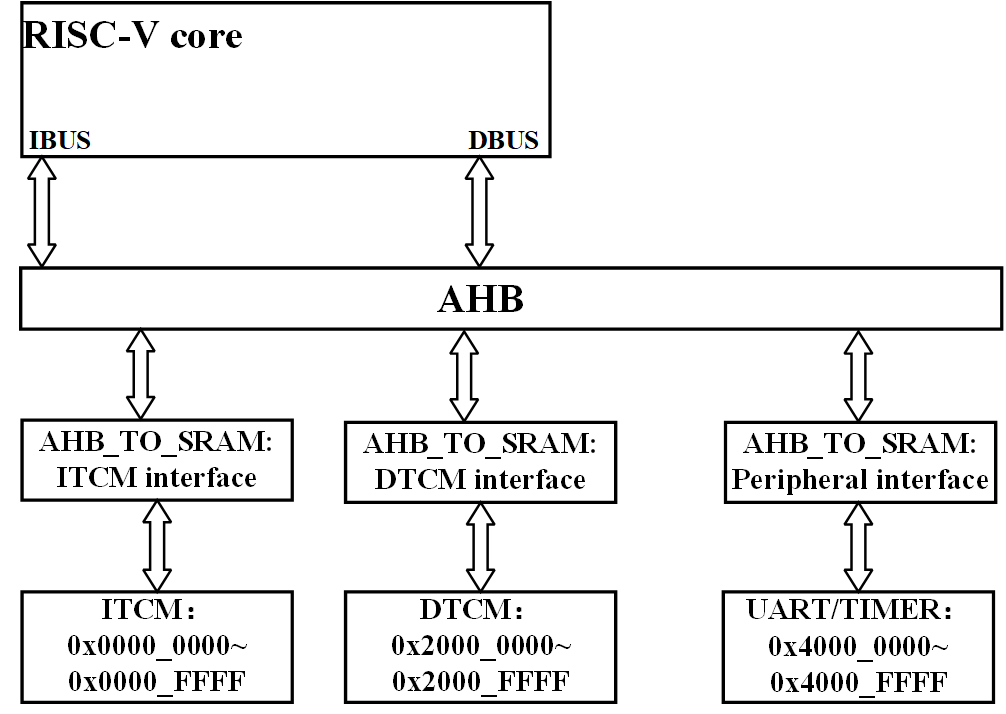
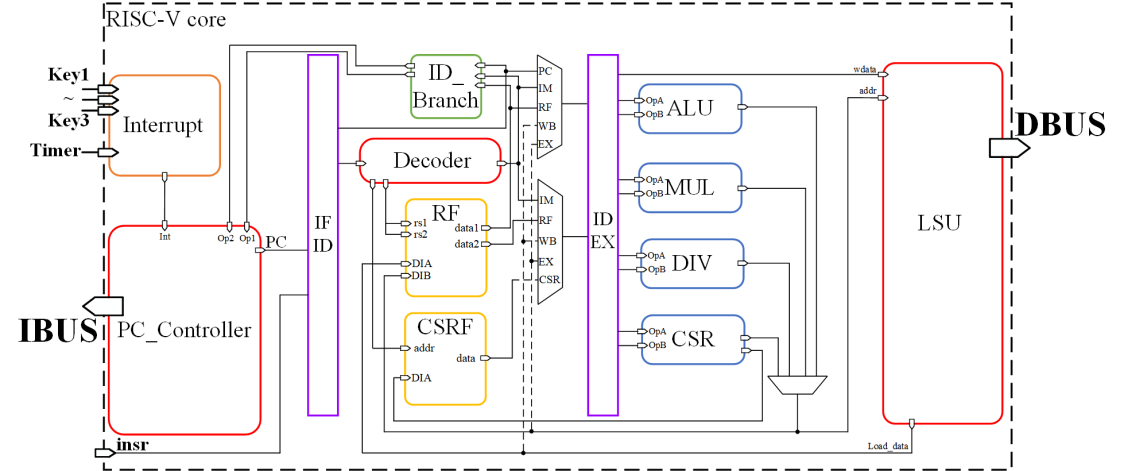


图1. SoC架构

## 处理器核架构

图2.表示所设计的处理器核架构，一款32位、3级流水架构、顺序执行顺序写回的risc-v处理器，支持基本整数指令集I、乘除法指令集M，支持按键、定时中断。



### 取指IF

三级流水：取指、译码、执行（访存）写回。取指模块主要负责运算下一周期PC寄存器中的指令地址：1）没有中断触发时，其PC地址运算如下。其中wr\_stop信号来自ID译码模块和RegFile模块，表示指令间发生了RAW相关性，需要流水线取指停顿。div\_alu\_time表示执行模块正在执行除法指令，由于除法指令执行周期不定长（1-32 clks），为长周期指令，执行除法时需要流水线停顿。i\_hready表示通过IBUS接口取PC地址处的指令时，总线还没有准备好数据，需要取指模块停顿等待。d\_hready表示在执行load、store指令时，访问某一存储地址或是外设时，DBUS总线还没有准备好读回的数据（load）或是接收写出数据（store），此时执行模块需要停顿等待，导致取指模块也要停顿等待。d\_ITCM\_r表示DBUS访问的存储区间是ITCM，此时IBSU接口没办法访问ITCM取指，需要停顿取指，优先DBUS接口的访问，当DBUS接口取到或是写回相应数据后，再恢复IBUS对ITCM的访问，开始取指。以上信号表示可能发生的各种流水线取指停顿的情况，其中任何一种发生都要暂停流水线取指，此时PC取值PC\_r，即保持上一个周期的PC值不变。当取指没有发生停顿时，执行Add2PC\_op1+Add2PC\_op2，即运算得到下一次的正确取指地址。start\_2信号在复位信号失效后的一个周期后一直拉高，用来在复位后经过一个周期再开始流水线运行。`ADDRESS\_rst表示复位时PC地址值，设定为第一条指令的起始代码，即程序的start入口地址。

*//没有异常发生时，PC每次正常取值，包括顺序取指和跳转取指*

assign PC       = |{wr\_stop, div\_alu\_time, ~i\_hready, ~d\_hready, d\_ITCM\_r} ? PC\_r : (start\_2 ? (Add2PC\_op1 + Add2PC\_op2)

                  : `ADDRESS\_rst );

assign normal\_PC= PC;

Add2PC\_op1、Add2PC\_op2表示程序正常运行下一条指令的地址，可能为顺序取指或者分支跳转取指。if\_Jump信号为真表示上一条指令为跳转指令（包括分支预测指令，需要跳转），计算PC的两个操作数来自prdt\_pc\_add\_op1、prdt\_pc\_add\_op2来自译码模块；否则顺序取指，PC计算值为PC\_r+4. 如果当前周期取到指令为分支预测指令或是跳转指令，则下一时钟周期在译码模块即可执行完成该条指令，得到是否需要跳转以及跳转译码信息，传递给取指IF模块，用于计算下一条指令地址。也就是该处理器设计所有分支预测指令都会成功执行，没有预测失败情况。

wire [`PC\_WIDTH-1:0] Add2PC\_op2 = if\_Jump ? prdt\_pc\_add\_op2 : `PC\_add\_insr;

wire [`PC\_WIDTH-1:0] Add2PC\_op1 = if\_Jump ? prdt\_pc\_add\_op1 : PC\_r;

取指上面是正常运算情况。实际本模块中取指分为三种情况：进入中断、退出中断、正常运算PC地址。进入中断信号trap\_entry\_en来自中断控制模块interrupt\_ctrl，检测到按键上升沿或是定时器定时完成；触发中断后的中断例程入口地址trap\_entry\_pc来自CSRFile，需要在系统start.s文件中提前初始化好中断例程入口地址相应寄存器。退出中断信号trap\_exit\_en来自译码模块mret指令，检测到该信号后PC计算值restore\_pc表示进入中断前程序正常执行位置的下一条指令，在进入中断前，自动将下一条指令位置PC+4存入相应CSR寄存器。

wire [`PC\_WIDTH-1:0] PC\_cur;

assign PC\_cur   = trap\_entry\_en ? trap\_entry\_pc :

                  trap\_exit\_en  ? restore\_pc

                  : PC;

### 译码ID

译码模块将来自ITCM的指令进行译码，得到相关信息，传给下一级EX\_top2模块执行。rv32\_insr表示传给ID模块待译码的指令。如果此时正在执行除法指令期间，div\_alu\_time有效；或者load、store指令访问DBUS总线需要等待时钟周期传递数据，d\_hready为0，则保持ID模块的指令为上一周期的指令rv32\_insr\_mem\_r. 如果IF模块访问IBUS总线没有得到指令数据，i\_hready为0，或者DBUS访问ITCM指令地址区间，导致IBUS访问指令停顿，ID模块译码`ADDIR0空转指令，即x0=x0+0. 该指令不会改变处理器的状态。正常流水线运行情况下，ID模块译码IBUS总线读出的指令rv32\_insr\_mem. 前两种特殊停顿情况译码指令有所不同，第一种需要保持ID模块指令不变，因为这种情况的指令（除法、访问指令）执行周期长，还未执行结束，需要ID模块一直保持该指令信息，供给EX\_top2模块；第二种情况译码指令为空转指令，是因为取指模块没有取到有效指令，后面的ID、EX流水级只能跑空转指令。

需要注意的是，对于跳转指令和分支预测指令，直接在ID模块执行完成，得到跳转信息及跳转操作数，传给IF模块。

assign rv32\_insr   = (div\_alu\_time | ~d\_hready) ? rv32\_insr\_mem\_r

                   : ((~i\_hready) | d\_ITCM\_r) ? `ADDIR0 : rv32\_insr\_mem;

### 执行EX\_top2

执行模块EX\_top2根据ID模块传来的译码信息，执行指令，分为三种情况：1）普通逻辑运算指令、2）除法指令、3）访存指令。第1）种还包括乘法指令、CSR指令，运算延迟为1个时钟周期；第2）种为不定时钟周期数，与操作数有个，1-32个时钟周期；第三种需要将load/store指令信息配置写入DBUS总线接口，要配合AHB master端接口时序（地址相位和数据相位），运算延迟也不定，和DBUS访问的从设备有关，d\_hready拉高时表示从设备端传来数据信息或是准备好接收数据，即访存指令最后一个周期。

### 寄存器对RegFile

该模块负责寄存器的读出、写入以及RAW相关检测。读出数据接口接收来自译码ID模块的寄存器地址信号，在当前周期立即给出读出数据结果。写入数据接口包括两组，一组是指令执行结果写回，一般在指令第三个周期执行完成便立即发送写回数据及控制信号，下个周期写入有效（除法指令除外，为长周期执行指令）；另一组是load指令写回接口，load指令也是长周期执行指令。在同一周期内能够同时写入两个接口，提高执行效率。

对RAW检测，也是主要针对长周期执行指令，普通指令间不会发生相关性，因为普通指令三个周期写回目的寄存器，支持运算结果前递旁路到运算模块。

### 中断检测模块interr\_ctrl

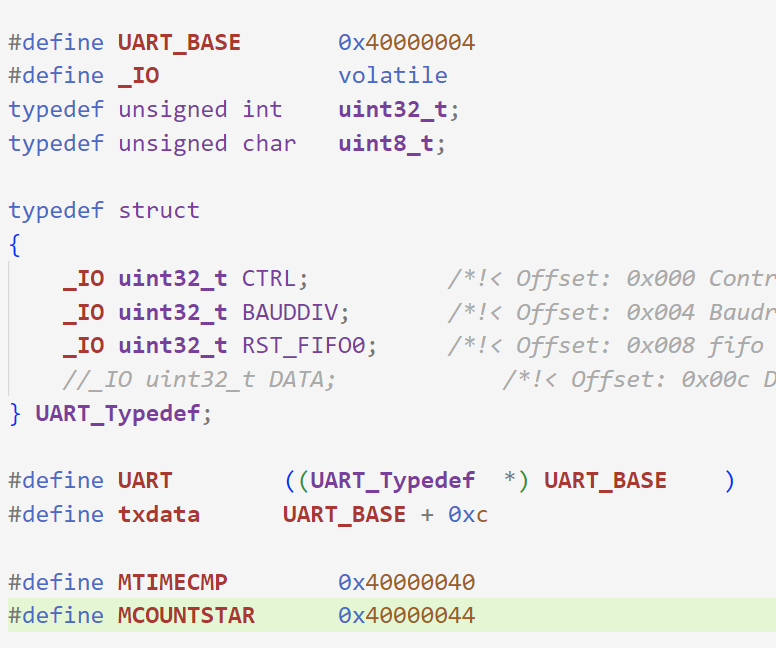
该模块检测中断信号源：三个按键中断（按键上升沿检测、定时器检测、SD卡读取完成信号，SD卡读取完成信号目前版本用不上），输出对应的中断原因int\_index、进入中断例程信号trap\_entry\_en、退出中断trap\_exit\_en，分别输入给CSR寄存器堆、取指IF模块等。

### CSRFile

主要是完成CSR寄存器的写回，以及一些特殊寄存器的赋值，包括中断使能、中断例程入口地址、进入中断时保留当前PC寄存器地址等等。

### UART、TIMER外设

UART外设需要配置波特率、数据位等寄存器。CPU通过store指令写入相应地址区间寄存器完成配置，数据位配置地址为0x4000\_0004，波特率配置地址为0x4000\_0008，需要配合uart软件代码进行初始化。串口发送地址为0x4000\_0010。定时器定时寄存器MTIMECMP配置地址为0x4000\_0040。当0x4000\_0044地址处寄存器MCOUNTSTAR配置后，从0开始计数，计数到MTIMECMP时触发定时中断，该信号传送给处理器核。



## Modelsim、vivado仿真

在tb.v文件中，将生成机器二进制码文件读入ITCM中，即riscv\_boot行读入本地位置的二进制文件（riscv gcc工具链编译c文件生成的可执行文件），修改执行的代码即修改这一行。执行过程：将程序初始化存入ITCM中，系统启动引导start.s文件会将程序中的数据段.data搬移到DTCM中，再跳转进入main函数。这样在以后执行访存指令时，DBUS接口和IBUS接口访问不同的地址区间，不会影响流水线取指停顿，提高执行效率。

Vivado版本则将原来代码的存储模块ITCM、DTCM（cmsdk\_fpga\_sram）换成了vivado的ram ip核，综合性能更好，时序都是一样的。下面的coe内容用于生成初始化vivado ram ip核的coe文件，将二进制文件转换成coe文件，初始化时填入ITCM的ram ip核。使用vivado自带ram ip核时，注意时序，要求当前周期输入地址，下个周期即可读出数据，因此使用vivado的ram ip核时，要去掉一些优化时序的默认选项。

