蜂鸟 E203 开源 SoC 介绍

Content

Tables

Figures

1 Preface

1.1 Revision History

Date	Version	Author	Change Summary
Oct 20,2018	1.0	Bob Hu	Initial Version

1.2 Reference Documentation

https://github.com/SI-RISCV/e200_opensource/tree/master/doc/SiFive-E300-platform-reference-manual-v1.0.1.pdf
https://github.com/SI-RISCV/e200_opensource/tree/master/doc/SiFive-E310-G000-manual-v1.0.1.pdf
https://github.com/SI-RISCV/e200_opensource/tree/master/doc/SiFive-E3-Coreplex-v1.2.pdf
https://github.com/SI-RISCV/e200_opensource/tree/master/doc/I2CMasterwithWISHBONEBusInterface-Documentation.pdf
中文书籍《手把手教你设计 CPU：RISC-V 处理器篇》
中文书籍《RISC-V 架构与嵌入式开发快速入门》

2 总体介绍

本文介绍全开源的蜂鸟 E203 RISC-V 处理器核以及配套开源 SoC，为了方便描述，本文档中将此 SoC 简称为“蜂鸟 E203 SoC”。

注意：

- 本文档对蜂鸟 E203 内核以及 RISC-V 指令集架构的介绍尚不够详细，在中文书籍《手把手教你设计 CPU：RISC-V 处理器篇》或者《RISC-V 架构与嵌入式开发快速入门》中对其进行深入浅出地系统的讲解，感兴趣的用户可以自行搜索书籍。
- 本文档对 SoC 的各外设的介绍尚不够详细，在中文书籍《RISC-V 架构与嵌入式开发快速入门》中进行了深入浅出的系统讲解。感兴趣的用户可以自行搜索此书。

注意：

- 本 SoC 的代码全开源，有关其代码结构的详情，请参见《蜂鸟 E203 快速上手介绍》。
- 本开源 SoC 主要面向教育教学和爱好者领域，源代码全部开放。理论上可以用于商业目的，但是不保证其商用质量和服务。
- The main purpose of this open-sourced core is to be used by students/university/research and entry-level-beginners, hence, the commercial quality (bug-free) and service of this core is not not not warranted!!!

2.1 总体特性

本 SoC 总体特性如下：

- 使用全开源的蜂鸟 E203 处理器核，此处理器核包括如下特性：
  - 超低功耗 2 级流水线处理器核
  - 64KB ITCM，64KB DTCM
  - 有关蜂鸟 E203 的更多信息参见第 3 章
- 为了尽量的共享当前 RISC-V 的软件生态，将尽可能的复用 SiFive 公司开源的 Freedom 310（HiFive1 开发板所使用）SoC，包括：
  - 对其已有的 IP 进行复用兼容
  - 对其总线地址分配进行兼容
- 在兼容 Freedom 310 SoC 的基础上，增加如下 IP，使其 SoC 功能更加丰富：
  - I2C Master

2.2 SoC 整体框图

整个 SoC 的框图如下所示。

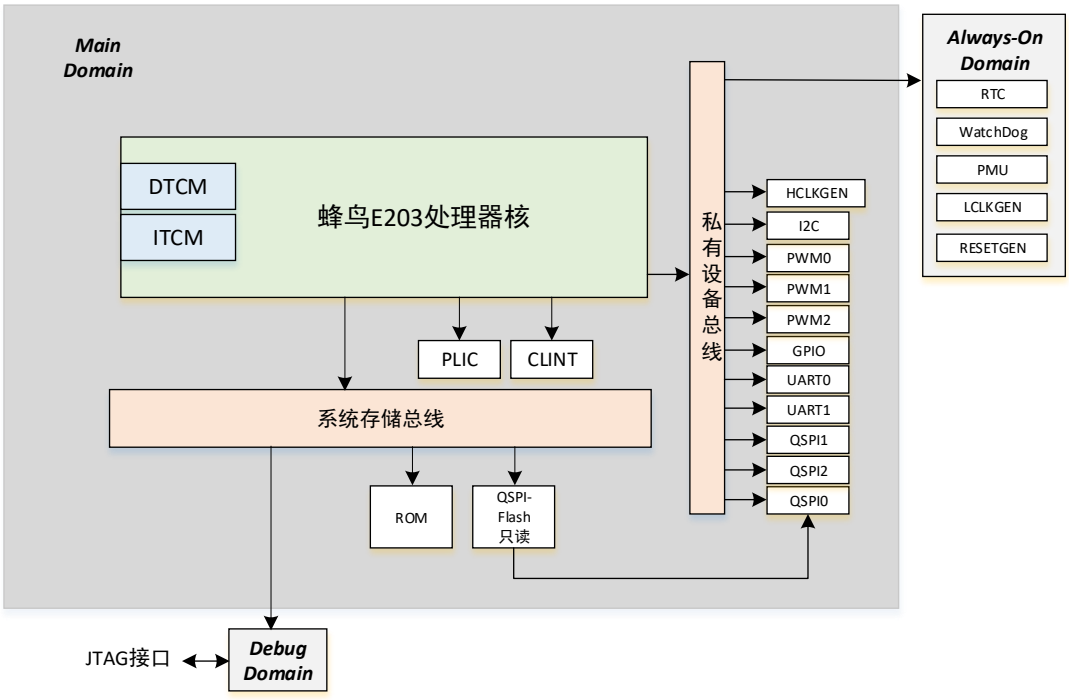

Figure 2-1 SoC 整体框图

上图中除了蜂鸟 E203 处理器核和总线之外，其他的主要外部设备 IP 模块均复用来自 Freedom E310 SoC 平台。图中用红色线框包含的模块为开源 Freedom E310 SoC 平台所不具备的（因为其开源的是 FPGA 平台源代码），需要我们自己设计添加，包括：

- I2C Master：使用开源的 I2C Master IP。
- HCLKGEN：用于使用 PLL 生成高速时钟（用于 Main Domain）。
- LCLKGEN：生成低速时钟（用于 Always-onDomain）。
- RESETGEN：用于为 SoC 生成复位逻辑。

2.3 总线地址分配

本 SoC 的总线地址分配如下表所示。注意，下表中：

- 普通部分为与 Freedom E310 兼容且已经存在的 IP 及其地址分配。
- 绿色高亮部分为本项目自研 RISC-V 核特有的部分。

- 黄色高亮部分为本项目新定义的系统 IP 及其地址分配区间。
- 蓝色高亮部分为 Freedom E310 芯片中存在的 IP 及其地址分配，但是目前本 SoC 中还没有的 IP。

Table 2-1 SoC 地址分配表

总线分组	组件	地址区间	描述
Core 直属	CLINT	0x0200_0000 ~ 0x0200_FFFF	Core Local Interrupt Controller 模块寄存器地址区间
	PLIC	0x0C00_0000 ~ 0x0CFF_FFFF	Platform Level Interrupt Controller 模块寄存器地址区间
	ITCM	0x8000_0000 ~ 0x8001_FFFF	ITCM 地址区间
	DTCM	0x9000_0000 ~ 0x8001_FFFF	DTCM 地址区间
系统存储总线接口	Debug Module	0x0000_0000 ~ 0x0000_0FFF	注意：Debug Module 主要用于调试器使用，普通软件程序不应该使用此区间
	ROM	0x0000_1000 ~ 0x0000_1FFF	注意：开源的 E203 项目由于是 FPGA 原型，因此 Mask-ROM 代码为一行为模型。 在本 SoC 中考虑将其固化成为逻辑，而不是真的使用 ROM IP。
	Off-Chip QSPI0 Flash Read	0x2000_0000 ~ 0x3FFF_FFFF	外部 SPI Flash 只读地址区间。 有关 SPI Flash，详见后文介绍。
私有设备总线接口（总区间为 0x1000_0000 ~ 0x1FFF_FFFF）	Always-On	0x1000_0000 ~ 0x1000_7FFF	Always-on 模块包含 PMU，RTC，WatchDog，LCLKGEN。
	HCLKGEN	0x1000_8000 ~ 0x1000_8FFF	高速时钟生成模块

	GPIO	0x1001_2000 ~ 0x1001_2FFF	GPIO 地址区间。 有关 GPIO，详见后文介绍。
	UART0	0x1001_3000 ~ 0x1001_3FFF	第一个 UART 模块地址区间。有关 UART，详见后文介绍。
	QSPI0	0x1001_4000 ~ 0x1001_4FFF	第一个 QSPI 模块地址区间。有关 SPI，详见后文介绍。
	PWM0	0x1001_5000 ~ 0x1001_5FFF	第一个 PWM 模块地址区间。有关 PWM，详见后文介绍。
	UART1	0x1002_3000 ~ 0x1002_3FFF	第二个 UART 模块地址区间。
	QSPI1	0x1002_4000 ~ 0x1002_4FFF	第二个 QSPI 模块地址区间。
	PWM1	0x1002_5000 ~ 0x1002_5FFF	第二个 PWM 模块地址区间。
	QSPI2	0x1003_4000 ~ 0x1003_4FFF	第三个 QSPI 模块地址区间。
	PWM2	0x1003_5000 ~ 0x1003_5FFF	第三个 PWM 模块地址区间。
	I2C Master	0x1004_2000 ~ 0x1004_2FFF	IC2 Master
其他地址区间	上表中未使用到的地址区间，则均为写忽略，读返回 0		

3 RISC-V 核介绍

注意：

- 本文档对蜂鸟 E203 处理器内核以及 RISC-V 指令集架构的介绍尚不够详细，在中文书籍《手把手教你设计 CPU：RISC-V 处理器篇》中对其进行深入浅出地系统的讲解。感兴趣的用户可以自行搜索此书。

3.1 处理器核简介

本 SoC 使用的处理器核为开源的蜂鸟 E203 RISC-V 处理器核。

蜂鸟 E203 内核的特性简介如下：

- E203 内核采用 2 级流水线结构，通过一流的处理器架构设计，该 CPU 核的功耗与面积均优于同级 ARM Cortex-M 核，实现业界最高的能效比与最低的成本。
- E203 内核能够运行 RISC-V 指令集，支持 RV32IMAC 等指令子集的配置组合，仅支持机器模式（Machine Mode Only）。
- E203 内核提供标准的 JTAG 调试接口，以及成熟的软件调试工具。
- E203 内核提供成熟的 GCC 编译工具链。

参见《手把手教你设计 CPU——RISC-V 处理器篇》了解开源蜂鸟 E203 的详情。

3.2 处理器核中断

蜂鸟 E203 内核的中断分为三种类型：

- 计时器中断，由 CLINT 模块产生。
- 软件中断，由 CLINT 模块产生。
- 外部中断，由 PLIC 管理产生。

对于 RISC-V 架构中断的详细介绍，本文档在此不展开赘述，用户参见中文书籍《手把手教你设计 CPU：RISC-V 处理器篇》第 13 章中对其进行深入浅出地系统的讲解。

3.2.1 CLINT

本文在此仅对 CLINT 进行中文简述。

有关 CLINT 的详细，用户可以选择中文书籍《手把手教你设计 CPU：RISC-V 处理器篇》第 13 章中对其进行深入浅出地系统的讲解。

CLINT 全称为 Core Local Interrupts Controller。CLINT 是一个存储器地址映射（Memory Address Mapped）的模块，挂载在处理器核为其实现的专用总线接口上，在蜂鸟 E203 内核配套的 SoC 中其寄存器的地址区间如下表所示。CLINT 的寄存器只支持 size 为 32 位的读写访问。

Table 3-1 CLINT 寄存器的 Memory Mapped Address

地址	寄存器名称	功能描述
0x0200_0000	msip	生成软件中断
0x0200_4000	mtimecmp	配置计时器的比较值
0x0200_BFF8	mtime	反映计时器的值

CLINT 可以用于生成软件中断，其要点如下：

- CLINT 中实现了一个 32 位的 msip 寄存器，该寄存器只有最低位为有效位，该寄存器有效位直接作为软件中断（Software Interrupt）信号通给处理器核。
- 当软件写 1 至 msip 寄存器触发了软件中断之后，前文中介绍的 CSR 寄存器 mip 中的 MSIP 域便会置高指示当前中断 pending 状态。
- 软件可通过写 0 至 msip 寄存器来清除该软件中断。

CLINT 可以用于生成计时器中断，其要点如下：

- CLINT 中实现了一个 64 位的 mtime 寄存器，该寄存器反映了 64 位计时器的值。计时器根据低速的输入节拍信号进行计时。
- CLINT 中实现了一个 64 位的 mtimecmp 寄存器，该寄存器作为计时器的比较值，假设计时器的值 mtime 大于或者等于 mtimecmp 的值时，则产生计时器中断。软件可以通过改写 mtimecmp 的值来清除计时器中断。

3.2.2 PLIC

本文在此仅对 PLIC 进行中文简述。

有关 PLIC 的详细，用户可以参见中文书籍《手把手教你设计 CPU：RISC-V 处理器篇》第 13 章中对其进行深入浅出地系统的讲解。

PLIC 全称为 Platform Level Interrupt Controller。PLIC 是 RISC-V 架构标准定义的系统中断控制器，主要用于多个外部中断源的优先级仲裁和派发。

如图 2-1 中所示，PLIC 是一个存储器地址映射（Memory Address Mapped）的模块，挂载在处理器核为其实现的专用总线接口上，在蜂鸟 E203 内核配套的 SoC 中其寄存器的地址区间如下表所示。PLIC 的寄存器只支持 size 为 32 位的读写访问。

Table 3-2 PLIC 寄存器的 Memory Mapped Address

地址	寄存器名称	功能描述
0x0C00_0004	Source 1 priority	中断源 1 的优先级
0x0C00_0008	Source 2 priority	中断源 2 的优先级
……	……	……
0x0C00_0FFC	Source 1023 priority	中断源 1023 的优先级
……	……	……
0x0C00_1000	Start of pending array（read-only）	中断等待标志的起始地址
……	……	……
0x0C00_107C	End of pending array	中断等待标志的结束地址
0x0C00_2000	Target 0 enables	Target 0 的使能位
……	……	……
0x0C20_0000	Target 0 priority threshold	Target 0 的优先级门槛
注意： ➢ 该 PLIC 理论上可以支持多大 1024 个中断源。 ➢ 该 PLIC 理论上可以支持多个中断目标（Target）。由于蜂鸟 E203 内核是一个单核处理器，且仅实现了 Machine Mode，因此仅用到 PLIC 的 Target0，表中的 Target0 即为蜂鸟 E203 内核。		

PLIC 理论上可以支持高达 1024 个外部中断源，在具体的 SoC 中连接的中断源个数可以不一样。蜂鸟 E203 内核的 SoC 系统是基于 Sifive 公司开源的 Freedom E310 SoC 开发所得。PLIC 在此 SoC 中连接了 GIPO，UART，PWM 等等多个外部中断源，其中断分配如下表所示。PLIC 将多个外部中断源仲裁为一个单比特的中断信号送入蜂鸟 E203 内核。

注意：由于本 SoC 增加了 I2C Master，因此增加了 I2C 的中断。

Table 3-3 PLIC 中断分配表

PLIC 源中断号	来源
0	预留为表示没有中断
1	wdogcmp
2	rtccmp

3 4	uart0 uart1
5 6 7	qspi0 qspi1 qspi2
8 …… 39	gpio0 …… gpio31
40 …… 43	pwm0cmp0 …… pwm0cmp3
44 …… 47	pwm1cmp0 …… pwm1cmp3
48 …… 51	pwm2cmp0 …… pwm2cmp3
52	i2c

3.3 JTAG 调试模块

蜂鸟 E203 内核采用标准（1149.1）JTAG 连接模块用于连接系统外部调试器（Debugger）与内部的调试模块（Debug Module）。

如图 2-1 中所示，Debug Module 调试模块，用于支持外部 JTAG 通过该模块调试处理器核，是的处理器核能够通过 GDB 对其进行交互式调试，譬如设置断点，单步执行等调试功能。

4 SoC 总线介绍

本 SoC 采用蜂鸟 E203 内核开发过程中定义了一种自定义总线协议 ICB（Internal Chip Bus），该总线用于蜂鸟 E203 内核内部使用，同时也可作为 SoC 中的总线使用。

ICB 总线的初衷是为了能够尽可能地结合 AXI 总线和 AHB 总线的优点，兼具高速性和易用性，它具有如下特性：

- 相比 AXI 和 AHB 而言，ICB 的协议控制更加简单，仅有两个独立的通道，如图 3-10 所示，读和写操作共用地址通道，共用结果返回通道。
- 与 AXI 总线一样采用分离的地址和数据阶段。
- 与 AXI 总线一样采用地址区间寻址，支持任意的主从数目，譬如一主一从，一主多从，多主一从，多主多从等拓扑结构。
- 与 AHB 总线一样每个读或者写操作都会在地址通道上产生地址，而非像 AXI 中只产生起始地址。
- 与 AXI 总线一样支持地址非对齐的数据访问，使用字节掩码（Write Mask）来控制部分写操作。
- 与 AXI 总线一样支持多个滞外交易（Multiple Oustanding Transaction）。
- 与 AHB 总线一样不支持乱序返回乱序完成。反馈通道必须按顺序返回结果。
- 与 AXI 总线一样非常容易添加流水线级数以获得高频的时序。
- 协议非常简单，易于桥接转换成其他总线类型，譬如 AXI，AHB，APB 或者 TileLink 等总线。

对于蜂鸟 E203 内核这样的低功耗处理器而言，ICB 总线能够被用于几乎所有的相关场合，包括：作为内部模块之间的接口，SRAM 模块接口，低速设备总线，系统存储总线等等。

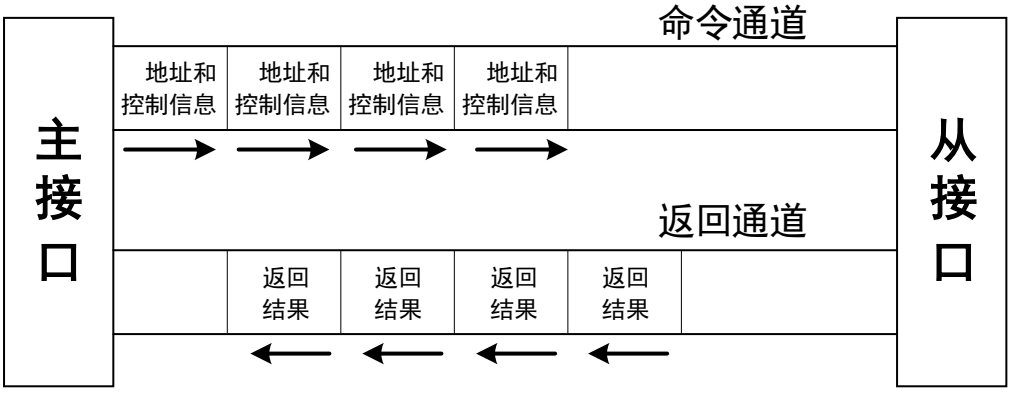

Figure 4-1 ICB 总线通道结构

4.1 ICB 总线协议信号

ICB 总线主要包含 2 个通道，如图 4-1 所示。ICB 总线信号列表如下表所示：

- 命令通道（Command Channel）
  Command Channel 主要用于主设备向从设备发起读写请求。
- 返回通道（Response Channel）
  Response Channel 主要用于从设备向主设备返回读写结果。

Table 4-1 ICB 总线信号

通道	方向	宽度	信号名	介绍
Command Channel	Output	1	icb_cmd_valid	主设备向从设备发送读写请求信号
	Input	1	icb_cmd_ready	从设备向主设备返回读写接受信号
	Output	32	icb_cmd_addr	读写地址
	Output	1	icb_cmd_read	读或是写操作的指示。读则以总线宽度（譬如 32 位）为单位读回一个数据。写则靠字节掩码（icb_cmd_wmask）控制写数据的大小（Size）。
	Output	32	icb_cmd_wdata	写操作的数据，数据的摆放格式与 AXI 协议一致
	Output	4	icb_cmd_wmask	写操作的字节掩码，掩码的摆放格式与 AXI 协议一致
Reponse Channel	Input	1	icb_rsp_valid	从设备向主设备发送读写反馈请求信号
	Output	1	icb_rsp_ready	主设备向从设备返回读写反馈接受信号
	Input	32	icb_rsp_rdata	读反馈的数据，数据的摆放格式与 AXI 协议一致
	Input	1	icb_rsp_err	读或者写反馈的错误标志

4.2 ICB 总线协议时序

本节将描述 ICB 总线的若干典型时序。

- 如下图所示：主设备向从设备通过 ICB 的 Command Channel 发送写操作请求（icb_cmd_read 为低），从设备立即接收该请求（icb_cmd_ready 为高）。从设备在同一个周期返回读结果且结果正确（icb_rsp_err 为低），主设备立即接收该结果（icb_rsp_ready 为高）。

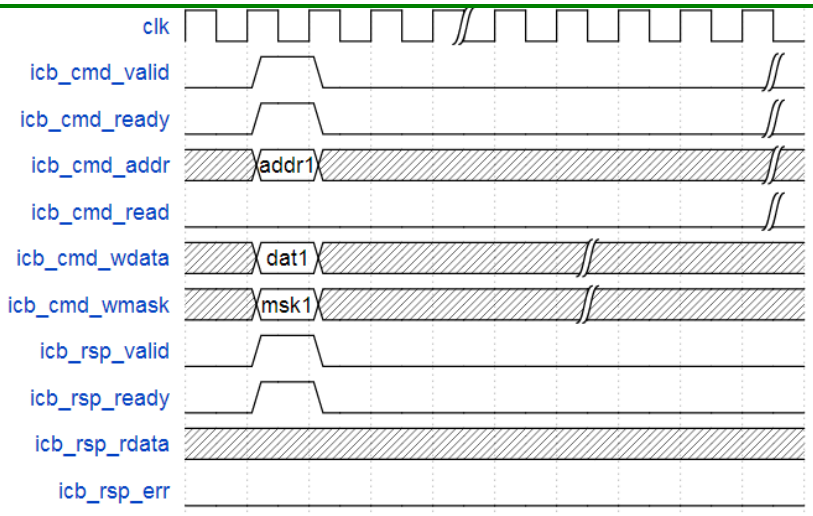

Figure 4-2 写操作同一周期返回结果

- 如下图所示：主设备向从设备通过 ICB 的 Command Channel 发送读操作请求（icb_cmd_read 为高），从设备立即接收该请求（icb_cmd_ready 为高）。从设备在下一个周期返回读结果且结果正确（icb_rsp_err 为低），主设备立即接收该结果（icb_rsp_ready 为高）。

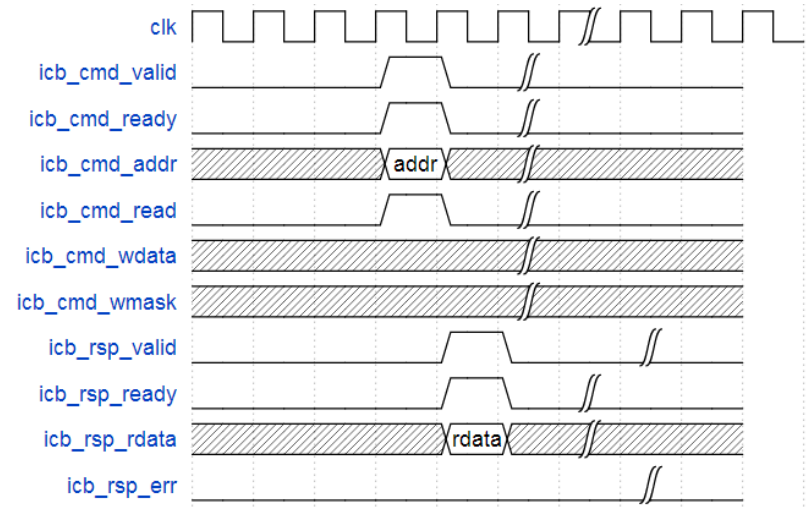

Figure 4-3 读操作下一周期返回结果

- 如下图所示：主设备向从设备通过 ICB 的 Command Channel 发送写操作请求（icb_cmd_read 为低），从设备立即接收该请求（icb_cmd_ready 为高）。从设备在下一个周期返回读结果且结果正确（icb_rsp_err 为低），主设备立即接收该结果（icb_rsp_ready 为高）。

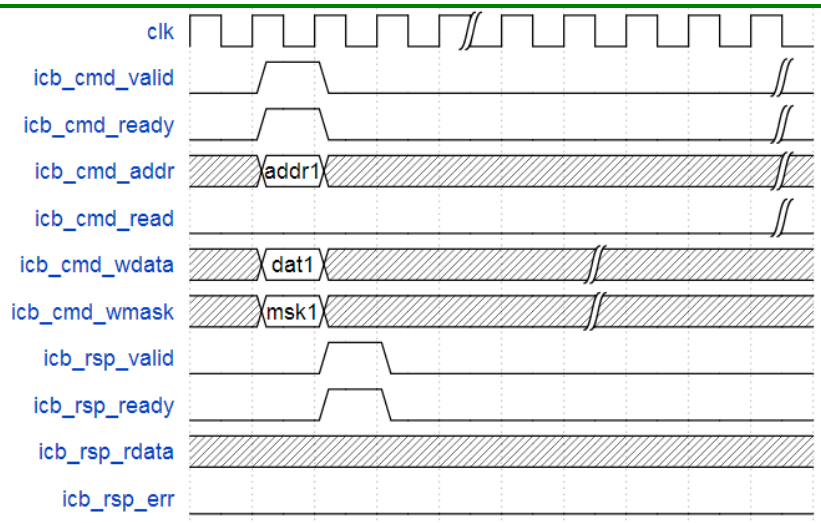

Figure 4-4 写操作下一周期返回结果

- 如下图所示：主设备向从设备通过 ICB 的 Command Channel 发送读操作请求（icb_cmd_read 为高），从设备立即接收该请求（icb_cmd_ready 为高）。从设备在四个周期后返回读结果且结果正确（icb_rsp_err 为低），主设备立即接收该结果（icb_rsp_ready 为高）。

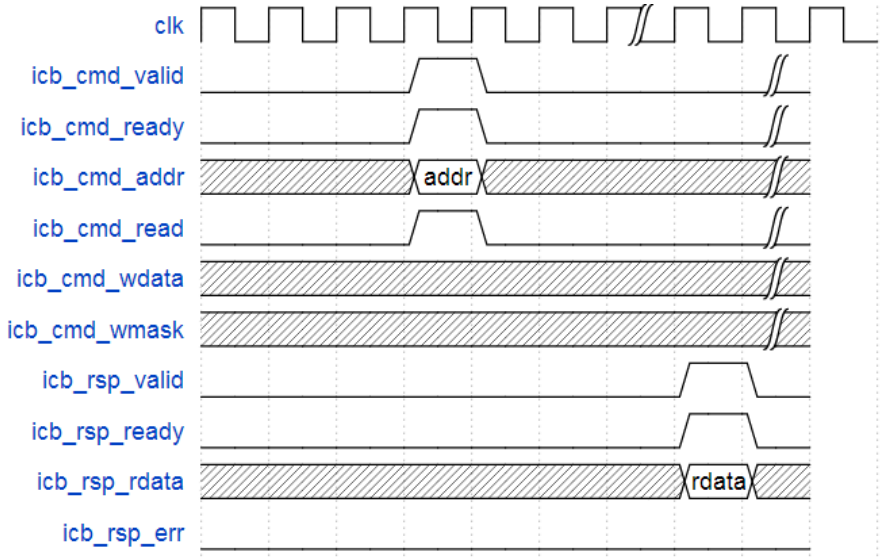

Figure 4-5 读操作四个周期返回结果

- 如下图所示：主设备向从设备通过 ICB 的 Command Channel 发送写操作请求（icb_cmd_read 为低），从设备立即接收该请求（icb_cmd_ready 为高）。从设备在四个周期后返回结果且结果正确（icb_rsp_err 为低），主设备立即接收该结果（icb_rsp_ready 为高）。

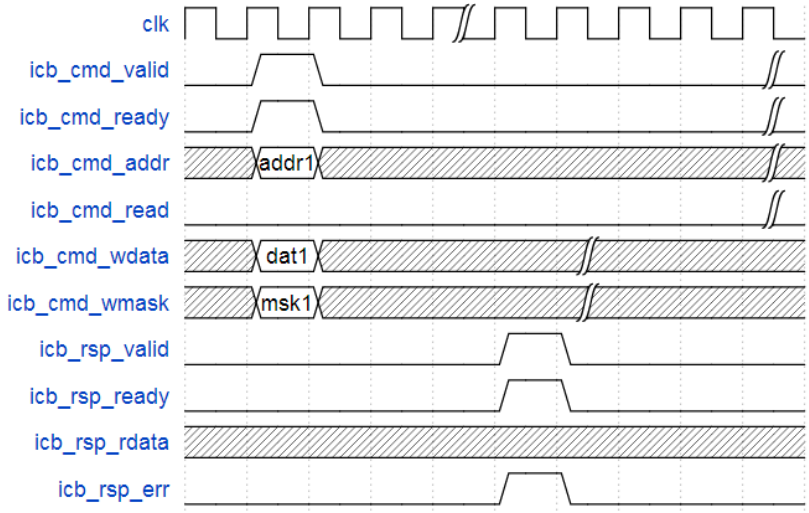

Figure 4-6 写操作四个周期返回结果

- 如下图所示：主设备向从设备通过 ICB 的 Command Channel 连续发送四个读操作请求（icb_cmd_read 为高），从设备均立即接收该请求（icb_cmd_ready 为高）。从设备在四个周期后连续返回四个读结果，其中前三个结果正确（icb_rsp_err 为低），第四个结果错误（icb_rsp_err 为高），主设备均立即接收此四个结果（icb_rsp_ready 为高）。

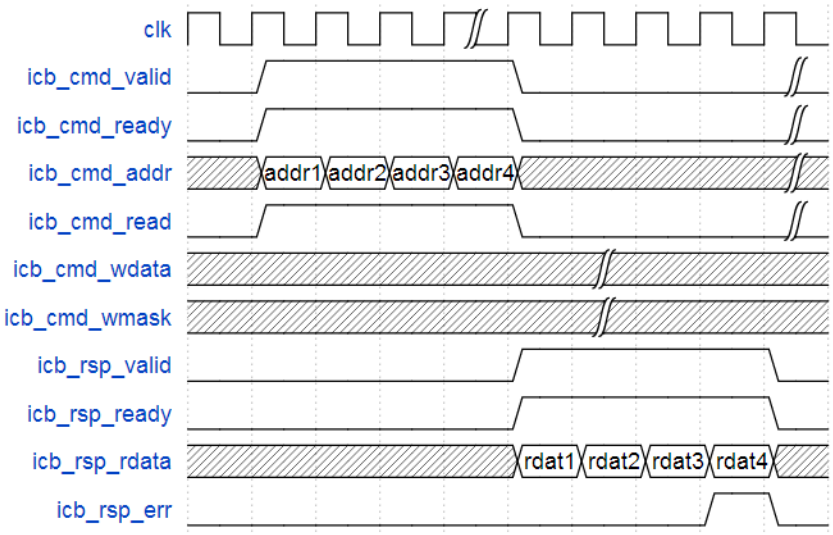

Figure 4-7 连续四个读操作均四个周期返回结果

- 如下图所示：主设备向从设备通过 ICB 的 Command Channel 连续发送四个写操作请求（icb_cmd_read 为低），从设备均立即接收该请求（icb_cmd_ready 为高）。从设备在四个周期后连续返回四个写结果，其中前三个结果正确（icb_rsp_err 为低），第四个结果错误（icb_rsp_err 为高），主设备均立即接收此四个结果（icb_rsp_ready 为高）。

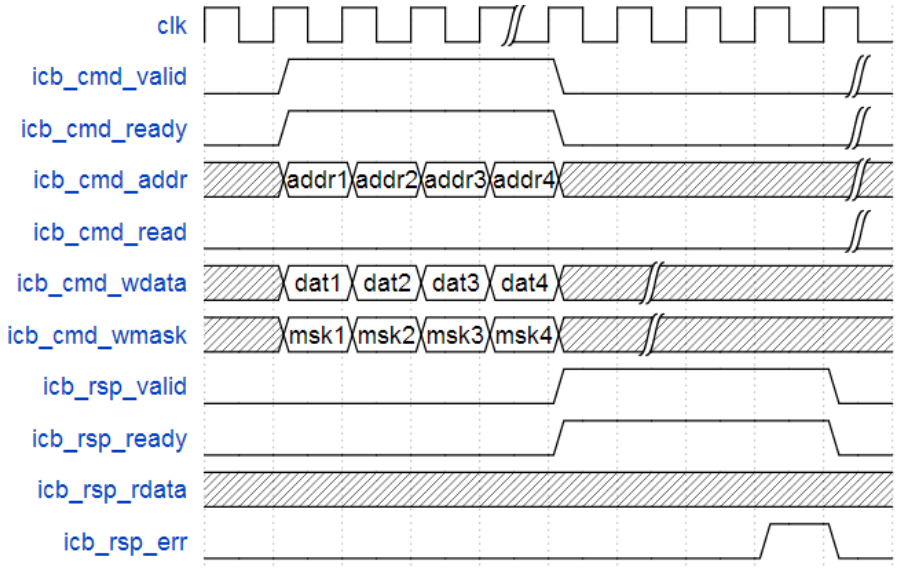

Figure 4-8 连续四个写操作均四个周期返回结果

- 如下图所示：主设备向从设备通过 ICB 的 Command Channel 相继连续发送三个读和写操作请求。从设备立即接收了第一个和第三个请求，但是第二个请求第一个周期并没有立即接受（icb_cmd_ready 为低），因此主设备一直将地址控制和写数据信号保持不变，直到下一周期该请求被从设备接受（icb_cmd_ready 为高）。从设备对于第一个和第二个请求都是在同一个周期就返回结果且被主设备立即接受，但是对于第三个请求则是在下一个周期才返回结果，并且主设备还没有立即接受（icb_rsp_ready 为低），因此从设备一直将返回信号保持不变，知道下一周期该返回结果被主设备接受。

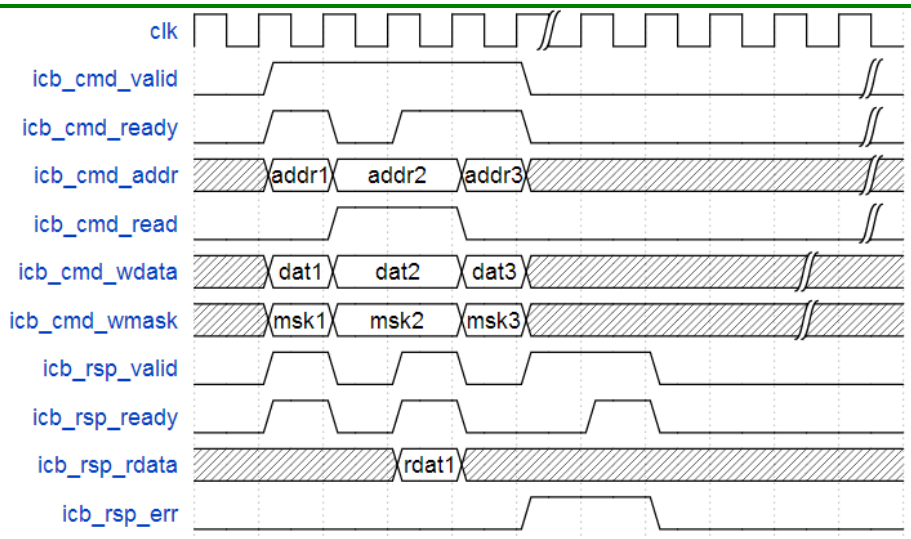

Figure 4-9 读写混合发生

4.3 SoC 总线结构

本 SoC 总线结构如下图所示：

- 蜂鸟 E203 内核 BIU 的系统存储接口 ICB 连接系统存储总线，通过其访问 SoC 中的若干存储组件，譬如 ROM，Flash 的只读区间等。
- 蜂鸟 E203 内核 BIU 的私有设备接口 ICB 连接私有设备总线，通过其访问 SoC 中的若干设备，譬如 UART，GPIO 等等。

有关 SoC 的结构和组件以及地址分配，请参见第 2.3 节。

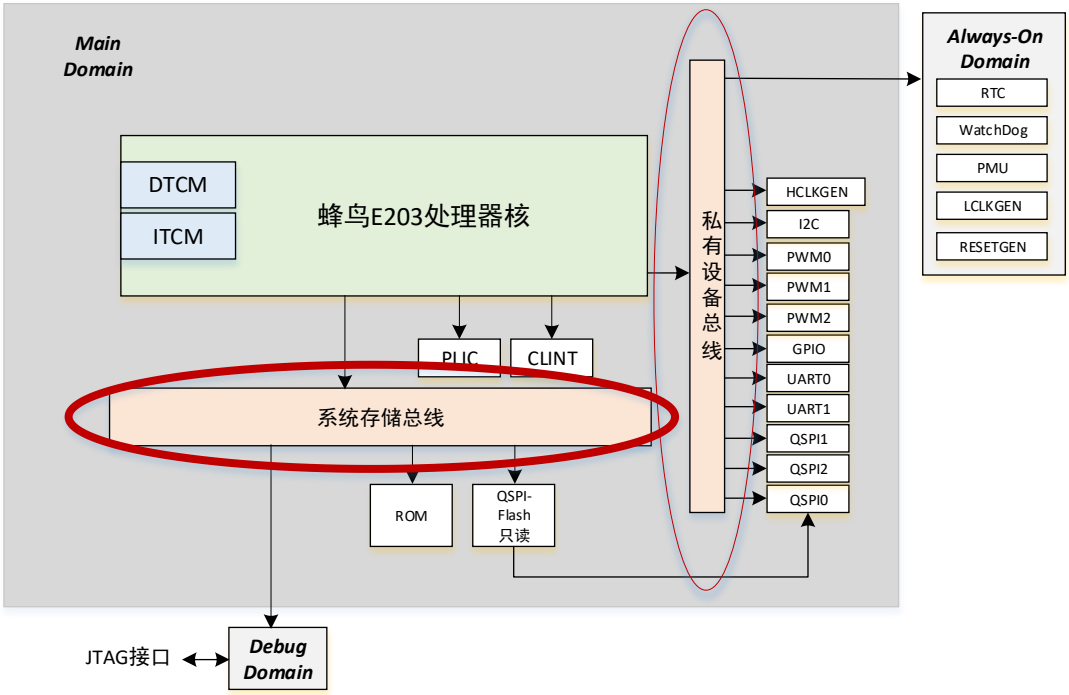

Figure 4-10 SoC 总线示意图

5 SoC 外设介绍

本章将对挂载在私有设备总线上的外设模块进行介绍。本文在此仅对每个模块功能进行极为简略的描述。

注意：

- 本文档对 SoC 的各外设的介绍尚不够详细，由于本 SoC 中外设模块重用自 Freedom E310 SoC 且软件完全兼容，可以参阅 Freedom E310 SoC 的技术文档了解其细节。分别为 SiFive-E300-platform-reference-manual-v1.0.1.pdf 与 SiFive-E310-G000-manual-v1.0.1.pdf，其中有对每个模块的详细功能描述与详细配置寄存器描述供参阅。
- 或者，在中文书籍《RISC-V 架构与嵌入式开发快速入门》中对英文 SoC 的材料进行了通俗化翻译，并进行深入浅出的系统讲解。感兴趣的用户可以自行搜索此书。

5.1 QSPI Master

有关 QSPI 的详细介绍请参见文档《SiFive-E300-platform-reference-manual-v1.0.1.pdf》或者中文书籍《RISC-V 架构与嵌入式开发快速入门》。本文在此进行中文简述。

在本 SoC 中，有三个 QSPI Master，分别是 QSPI0，QSPI1 和 QSPI2，均是直接复用来自 Freedom E310 SoC 平台。其中 QSPI0 专用于外部的 Flash，有专用的 Pad 接口，而 QSPI1 和 QSPI2 则通过 GPIO 复用管脚。

- Quad-SPI Flash：
  - 专用于连接外部 Flash 的 Quad-SPI（QSPI）接口。
  - 并且该 QSPI 接口还可以被软件配置成为 eXecute-In-Place 模式，在此模式下，Flash 可以被当作一段只读区间直接被当做存储器读取。在默认上电之后，QSPI 即处于该模式之下，由于 Flash 掉电不丢失的特性，因此可以将系统的启动程序存放于外部的 Flash 中，然后处理器核通过 eXecute-In-Place 模式的 QSPI 接口直接访问外部 Flash 加载启动程序启动。
- QSPI：
  - 除了上述专用于 Flash 的 QSPI 接口之外，SoC 还有两个独立的 QSPI 接口控制器。一个 QSPI 使用四个片选信号（Chip Selects），一个 QSPI 使用一个片选信号。两个 QSPI 均使用 GPIO 的 IOF 功能与外界通信。

5.2 GPIO

有关 GPIO 的详细介绍请参见文档《SiFive-E300-platform-reference-manual-v1.0.1.pdf》或者中文书籍《RISC-V 架构与嵌入式开发快速入门》。本文在此进行中文简述。

GPIO 全称为 General Purpose I/O，其要点如下：

- GPIO 用于提供一组 32 I/O 的通用输入输出接口。每个 I/O 可用被软件配置为输入或者输出，如果是输出可以设置具体的输出值。
- 每个 I/O 还可以被配置为 IOF（Hardware I/O Functions），也就是将 I/O 供 SoC 内部的其他模块复用，譬如 SPI，UART，PWM 等等。
- 另外，每个 GPIO 的 I/O 均作为一个中断源连接到 PLIC 的中断源上。
- GPIO 的 32 个 I/O 被 SoC 内部模块的复用分配如下表所示，其中每个 I/O 均可以供两个内部模块复用，软件可以通过配置每个 I/O 使其选择 IOF0 或者 IOF1 来选择信号来源。注意：表中黄色高亮部分为本 SoC 新添加的 IO 映射，其他 IO 均与 Freedom E310 I/O 映射相同。

Table 5-1 GPIO 的接口分配表

GPIO Pin 编号	IOF0	IOF1
0		PWM0_0
1		PWM0_1
2	QSPI1:SS0	PWM0_2
3	QSPI1:SD0/MOSI	PWM0_3
4	QSPI1:SD1/MISO	
5	QSPI1:SCK	
6	QSPI1:SD2	
7	QSPI1:SD3	
8	QSPI1:SS1	
9	QSPI1:SS2	
10	QSPI1:SS3	PWM2_0
11		PWM2_1
12	I2C:SDA	PWM2_2
13	I2C:SCL	PWM2_3
14		
15		
16	UART0:RX	
17	UART0:TX	
18		
19		PWM1_1
20		PWM1_0
21		PWM1_2
22		PWM1_3
23		
24	UART1:RX	
25	UART1:TX	
26	QSPI2:SS	
27	QSPI2:SD0/MOSI	

28	QSPI2:SD1/MISO	
29	QSPI2:SCK	
30	QSPI2:SD2	
31	QSPI2:SD3	

5.3 UART

有关 UART 的详细介绍请参见文档《SiFive-E300-platform-reference-manual-v1.0.1.pdf》或者中文书籍《RISC-V 架构与嵌入式开发快速入门》。本文在此进行中文简述。

UART 全称为 Universal Asynchronous Receiver-Transmitter（通用异步接收-发射器），本 SoC 有两个独立的 UART，两个 UART 均使用 GPIO 的 IOF 功能与外界通信。

5.4 PWM

有关 PWM 的详细介绍请参见文档《SiFive-E300-platform-reference-manual-v1.0.1.pdf》或者中文书籍《RISC-V 架构与嵌入式开发快速入门》。本文在此进行中文简述。

PWM 全称为 Pulse-Width Modulator（脉宽调节器）。本 SoC 有三个独立的 PWM，其中两个是 16 比特的精度，另外一个是 8 比特的精度，均使用 GPIO 的 IOF 功能与外界通信。

5.5 Always-On 模块

有关 Always-On 模块的详细介绍请参见文档《SiFive-E300-platform-reference-manual-v1.0.1.pdf》或者中文书籍《RISC-V 架构与嵌入式开发快速入门》。本文在此进行中文简述。

Always-ON 模块包含了三个主要子模块，分别是 WatchDog，RTC 和 PMU。

5.5.1 WatchDog

有关 WatchDog 模块的详细介绍请参见文档《SiFive-E300-platform-reference-manual-v1.0.1.pdf》或者中文书籍《RISC-V 架构与嵌入式开发快速入门》。本文在此进行中文简述。

WatchDog 全称为 Watch Dog Timer（看门狗计数器），该计数器位于 Always-on Domain 中，因此使用低速时钟进行计数，并且可以通过配置其计数的目标值产生中断。

5.5.2 RTC

有关 RTC 模块的详细介绍请参见文档《SiFive-E300-platform-reference-manual-v1.0.1.pdf》或者中文书籍《RISC-V 架构与嵌入式开发快速入门》。本文在此进行中文简述。

RTC 全称为 Real-Time Counter（实时计数器），该计数器位于 Always-on Domain 中，因此使用低速时钟进行计数，并且还能产生中断。

5.5.3 PMU

有关 PMU 模块的详细介绍请参见文档《SiFive-E300-platform-reference-manual-v1.0.1.pdf》或者中文书籍《RISC-V 架构与嵌入式开发快速入门》。本文在此进行中文简述。

PMU 全称为 Power Management Unit（电源管理单元），用于控制 SoC 的电源管理。整个 SoC 除了 WatchDog、RTC、PMU 等模块处于 Always-on Domain 之外，其他 Main Domain 可以在 PMU 的控制下被置于断电状态以节省功耗，或者重新唤醒等等。

5.5.4 LCLKGEN

LCLKGEN 全称为 Low-Speed Clock Generation。LCLKGEN 主要为 Always-On Domain 生成时钟。Always-On Domain 主要使用低速的实时时钟，频率应为 32.768KHz，可以选择来自片上振荡器、外部晶振或者直接通过芯片引脚输入。

LCLKGEN 模块的结构取决于具体芯片的工艺和 IP，因此本文在此不做介绍。

注意：在 FPGA 开发板中，LCLKGEN 模块为空模块，直接输出由 FPGA 产生的 32.768KHz 时钟。

5.5.5 RESETGEN

RESETGEN 模块主要为整个 SoC 模块产生复位信号，请参见第 9 章了解 SoC 复位管理。

5.6 I2C Master

I2C Master 模块是采用开源的并经过测试验证后的 IP core；内部采用 wishbone 总线通信。寄存器列表如下，关于寄存器的详细介绍请参考《I2CMasterwithWISHBONEBusInterface-Documentation.pdf》或者中文书籍《RISC-V 架构与嵌入式开发快速入门》。

Name	Address	Width	Access	Description
PRERlo	0x00	8	RW	Clock Prescale register lo-byte
PRERhi	0x01	8	RW	Clock Prescale register hi-byte
CTR	0x02	8	RW	Control register
TXR	0x03	8	W	Transmit register
RXR	0x03	8	R	Receive register
CR	0x04	8	W	Command register
SR	0x04	8	R	Status register

注意，上表中地址为偏移地址，总线分配的基地址为 0x1004_2000。

5.7 HCLKGEN

HCLKGEN 全称为 High-Speed Clock Generation。HCLKGEN 其主要为 Main Domain 生成高速时钟（譬如频率为 100MHz）。HCLKGEN 可以使用片上振荡器、外部晶振和片上 PLL 产生高速时钟，PLL 也可以通过软件配置将其旁路。HCLKGEN 模块的结构取决于具体芯片的工艺和 IP，因此本文在此不做介绍。

注意：在 FPGA 开发板中，HCLKGEN 模块为空模块，直接输出由 FPGA 产生的 16MHz 时钟。

HCLKGEN 模块有若干可编程寄存器（地址区间为 0x1000_8000~0x1000_8FFF），其用于控制 HCLKGEN 相关功能。由于 HCLKGEN 模块的结构取决于具体芯片的工艺和 IP，因此本文在此对相关寄存器不做介绍。

6 SoC 片上存储器介绍

6.1 ITCM

ITCM 为 RISC-V Core 私有的指令存储器，其特性如下：

- 大小为 64KB。
- ITCM 数据宽度为 64 位。
- ITCM SRAM 虽然主要用于存放指令，但是其地址区间也可以被 Load、Store 指令访问，从而用来存放数据。

6.2 DTCM

DTCM 为 RISC-V Core 私有的数据存储器，其特性如下：

- 大小为 64KB。
- DTCM SRAM 数据宽度为 32 位。

6.3 ROM

在本 SoC 中使用 Verilog 常数逻辑化的 ROM，本质上就是常数逻辑。

7 SoC 电源域管理

7.1 电源域划分

为了保证芯片能够进入低功耗模式，将整个芯片划分为两个主要的电源域：

- Always-On Domain：
  - 此 Domain 即第 5 章中介绍的 Always-on 模块，包括子模块 WatchDog，RTC，和 PMU，还有 Always-On 时钟生成电路。
- MOFF Domain：
  - MOFF 是 Most-Off 的简称，即芯片中除了 Always-On Domain 之外的所有其他主体部分。

注意：在 FPGA 平台上没有真正的电源域。

7.2 低功耗模式

整个芯片分为三个工作模式：

- 正常模式：Always-On 和 MOFF 均处于正常供电状态。
- 等待模式：Always-On 和 MOFF 均处于正常供电状态。但是 Core 执行了一条 WFI 指令，因此处理器停止执行，时钟被关闭。直到下次被中断唤醒。
- 休眠模式：Always-On 正常供电，但是 MOFF 的外部电源切断。
  - 进入休眠模式由 Always-On 的 PMU 模块控制，PMU 模块会通过芯片的 Pad 输出一根控制信号 AON_PMU_VDDPADEN，该信号会控制芯片外部的 MOFF 供电电路使其切断电源。注意：在 FPGA 平台上没有真正的电源域和切断电源。
  - 退出休眠模式由 PMU 定义的若干唤醒条件唤醒，具体的唤醒条件请参见有关 PMU 模块的详细介绍请参见文档《SiFive-E300-platform-reference-manual-v1.0.1.pdf》或者中文书籍《RISC-V 架构与嵌入式开发快速入门》。

8 SoC 时钟管理

8.1 时钟域划分

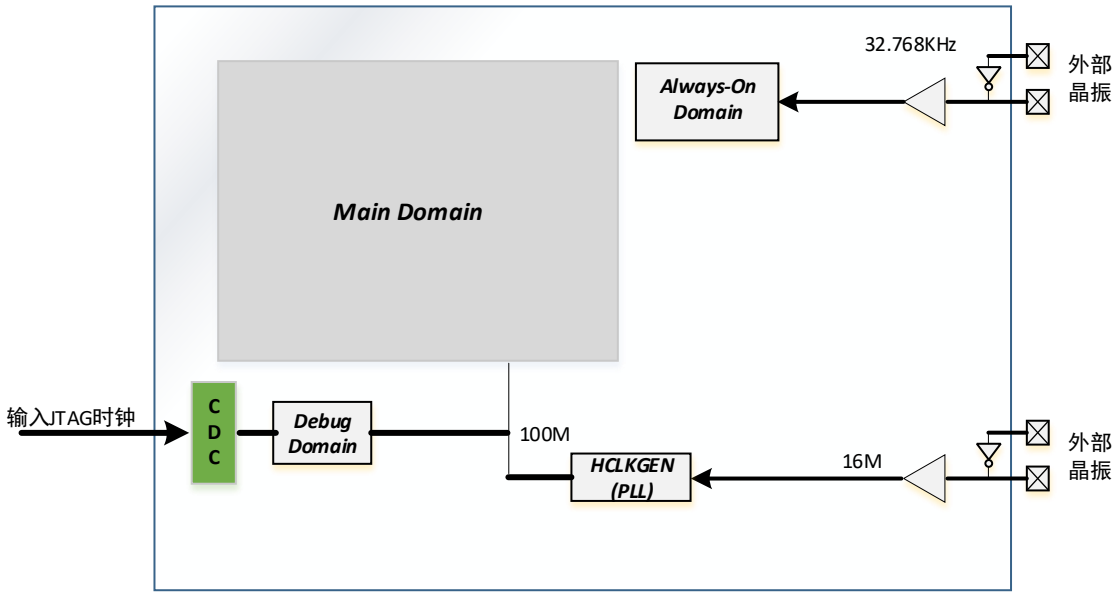

Figure 8-1 SoC 时钟域划分

如上图所示，将整个芯片划分为三个主要的时钟域：

- Always-On Domain：
  - 此 Domain 主要使用低速的 Always-on 时钟，频率为 32.768KHz。
  - 时钟可以选择来自外部的晶振或者直接通过芯片引脚输入。
- Main Domain：
  - 此 Domain 包含了芯片的主体，在此 Domain 中没有再划分时钟域，因此处理器核和总线以及外设 IP 均使用同样的时钟。
  - 此 Domain 自带 HCLKGEN 时钟生成模块，使用片上 PLL 产生高速时钟。PLL 也可以通过软件配置旁路其时钟，请参见第 5 章对于 HCLKGEN 模块的详细介绍。
- Debug Domain：
  - 此 Domain 包含了为了支持 JTAG 对 RISC-V 调试功能而添加的相关逻辑。
  - 此模块由两个不同的时钟域组成，分别是 JTAG 时钟和 RISC-V Core 时钟（即 Main Domain 的时钟），因此其内部有时钟跨越异步处理。

9 SoC 复位管理

9.1 芯片复位策略

首先，Always-On Domain 的复位可以来源于三个来源：

- 来自于 POR（Power-On-Reset）电路；
- 来自于芯片引脚 AON_ERST_N；
- 来自于 WatchDog 生成的 Reset；

9.1.1 POR 电路 Reset

Power-On Reset Circuit

This optional circuit holds it's output low until the voltage in the AON block rises above a design-time configurable preset threshold.

9.1.2 WatchDog Reset

有关 Watchdog 模块的详细介绍请参见文档《SiFive-E300-platform-reference-manual-v1.0.1.pdf》或者中文书籍《RISC-V 架构与嵌入式开发快速入门》。

9.1.3 芯片引脚 AON_ERST_N

此部分与 Freedom E310 完全相同：

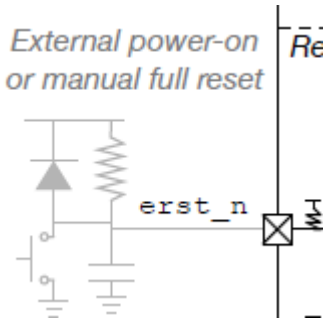

Figure 9-1 外部复位电路

The E300 can be reset by pulling down on the external reset pin (erst_n), which has a weak pullup. An external power-on reset circuit consisting of a resistor and capacitor can be to provided to generate a sufficiently long pulse to allow supply voltage to rise and then initiate the reset stretcher.

The external reset circuit can add a diode as shown to quickly discharge the capacitor after the supply is removed to rearm the external power-on reset circuit.

A manual reset button can be connected in parallel over the capacitor.

9.1.4 复位树关系

上述描述的三种复位来源，任一种驱动都将触发系统进行复位，具体的 reset 原因都会反应在 PMU 的 pmucause 寄存器中，供复位后软件读取查询。

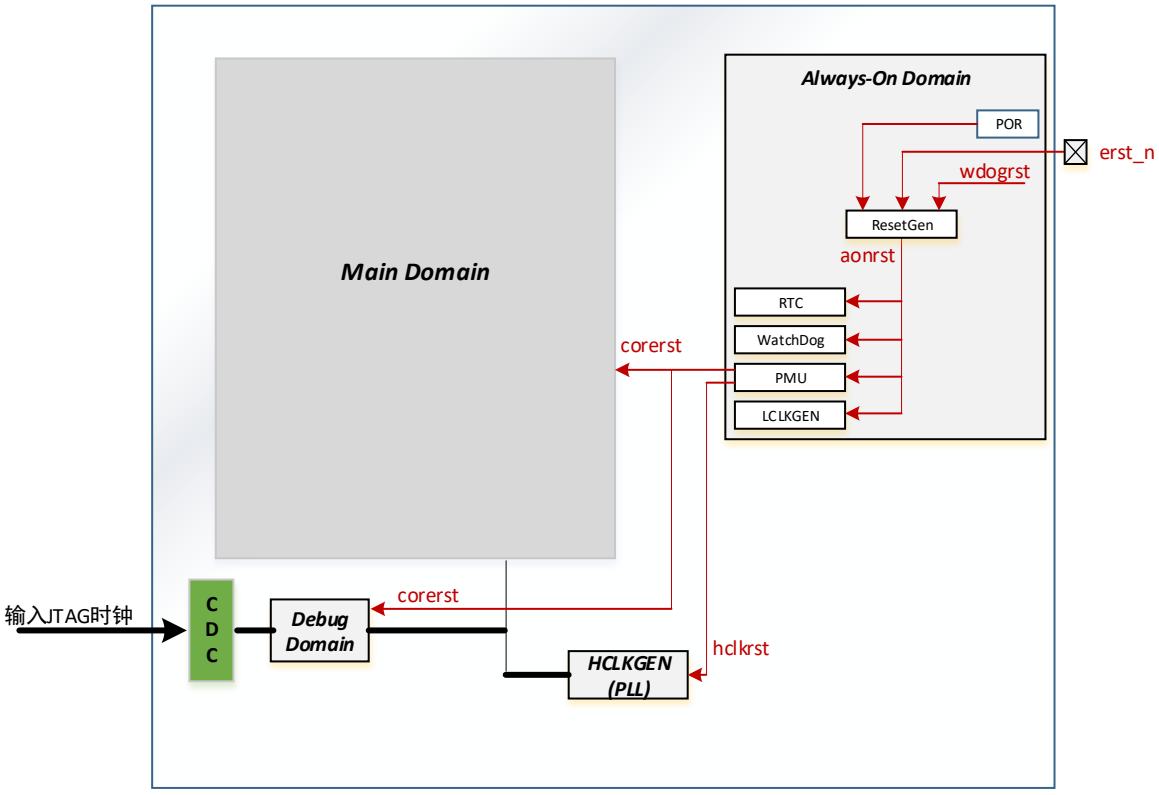

Figure 9-2 SoC 复位结构图

经三个来源最终生成的 reset 经过 Always-On Clock 同步成为异步置位、同步释放的 aonrst 信号，由 aonrst 信号进一步驱动 PMU 复位芯片的其他部分，其复位树的关系如上图所示，要点如下：

- aonrst 信号将作为 Always-On 模块本身的主复位信号，复位 Always-On 模块的 PMU，WatchDog，RTC 等。

- PMU 被 aonrst 复位之后，将进入默认的 wakeup sequence，生成 hclkrst 和 corerst 信号：
  - hclkrst 信号用于驱动 HCLKGEN 模块（主要包含 PLL 时钟生成）。
  - corerst 将用于除了 HCLKGEN 模块之外的所有 Main Domain 中的复位。

注意：每个复位信号都应该在相应的时钟域内进行同步，使之成为异步置位同步释放的复位信号。

10 上电流程控制

10.1 上电流程

本 SoC 中处理器核上电复位之后，可以从两个不同的地址进行执行程序，分别为：

- 从外部 Flash 开始执行。
- 从内部 ROM 开始执行。

下文分别予以论述。

10.1.1 从外部 Flash 开始执行

如 2.3 节中所述，由于在本 SoC 中外部 Flash 的地址区间位于 0x2000_0000 ~ 0x3FFF_FFFF，因此，如果从外部 Flash 开始执行，则 RISC-V 处理器核的 PC 复位值为 0x20000000。

如果从 Flash 中启动，软件应该设置上电启动引导程序，将 Flash 中的代码和数据搬运到片上 ITCM 和 DTCM 中，然后跳转到 ITCM 中开始执行。

10.1.2 从内部 ROM 开始执行

如 2.3 节中所述，由于在本 SoC 中内部 ROM 的地址区间位于 0x0000_1000 ~ 0x0000_1FFF，因此，如果从内部 ROM 开始执行，则 RISC-V 处理器核的 PC 复位值为 0x0000_1000。

在 ROM 中存放的代码为固定代码，其直接跳转至 ITCM（地址为 0x8000_0000）中继续执行。

从内部 ROM（继而从 ITCM）开始执行的方式，只有在调试阶段（调试器初始化了 ITCM 之后）才有意义，否则由于 ITCM 中没有初始化程序，将无法正常执行。

10.2 上电地址选择

本 SoC 可以通过顶层引脚的值来选择上电地址，如下表所示。

Table 10-1 SoC 上电控制

芯片引脚	值	地址区间
BOOTROM_N	0	从内部 ROM 上电执行
	1	从外部 QSPI 上电执行

11 SoC 顶层引脚

11.1 SoC 顶层引脚分配

本 SoC 的顶层引脚分布如下表所示。

Table 11-1 SoC 顶层引脚分配表

类型	方向	名称	描述
MOFFClock	Input	hfextclk	High speed clock input
Always-On Clock	Input	lfextclk	32.768KHz clock input
Always-On I/O Connections	Output	PMU_VDDPADEN	Programmable SLEEP control.
	Output	PMU_PADRST	Programmable SLEEP control.
	Input	PMU_DWAKEUP_N	Digital Wake-from-sleep. Activelow.
	Input	AON_ERST_N	External System Reset. Activelow.
上电地址选择	Input	BOOTROM_N	The boot up PC address selector. ➢ When driven low,RISC-V Core boot up from internal ROM (0x00001000). ➢ When left unconnected ordriven high, RISC-V Core boot up from QSPI flash (0x20000000).
JTAG 调试	Input	JTAG TCK	JTAG Clock line for debug interface
	Output	JTAG TDO	JTAG Data Out for debug interface
	Input	JTAG TMS	JTAG Test Mode Select fordebug interface
	Input	JTAG TDI	JTAG Data In for debug interface
QSPI0 外部 Flash	Bidir	QSPI DQ 3	Quad SPI Data Line
	Bidir	QSPI DQ 2	Quad SPI Data Line
	Bidir	QSPI DQ 1	Quad SPI Data Line
	Bidir	QSPI DQ 0	Quad SPI Data Line
	Output	QSPI CS	Quad SPI Chip Select. ActiveLow.
	Output	QSPI SCK	Quad SPI Clock Signal.
GPIO	Bidir	GPIO_0	32 根 GPIO 管脚
	Bidir	GPIO_1	
	Bidir	GPIO_2	
	Bidir	GPIO_3	
	Bidir	GPIO_4	
	Bidir	GPIO_5	
	Bidir	GPIO_6	

	Bidir	GPIO_7	
	Bidir	GPIO_8	
	Bidir	GPIO_9	
	Bidir	GPIO_10	
	Bidir	GPIO_11	
	Bidir	GPIO_12	
	Bidir	GPIO_13	
	Bidir	GPIO_14	
	Bidir	GPIO_15	
	Bidir	GPIO_16	
	Bidir	GPIO_17	
	Bidir	GPIO_18	
	Bidir	GPIO_19	
	Bidir	GPIO_20	
	Bidir	GPIO_21	
	Bidir	GPIO_22	
	Bidir	GPIO_23	
	Bidir	GPIO_24	
	Bidir	GPIO_25	
	Bidir	GPIO_26	
	Bidir	GPIO_27	
	Bidir	GPIO_28	
	Bidir	GPIO_29	
	Bidir	GPIO_30	
	Bidir	GPIO_31	