ICARUS 流片工程经验总结与改进

1. 接口

ICARUS 芯片通过 16 位的并行接口(Bus)与 1 位串行接口(SPI)与 FPGA 进行数据交换，完成 FPGA 向 ICARUS 芯片写入数据、读出结果、配置寄存器等行为。本章节总结了上述两类接口的原理(时序)、芯片测试现象以及需要改进的问题。

1.1 并行 Bus 接口

A. 原理与时序

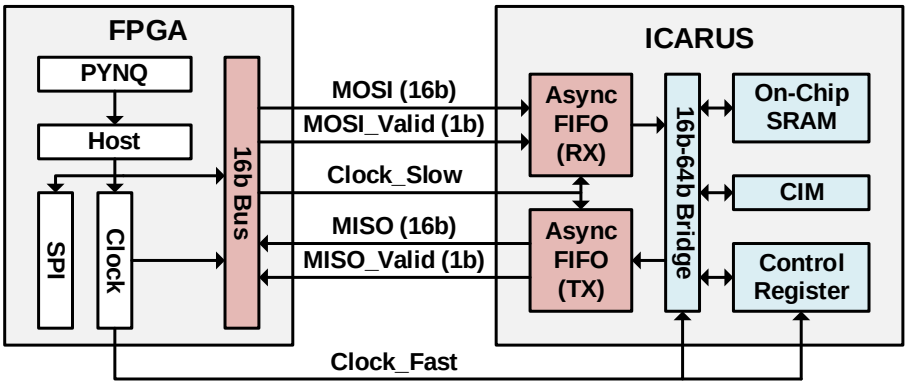

图 1 并行 Bus 接口框图

图 1 给出了 FPGA 与 ICARUS 芯片通过 16 位 Bus 接口进行数据通信的示意框图。其中 FPGA 一侧主要了例化了定制的 Host 模块和 Clock 模块：Host 模块用于控制并行 Bus 的行为，与 ICARUS 芯片进行数据通信；Clock 模块用于生成所需的时钟，慢时钟用于低速的数据通信，快时钟为 ICARUS 片上功能电路的工作时钟。FPGA 上的各模块均通过 PYNQ Overlay 完成配置。

并行 Bus 接口所需的引脚位宽、方向与功能汇总如下，并使用 FMC 接口在物理上连接 FPGA 与 ICARUS 芯片：

引脚名称	位宽	方向	功能
MOSI	16b	FPGA 至 ICARUS	FPGA 向 ICARUS 传输 16b 数据；
MOSI_Valid	1b	FPGA 至 ICARUS	FPGA 向 ICARUS 传输数据的有效标志，高电平代表数据有效；
MISO	16b	ICARUS 至 FPGA	ICARUS 向 FPGA 传输 16b 数据；
MISO_Valid	1b	ICARUS 至 FPGA	ICARUS 向 FPGA 传输数据的有效标志，高电平代表数据有效；
Clock_Slow	1b	FPGA 至 ICARUS	上述 4 组信号的参考慢时钟；

Clock_Fast	1b	FPGA 至 ICARUS	ICARUS 芯片的工作快时钟；

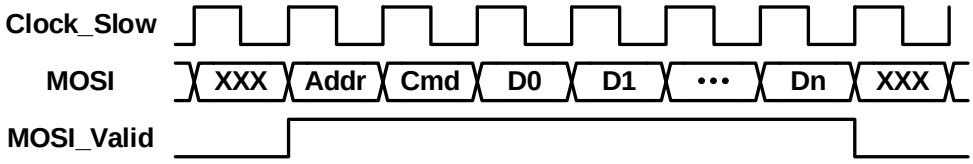

图 2 理想情况下 FPGA 向 ICARUS 传输数据时序图

图 2 给出了理想情况下 FPGA 向 ICARUS 传输数据 MOSI 侧的时序图，ICARUS 向 FPGA 返回数据 MISO 侧的时序与之相同，只是数据传输方向相反。当 FPGA 开启数据传输时，首先将数据有效表示 MOSI_Valid 拉高，同时向 ICARUS 发送数据。在发送有效数据 D0，D1，…，Dn 之前，FPGA 在前两个时钟周期内先发送控制位。其中，Addr 包含了需要访问的片上地址，Cmd 包含了数据传输方向和 Busrt 长度。当最后一周期的有效数据 Dn 发送完成后，MOSI_Valid 拉低代表整个数据传输过程结束。

由于存在数据传输的慢时钟域和片上工作的快时钟域，因此需要跨时钟域处理。ICARUS 片上只存在一组简单的异步 FIFO 用于跨时钟域。跨时钟域完成后，16 位数据拼接为 64 位数据，用于读写 ICARUS 片上的全局存储器(On-Chip SRMA)、存内单元(CIM)和控制寄存器(Control Register)。

B．测试现象与原因分析

现象一：芯片测试过程中，Bus 连续向片上 SRAM 存储器写数据，后再读出数据。从而根据两次数据是否一致，验证 Bus 功能是否正确。实验发现，Bus 读出之前写入的数据时，有大约 50%-70%的数据不一致。此外，一段地址空间内，数据全部出错；随后一段地址空间，数据全部正确；随后再出错，如此往复。

现象二：Bus 在向控制寄存器写数据时，某一控制寄存器内数据出错；或连续寄存器某几位或全部位错位，即向寄存器 A、B、C、D 写入数据 a、b、c、d，但寄存器 B、C、D 内的数据为 a、b、c 或 a、b、c 中的某几位。但复位后重新写入寄存器数据时，Bus 可能正常工作，此时 Burst 长度为 0。

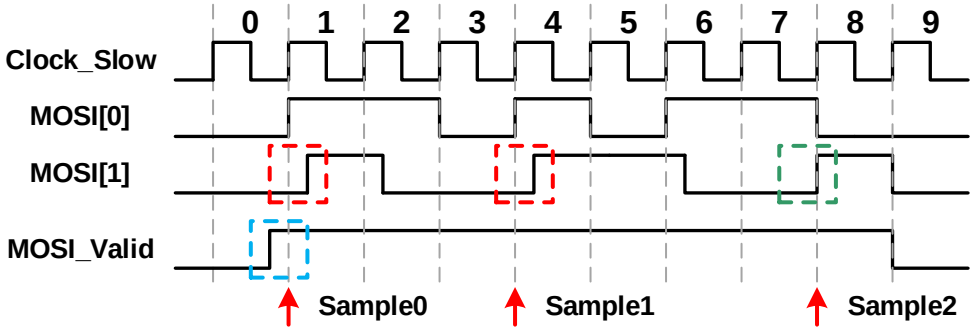

图 3 实际情况下并行 Bus 各数据位和有效位不与时钟上升沿对齐

产生上述两个现象的原因可能是跨时钟域处理不完善。在图 2 给出的理想时序图中，16 位 Bus 中各数据位及有效位的数据跳变严格与慢时钟上升沿对齐。但是在实际电路实现时，由于 PCB 开发板走线长度不同等原因，各数据位与有效位到达芯片的时间不同，其数据跳变可能不与慢时钟上升沿对齐，图 3 给出了这一现象的示意图，其中红色虚线框为数据跳变滞后于时钟上升沿，蓝色虚线框为超前，绿色虚线框为对齐。当数据有效位早于时钟

上升沿到达时(图 3 所示)，不影响芯片从 MOSI 上采样数据。相反，若数据有效位晚于时钟上升沿时，第一个周期对应的 MOSI 上的有效数据可能不会被采样。此外，MOSI 上各数据位之间、各数据位与时钟之间，时序也可能不对齐。图 3 中，MOSI 第 1 位数据跳变(MOSI[1])在第一次和第二次采样时均滞后于时钟上升沿，如此本应采样的数据 11 被采样为 01。

综上，ICARUS 芯片直接使用异步 FIFO 对慢时钟域到达的数据采样，会造成数据错位的问题。MISO 的数据传输过程也可能产生类似错误。再下次流片时，需要完善并行 Bus 跨时钟域处理。

C. 跨时钟域改进

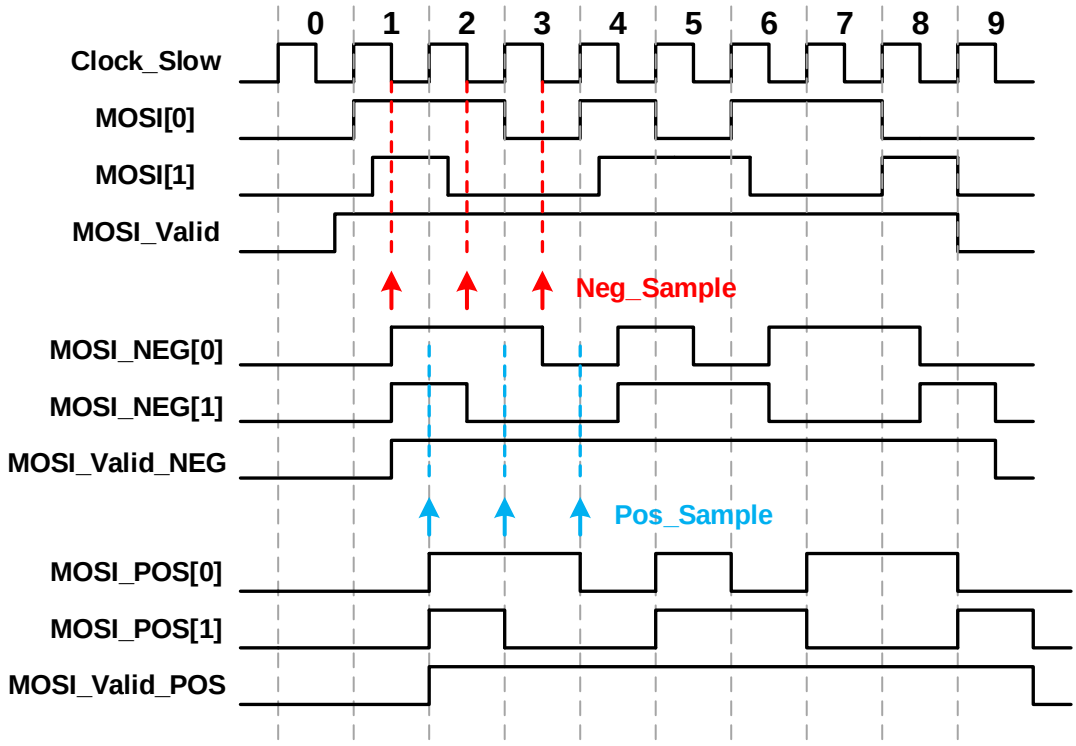

图 4 异步 FIFO 采样前数据跳变与时钟上升沿对齐

在芯片使用异步 FIFO 对慢时钟域到达的数据进行采样时，首先需要对该数据进行预处理，以使其数据跳变与时钟上升沿尽可能对齐。本小节建议的方法如图 4 所示。首先，芯片在慢时钟的下降沿对数据进行第一次采样(NEG)，此时数据稳定，采样后的各数据位与片上慢时钟下降沿对齐。随后，芯片在慢时钟的上升沿对第一次采样数据再次采样，如此得到的数据才在片内与慢时钟完全对齐。随后才能被异步 FIFO 采样，完成后续的跨时钟域操作。

1.2 串行 SPI 接口

A. 原理与时序

为防止并行接口不能正常工作时，依旧可以与片上完成数据交互，本次流片额外在 ICARUS 芯片上例化了串行 SPI 接口，如图 5 所示。其中片上 SPI 为 Slave，FPGA 为 Master。标准 SPI 接口支持四种传输模式(Mode-0 至 Mode-3)，本次流片的 SPI 接口仅支

持 Mode-0 模式，即时钟极性 CPOL=0，时钟相位 CPHA=0。时钟信号 SPI_CLK，选通信号 SPI_CSN，数据信号 SPI_MOSI、SPI_MISO 对应的时序与标准 Mode-0 SPI 时序相同，如若不了解 SPI 协议，请自行查阅有关资料。

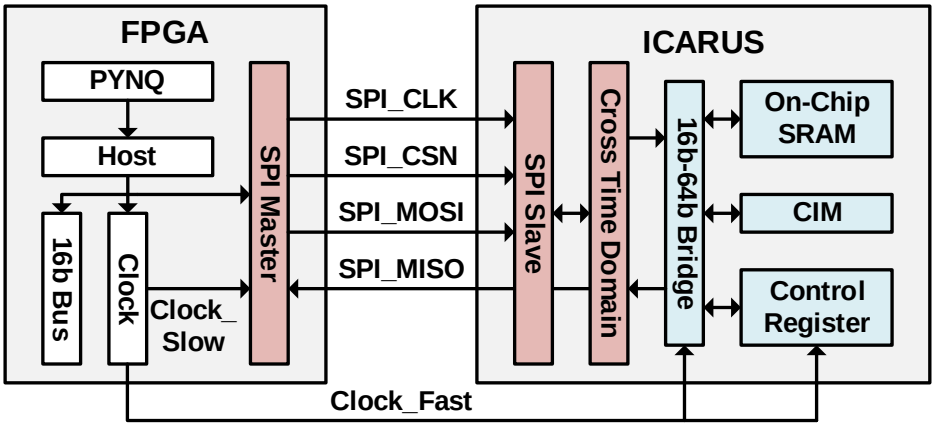

图 5 串行 SPI 接口框图

B. 现象

SPI 接口在在慢时钟(Clock_Slow)为 5-50MHz 下均可正常工作，此时对应的 SPI_CLK 时钟频率为 2.5-25MHz。

C. 改进

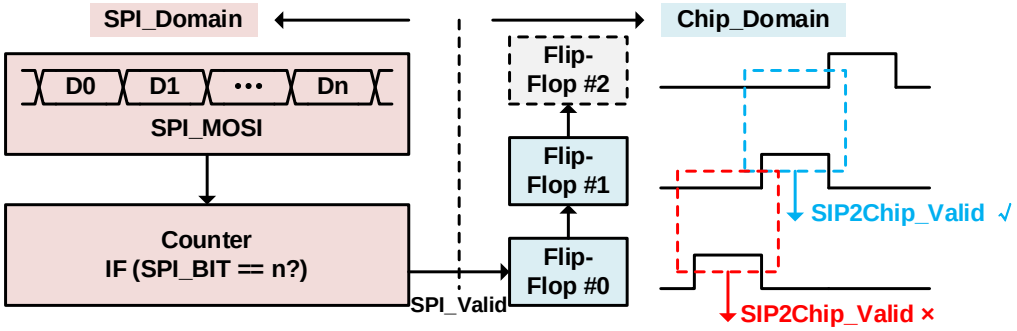

图 6 SPI 跨时钟域错误与改进

尽管此次流片的 SPI 接口可以正常工作，但跨时钟域处理时仍有错误。如图 6 所示，不同于使用异步 FIFO 进行跨时钟域的并行 Bus 接口，串行 SPI 使用采样(即打两排)的方法完成跨时钟域。当 FPGA 向 SPI 传输串行数据时，片上对应的 SPI 计数器开始计数。当计数结果与 SPI 传输的串行数据比特数相同时，该计数器拉高 SPI_Valid 信号，需要注意的是该 SPI_Valid 信号属于 SPI 时钟域(慢时钟域)。需要将其跨时钟域后，才能真正表示完成串行数据传输，即此时串行数据已经在移位寄存器中转为多位数据。使用打两排的方法对 SPI_Valid 采样，从而完成跨时钟域。首先使用寄存器#0 对来自 SPI 时钟域的数据进行采样，随后使用另外两个寄存器#1 和#2 对采样到的数据进行快时钟同步。当寄存器#1 对应的 Valid 信号为 1，而寄存器#2 对应的 Valid 信号为 0 时，SPI2Chip_Valid 拉高，代表跨时钟域完成。而在原设计中，只使用了寄存器#0 和#1，可能会引起时序上的不稳定。

2. 信号复位

本次流片中的慢时钟域(包括 SPI 接口中的 SPI_CLK)和快时钟域各自对应一个异步复位信号，该复位信号来自 PCB 测试版上的按键。在之后的流片中，依旧建议将复位信号分开，同时将异步复位改为异步复位同步释放。如若查询有关异步复位和同步复位的 Verilog 代码写法及优缺点等，请自行搜索。本章节着重介绍异步复位同步释放的内容。

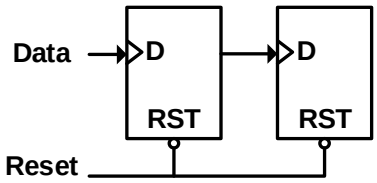

图 7 异步复位

如图 7 所示，单独使用异步复位时，复位信号在释放时(发生跳变)恰好处于时钟上升沿，此时寄存器接受到的复位信号处于亚稳态，即可能为 0 或 1。若有效数据又恰好传入，此时寄存器即可能对数据进行采样，又可能仍处于复位状态。使用异步复位同步释放可以解决这一问题。

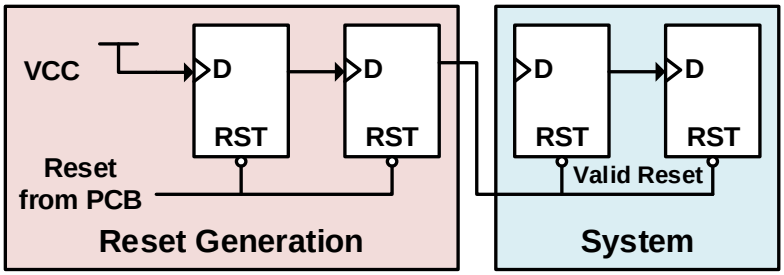

图 8 异步复位同步释放

所谓异步复位，同步释放就是在复位信号到来的时候不受时钟信号的同步，而是在复位信号释放的时候受到时钟信号的同步。图 8 给出了异步复位同步释放的示意图。以复位信号低电平有效为例(实际流片时，复位信号高电平有效)，当来自 PCB 的复位信号为 0 时，此时复位产生模块的寄存器被复位，产生低电平 0，因此系统模块中的寄存器也被正常复位，即异步复位。当 PCB 上的复位信号恰好在时钟上升沿释放时，复位产生模块中的第一级触发器处于亚稳态，但是由于两级触发器的缓冲作用，第二级触发器的输入为时钟到来前第一级触发器的输出，此时第二级触发器的输出仍为低电平 0，系统模块中的寄存器仍处于复位。当下一个周期，复位产生模块两级寄存器均脱离亚稳态，此时系统模块的复位信号拉高，系统完成复位。上述过程即同步释放。

3. 芯片测试

本次流片优先完成了电路功能模块的 Verilog 代码编写。在陈迟晓老师的提示下，加入了用于 Debug 调试的有关功能，这在芯片测试过程中被证明是十分有必要的。本章节特此总结了电路功能测试和性能测试所需的额外芯片模块，此外还总结了芯片哪些引脚需要在 PCB 测试板上引出，以便于芯片测试。

3.1 芯片上有关电路功能测试的模块

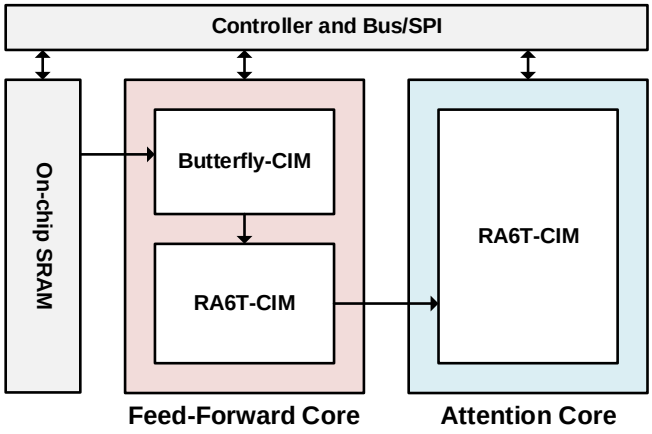

图 9 芯片功能模块简略

首先，并行 Bus 和串行 SPI 互为备份。二者可以访问共同的地址空间，以防止某一接口功能不正确后，仍能对芯片进行读写。

其次，要根据芯片数据流、控制流等对电路各模块进行划分。对于各个划分后的单元，接口需要能够对其进行读写操作，以单独验证其功能。图 9 给出了本次电路流片的功能模块简略图。FF-Core 从全局数据存储器中读取输入数据，完成运算，并将计算结果传入 ATTN-Core 中完成后续运算。本次流片，接口除了能对控制寄存器、片上全局存储器完成读写操作外，还能单独对 FF-Core 的输入、输出，ATTN-Core 的输入输出进行读写，以便于在芯片测试时单独验证两个模块的功能、性能。

最后，本次流片仍有一些测试功能不够完善。首先，FF-Core 包含 Butterfly 和已有的 RA6T 存内单元，二者串行工作。芯片不能单独对 Butterfly 模块进行读写，难以验证中间结果是否出错。其次，不同模块之间的 Clock Gating 没有认真考虑，即单独测试 FF-Core 或 ATTN-Core 的诸如电压、电流、功耗等性能指标时，另一模块中的某些电路单元和时钟网络不能完全关断。希望之后流片的有关同学能够对类似教训引起重视。

3.2 芯片性能测试注意事项

在测试芯片功能等性能时，芯片需要长时间的进行运算。以便在芯片稳定工作时，PCB 测试板上的相关电路有足够时间对电流、电压等进行采样。

本次流片，芯片上有专门的引脚返回其工作状态。当 FPGA 通过该引脚读到芯片完成计算后，马上发出新一轮计算的指令(配置寄存器)，随后芯片开始工作。当 FPGA 发出指令时，

芯片不处理任何数据。这就要求每一次芯片开启计算时，其对应的时钟周期远大于指令传输的周期，即保证芯片工作周期占比在 90%以上。此外，芯片重复计算时，不能破坏片上原有的输入数据，如芯片输出数据覆盖了输入数据等。否则，不能正确验证该情景下的电路功能。

此外，芯片上也可设置额外模块用于性能测试。例如该模块定义了芯片处理同一数据的次数(Loop)，如此在 FPGA 发出一次相关指令后，芯片可以一直处于工作状态。

3.3 PCB 测试板引出的引脚

为了能够使用示波器正确读到芯片引脚的状态，需要将引脚引入到 PCB 开发板上。否则，直接使用示波器探针接触引脚，读到的信号质量很差，且可能是引脚弯曲折断。

本次流片上对应的 PCB 测试板引入了以下引脚：

信号类引脚：

A. SPI 接口对应的四个引脚(SPI_CLK、SPI_CSN、SPI_MOSI、SPI_MISO)；
B. Bus 接口对应的引脚(MOSI_Valid、MOSI[3:0]、MISO_Valid、MISO[3:0])；

状态类引脚：

A. Bus 接口对应片上两个异步 FIFO 的 Empty，Full 信号；
B. 有关芯片 4 个工作模式的引脚，即是否某一模式处于工作状态，某一模式是否完成。共 4 个引脚；

时钟类引脚：

A. FPGA 传入 ICARUS 芯片的快时钟 Clock_Fast、慢时钟 Clock_Slow；
B. ICARUS 芯片返回的快时钟 Clock_Fast_out、慢时钟 Clock_Slow_out；芯片返回时钟既可以用于反应时钟是否能被灌入芯片，又可以用于芯片返回数据的同步。

在实际测试过程中，信号类引脚主要在测试初始阶段使用，以帮助验证接口功能是否正常；有关芯片状态的引脚会经常用到，便于 FPGA 判断芯片工作状态，并快速传入新的指令；时钟类引脚也经常用到，用于 Frequency-Voltage Scaling 中测试芯片性能。

在芯片流片前后，相关同学需要根据 PCB 实际布线(是否仿的下所有引脚)，芯片待测功能(与那些引脚有关)，来决定那些引脚需要引出在 PCB 上。