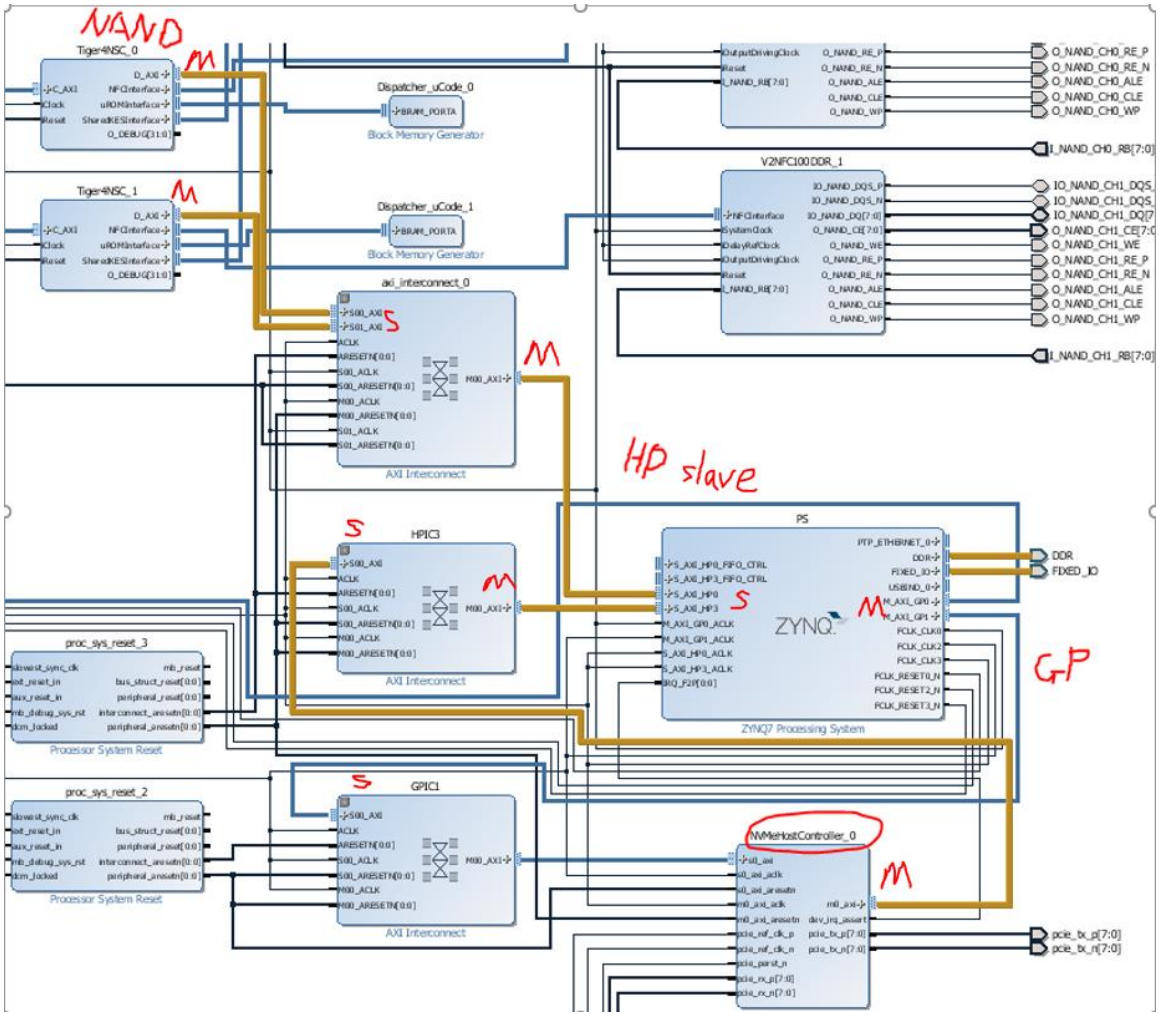
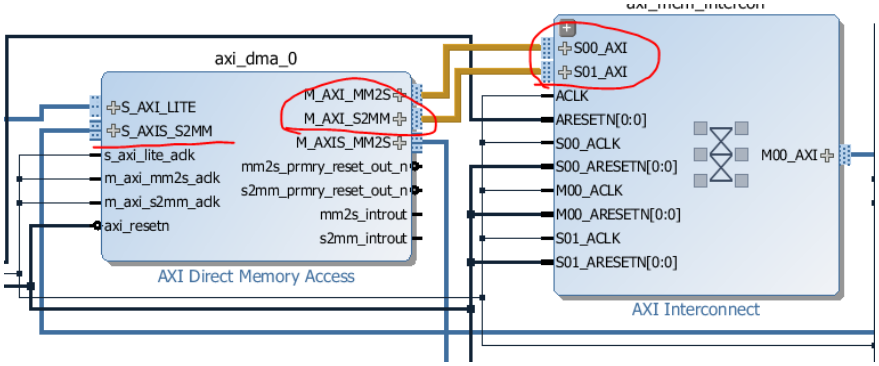
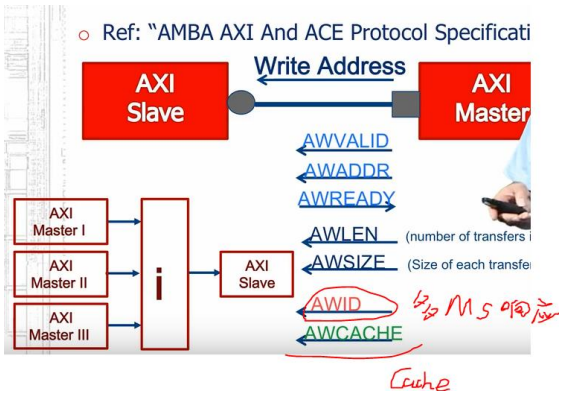


ILA: https://www.youtube.com/watch?v=acpIiRTxzcM



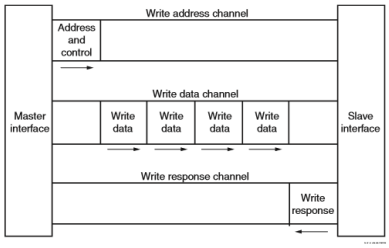
AXI 总线信号：

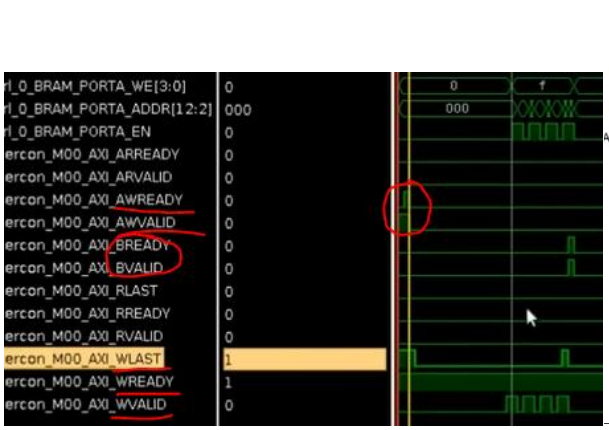
地址 Write address (注意 awid 和 awcache)

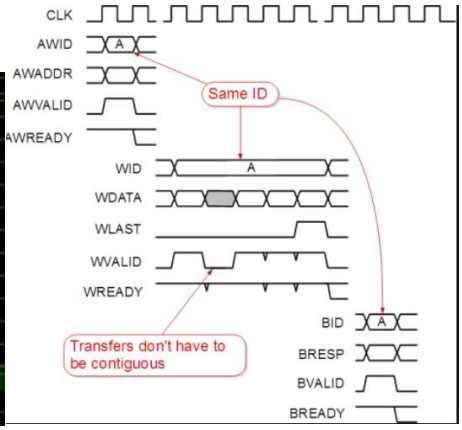


https://www.youtube.com/watch?v=sl8xCMu39Mk

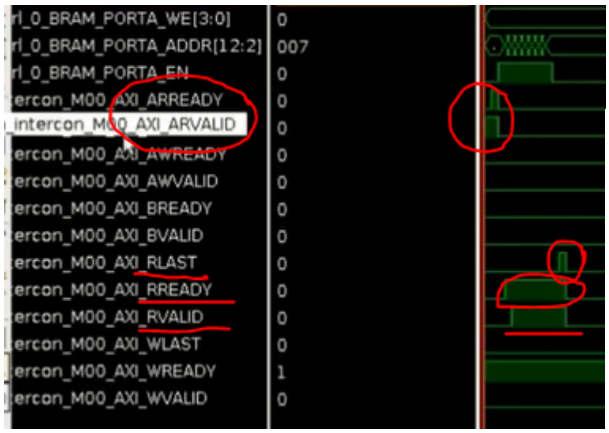
AXI DMA 写操作:

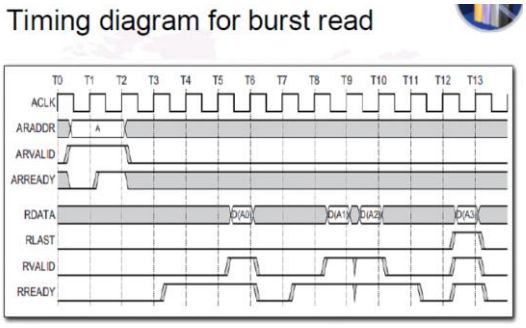


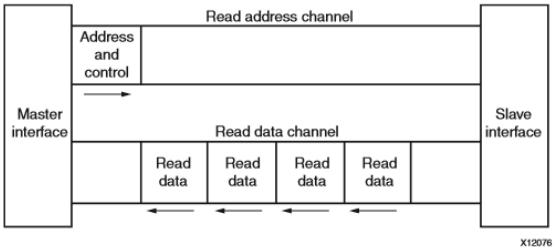




AXI DMA 读操作:







SoC, Zqny

MMU:

cacheable / non-cacheable / write-back / write-through

virtual memory addresses to physical memory addresses.

虚拟地址解释

A critical point of understanding is that multiple software applications try to reference memory at the same address locations because they are all compiled and linked in the same way, and have no knowledge of each other. If no memory translation were to take place in the processor, an application could potentially interfere with another because they all reference the same addresses in memory. The MMU solves this problem by fooling the applications into believing that they are accessing the memory addresses that they request, but then translate the addresses in such a way that each user application has its address space mapped to a different location in physical memory. This explains the basic concepts of virtual memory (which is what the user application sees), and physical memory (which is what the processor hardware sees). In the example of an operating system, these mappings of memory space from the

virtual to physical addresses may need to constantly change as new applications are loaded and closed, so the translation tables are held in a “soft” format in main memory. This dedicated “soft” region of memory is known as the translation table or page table, and is usually controlled by the operating system’s kernel.

a function which is designed to **update the MMU page tables:**

void Xil_SetTlbAttributes(u32 addr, u32 attrib);

so the purpose of the TLB is essentially a cache of the page tables.

TLB：Translation lookaside buffer,即旁路转换缓冲，或称为页表缓冲；里面存放的是一些页表文件（虚拟地址到物理地址的转换表）。又称为快表技术。由于“页表”存储在主存储器中，查询页表所付出的代价很大，由此产生了 TLB。

For example, **if the developer wants to set the region of memory starting at address 0x20000000 to be non-global, non-cacheable, and have access permissions in “privileged mode” only,**

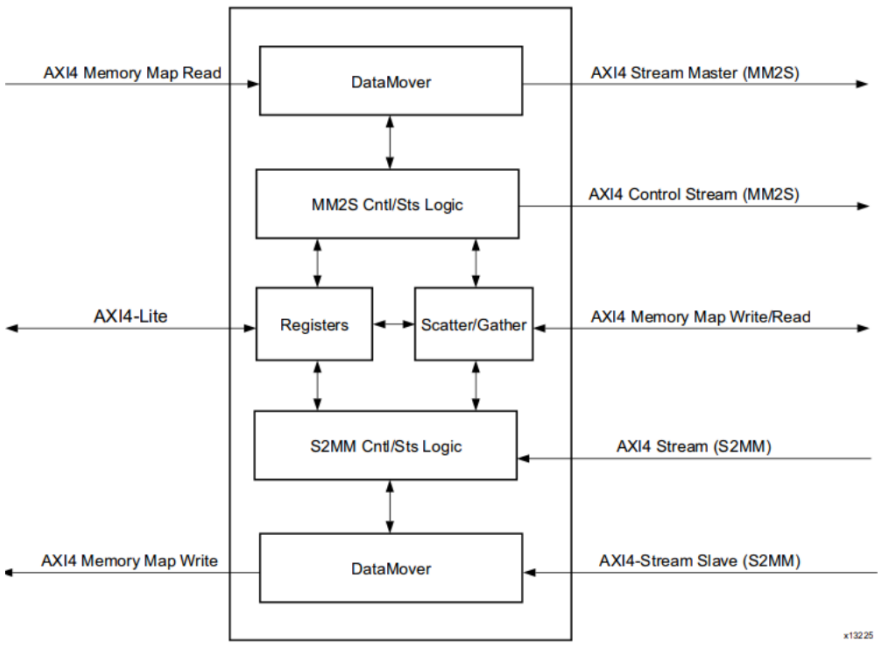
adjust_mmu_mode(0x20000000, NON_GLOBAL+NON_CACHEABLE+AP_PRIVIEGED_ACCESS_ONLY);

https://lauri.xn--vsandi-pxa.com/hdl/zynq/xilinx-dma.html

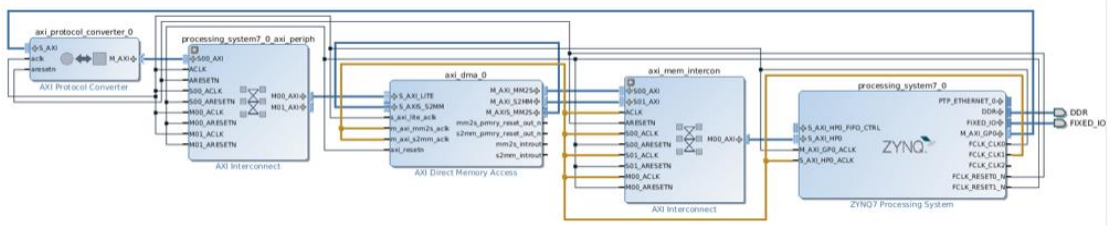
AXI DMA distinguishes two channels:

MM2S (memory-mapped to stream) transports data from DDR memory to FPGA

S2MM (stream to memory-mapped) transports arbitrary data stream to DDR memory.



AXI DMA internals



DMA SDK source code: (sg mode/simple mode)

https://github.com/Xilinx/embeddedsw/tree/master/XilinxProcessorIPLib/drivers/axidma/examples

Zynq DMA demo Vivado step by step:

https://www.youtube.com/watch?v=Yklu68WopBo&t=1201s

PS 到 PL 的高速与低速接口：https://pynq.readthedocs.io/en/v2.3/overlay_design_methodology/pspl_interface.html

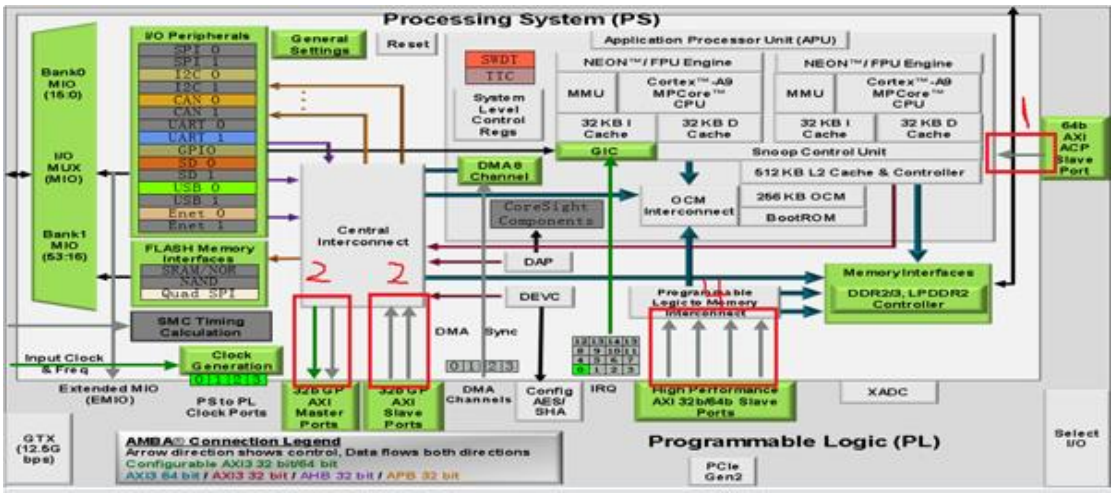
ZYNQ 中的 AXI 接口共有 9 个，主要用于 PS 与 PL 的互联，包含以下三个类型：

. AXI_ACP 接口，是 ARM 多核架构下定义的一种接口，中文翻译为加速器一致性端口，用来管理 DMA 之类的不带缓存的 AXI 外设，PS 端是 Slave 接口。

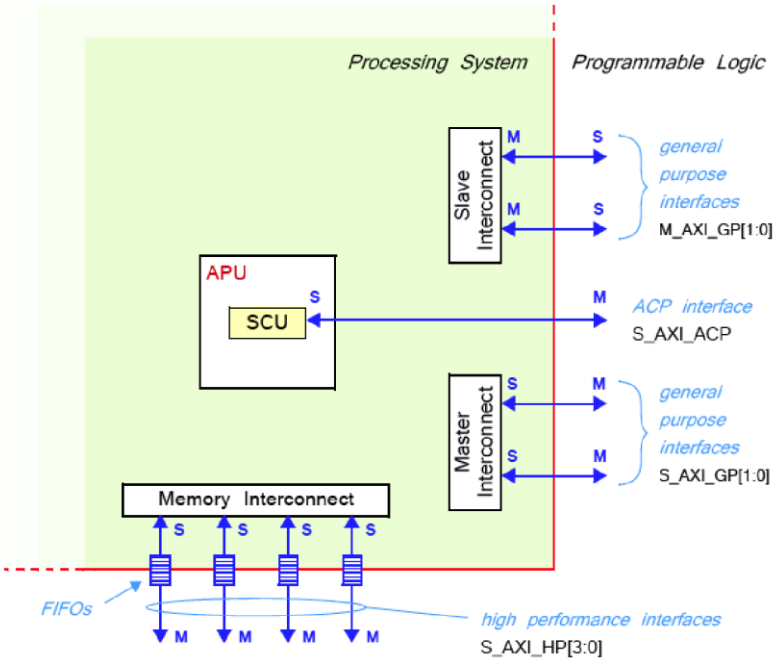
. AXI_HP 接口，是高性能/带宽的 AXI3.0 标准的接口，总共有四个，PL 模块作为主设备连接。主要用于 PL 访问 PS 上的存

储器（DDR 和 On-Chip RAM）

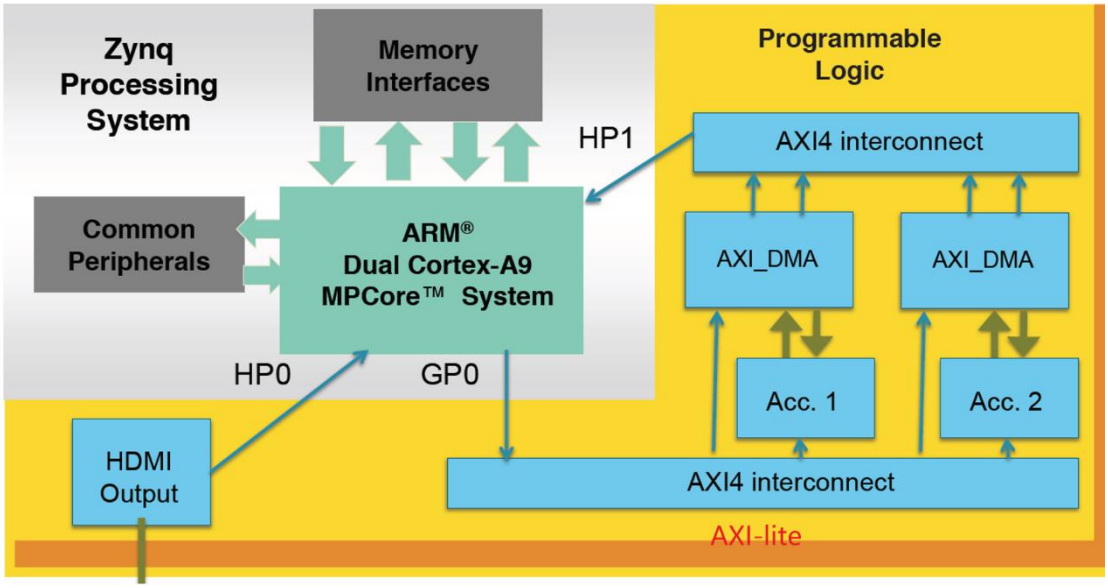
. AXI_GP 接口，是通用的 AXI 接口，总共有四个，包括两个 32 位主设备接口和两个 32 位从设备接口。



The Zynq has 9 AXI interfaces between the PS and the PL. On the PL side, there are 4x AXI Master HP (High Performance) ports, 2x AXI GP (General Purpose) ports, 2x AXI Slave GP ports and 1x AXI Master ACP port. There are also GPIO controllers in the PS that are connected to the PL.



file:///C:/Users/lidon/OneDrive/文档/面试题目/C%20firmware/AXI_interface_Port.pdf



https://blog.csdn.net/zhipao6108/article/details/81587117