EURESYS™
Excellence in vision

用户指南

CustomLogic

Coaxlink 11.2.1

3602 Coaxlink Octo

3603 Coaxlink Quad CXP-12

使用条款

本文档随Coaxlink11.2.1(doc build) 提供。

目录

1. CustomLogic 简介

警告 本规范为初步规范，可能会有所更改！

1.1. 原则

CustomLogic允许用户在装有“CustomLogic”固件变体的Euresys帧捕获器的FPGA(现场可编程门阵列)中，添加自定义的板载图像处理。

CustomLogic包括一个可提供文档化接口的设计框架，其用于将自定义图像处理功能与帧捕获器互连。

1.2. 可用性

备注 CustomLogic安装包仅在向本地 销售办公室 提出请求时可用。

CustomLogic可用于以下产品和固件变体：

3602 Coaxlink Octo

固件变体	描述
3602 Coaxlink Octo	□ CXP-6 DIN 4 Coaxress接口 □ 一个1-或2-或4-连接区域扫描相机 □ 2 GB RAM DDR4 板载内存 □ 8-通道Gen 3 PCI Express接口

3603 Coaxlink Quad CXP-12

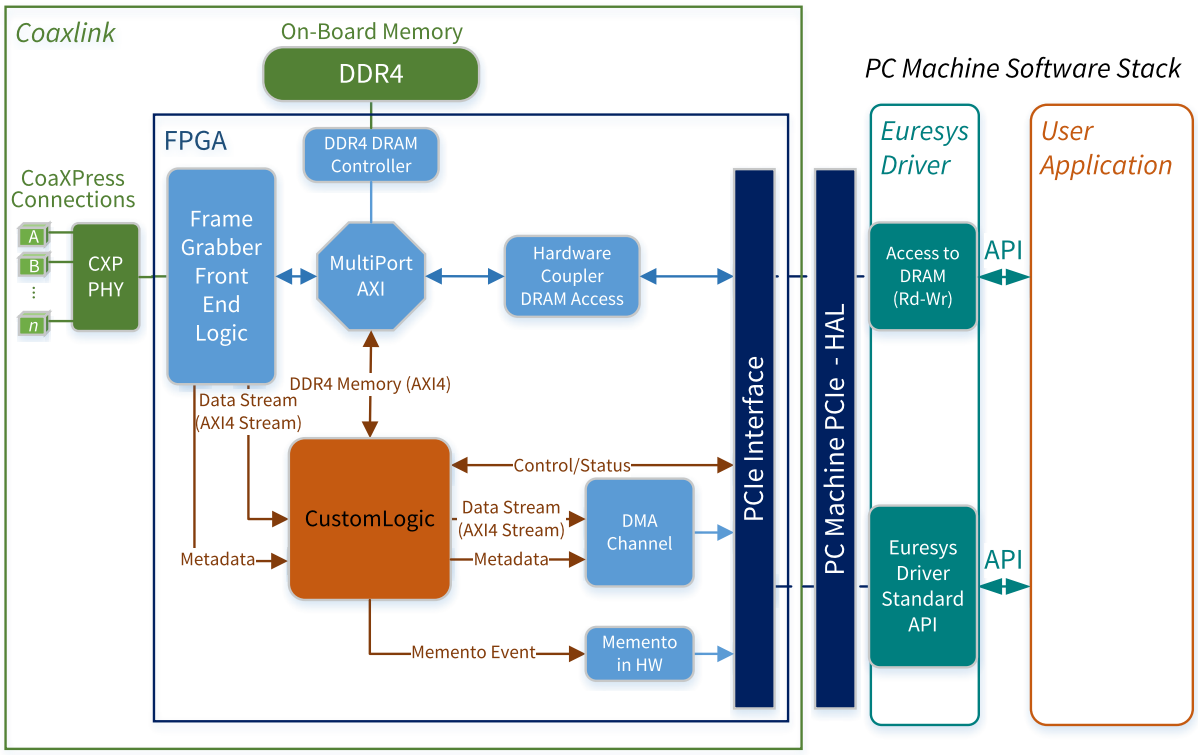
固件变体	描述
3603 Coaxlink Quad CXP-12	□ CXP-12 HD-BNC 4 CoaXPress 接口 □ 一个1-或2-或4-连接区域扫描相机 □ 2 GB RAM DDR4 板载内存 □ 8-通道Gen 3 PCI Express接口

1.3. 框架

一个 Coaxlink 框架提供以下内置模块：

1. 全功能CoaXPress帧捕获器。
2. 板载存储器接口。
3. 带有DMA后端通道的PCI Express接口。
4. “硬件备忘录”事件记录系统。
5. 用户通过Euresys驱动程序API注册访问。

示例方框图



Coaxlink CustomLogic 模型

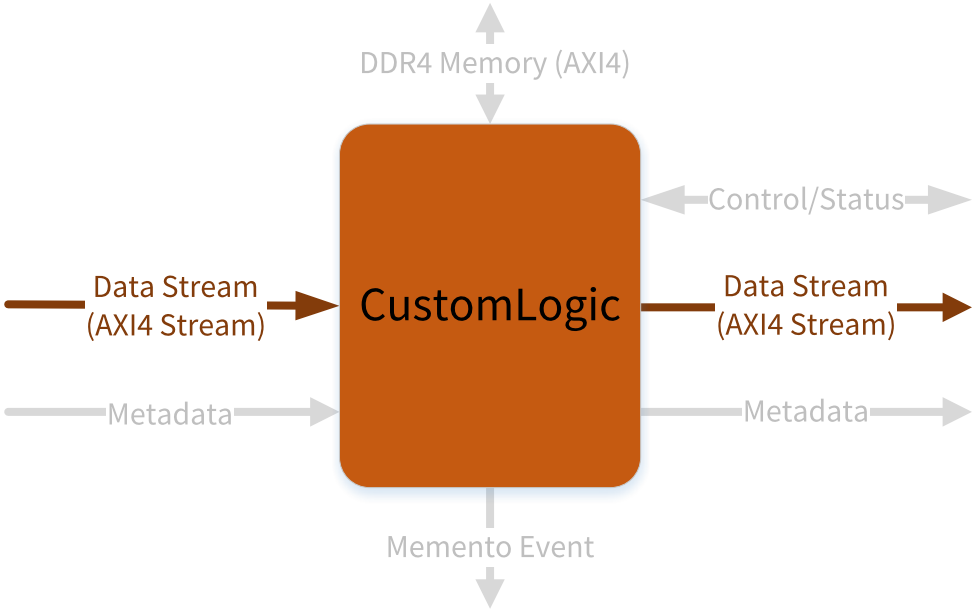
2. CustomLogic 接口

警告 本规范为初步规范，可能会有所更改！

2.1. 数据(像素)流接口

数据流接口基于 AMBA AXI4-流协议规范：

- 在源端，此接口向 *CustomLogic* 提供从Coaxress设备(例如Coaxress相机)获取的图像。
- 在目标端，数据流接口将 *CustomLogic* 生成的结果图像/数据传输到“PCI Express DMA Back-End”通道。



源接口信号

信号	指令	描述
axis_tvalid_in	IN	TVALID 表示 **源** 正在驱动有效传输。当Tvalid和Tready都被置位时，就会发生转移。
axis_tready_in	OUT	TREADY 表示 *CustomLogic* 可以接受当前循环中的传输。
axis_tdata_in [*]	IN	TData是主要的有效负载，用于提供通过接口传递的数据。 *[*] = [127:0] 针对* Octo *或 [255:0] 针对*
axis_tuser_in [3:0]	IN	Tuser是用户定义的边带信息，可以与数据流一起传输。Tuser内容编码如下： □ axis_tuser_in[0] => Start-of-Frame (SOF) □ axis_tuser_in[1] => Start-of-Line (SOL) □ axis_tuser_in[2] => End-of-Line (EOL) □ axis_tuser_in[3] => End-of-Frame (EOF)

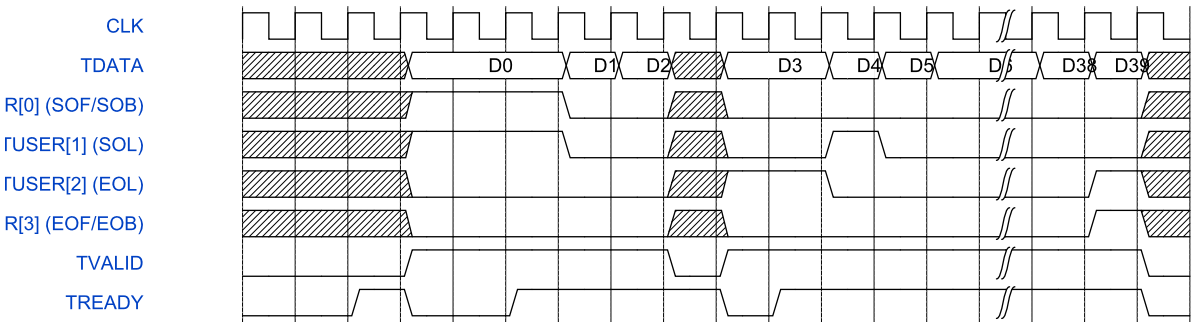
目的接口信号

信号	指令	描述
axis_tvalid_out	OUT	TVALID 表示 *CustomLogic* 正在驱动有效传输。当Tvalid和Tready都被置位时，就会发生转移。
axis_tready_out	IN	TREADY 表示 **PCI Express DMA Back-End** 可以接受当前循环中的传输。
axis_tdata_out [*]	OUT	TData是主要的有效负载，用于提供通过接口传递的数据。 *[*] = [127:0] 针对* Octo *或 [255:0] 针对*
axis_tuser_out [3:0]	OUT	Tuser是用户定义的边带信息，可以与数据流一起传输。Tuser内容编码如下： □ axis_tuser_out[0] => Start-of-Buffer (SOB) □ axis_tuser_out[1] => *Reserved* □ axis_tuser_out[2] => *Reserved* □ axis_tuser_out[3] => End-of-Buffer (EOB)

在 *CustomLogic* 目的端，*axis_tuser_out* 信号具有控制 PCI Express DMA Back-End的功能。*axis_tuser_out* 带有的标志解释如下：

- *Start-of-Buffer*：包含此标志的循环将启动新的缓冲区。
- *End-of-Buffer*：包含此标志的循环将结束缓冲区，即使缓冲区仍有可容纳新传输的空间。

时序图



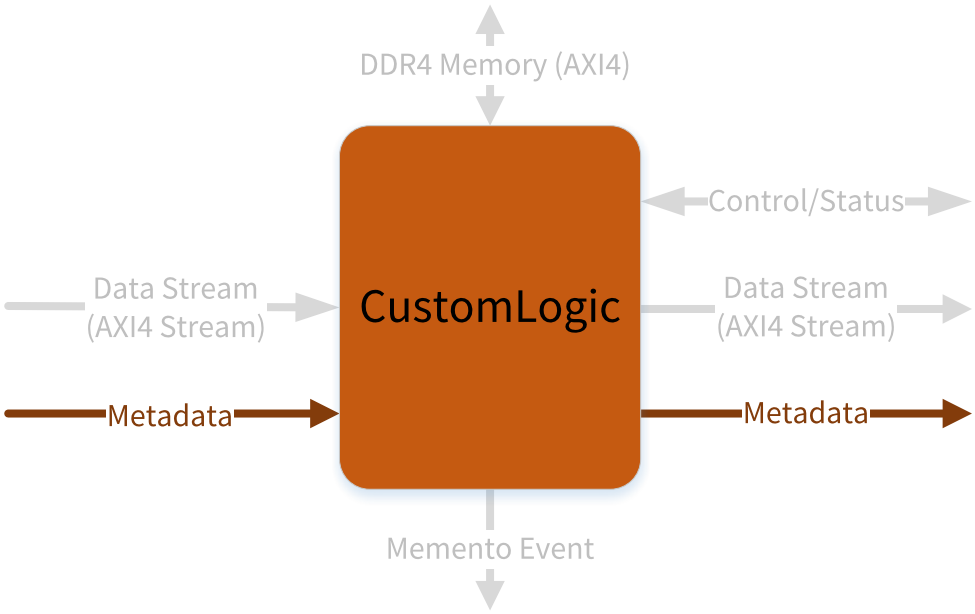
TVALID/TRADY信号交换和TUSER标志时序图

在这个例子中，我们认为 *LinePitch* 是64字节(4个传输周期，每个16字节)，而整个帧由10行/包组成。

有关AXI4-流协议的更多信息，请参阅 www.xilinx.com 上的Xilinx Axi参考指南和 www.amba.com上的AMBA AXI4-流协议规范。

2.2. 元数据接口

在元数据接口上显示由CoaXPress设备生成的CoaXPress图像头。除了CoaXPress图像头外，元数据接口还提供有关像素对齐、时间戳...



接口信号

元数据接口信号显示为名为 *Metadata_rec* 的VHDL记录类型(此记录类型的定义可在 *CustomLogicPkg* 包 (`CustomLogicPkg.vhd`中找到))。

- 在源端，元数据接口显示前缀 *Metadata_in*，并有方向 *IN*。
- 在目标端，它显示前缀 *Metadata_out* 并具有方向 *OUT*。

在元数据接口中有两组信号：

- CoaXPress图像头组–包含由CoaXPress设备发布的CoaXPress矩形图像头副本的信号。
- CustomLogic组–通知时间戳和数据流特征的信号。

CoaXPress图像头组

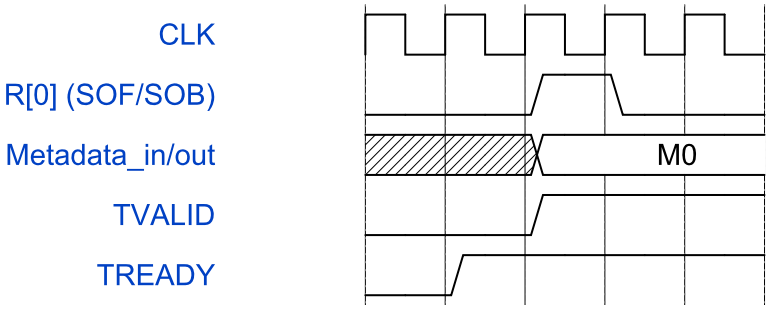
信号	描述
[prefix].StreamId [7:0]	*唯一流ID。*
[prefix].SourceTag [15:0]	*16位源图像索引。对于每个传输的图像递增，在0xFFFF处绕回至0。包含与同一图像相关的数据的每个流应使用相同的数字。*
[prefix].Xsize [23:0]	*以像素表示图像宽度的24位值。*
[prefix].Xoffs [23:0]	*24位值，表示图像相对于整个设备图像的左侧像素的水平偏移量(以像素为单位)。*
[prefix].Ysize [23:0]	*以像素为单位，表示图像高度的24位值。对于行扫描图像，该值应设置为0。*
[prefix].Yoff [23:0]	*24位值，表示图像相对于完整设备图像顶行的垂直偏移，(以像素为单位)。*
[prefix].DsizeL [23:0]	*24位值，表示每条图像行的数据字数。*
[prefix].PixelF [15:0]	*表示像素格式的16位值。*
[prefix].TapG [15:0]	*表示外螺纹几何图形的16位值。*
[prefix].Flags [15:0]	*图像标志。*

备注 斜体绿色 的描述摘自CoaXPress标准版本1.1.1。

CustomLogic 组

信号	描述
[prefix].Timestamp [31:0]	设备的读取开始事件的时间戳。
[prefix].PixProcessingFlgs [7:0]	像素处理标志： □ PixProcessingFlgs[0]=>已启用“RGB到BGR”交换。 □ PixProcessingFlgs[1]=>已启用“MSB像素对齐”。 □ PixProcessingFlgs[2]=>已启用“压缩采集”。 □ PixProcessingFlgs[7:4]=>LUT配置。
[prefix].Status [31:0]	32位向量，可由 *CustomLogic* 用于报告其状态。 **备注** 在源端，状态值为0x00000000。

时序图



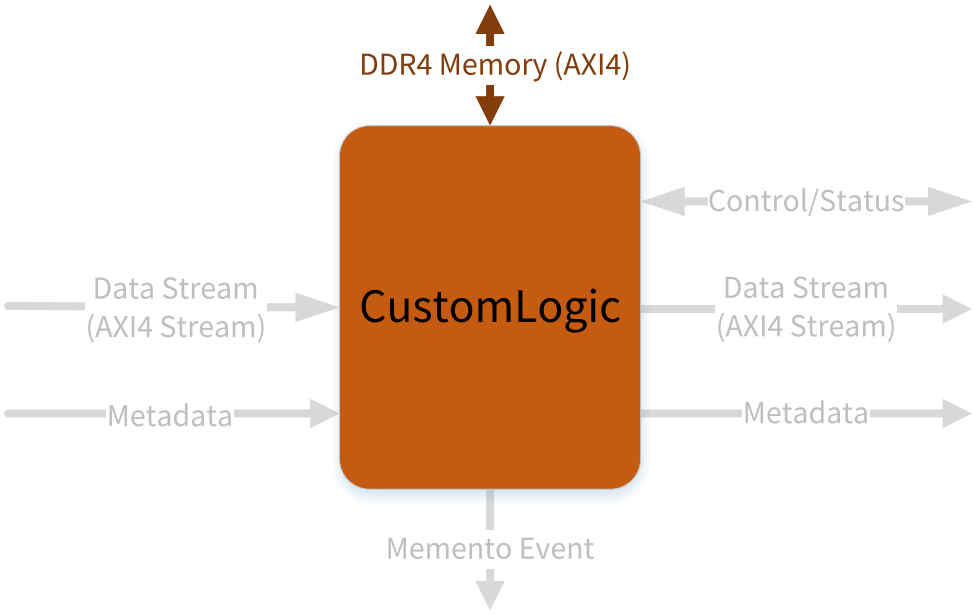
元数据时序图

元数据信号在每次声明 *TUSER* 标志SOF/SOB时都会更新。

警告 *CustomLogic* 不应修改记录类型 *Metadata_rec*中包含的信号内容。

2.3. DDR4内存接口

DDR4板载内存接口基于 AMBA AXI4-流协议规范：

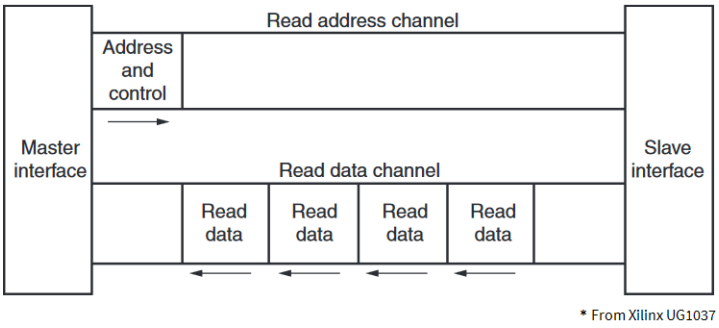


AXI4是一个内存映射接口，由五个通道组成：

- 写入地址通道
- 写入数据通道
- 写入响应通道
- 读取地址通道
- 读取数据通道

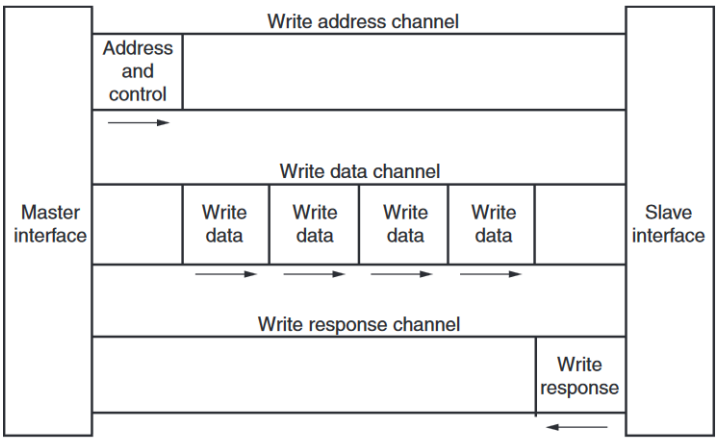
数据可以在主设备和从设备之间同时向两个方向移动，数据传输大小也可不同。在AXI4中的限值：最多256次数据传输的突发事务。

AXI4通道架构



* From Xilinx UG1037

Channel Architecture of Reads



* From Xilinx UG1037

Channel Architecture of Writes

AXI4信号描述

以下部分简要地描述了AXI4信号。

备注 若需完整了解有关信号、接口要求和事务属性内容，请参阅www.amba.com上的“AMBA AXI和ACE协议规范”文档。

主写入地址通道接口信号

信号	指令	描述
m_axi_awaddr [31:0]	OUT	*写入地址写入地址给出了写入突发事务中第一个传输地址。*
m_axi_awlen [7:0]	OUT	*突发长度突发长度给出了突发中传输的确切次数。该信息决定了与地址相关联的数据传输次数。* *Burst_Length = AWLEN[7:0] + 1*
m_axi_awsize [2:0]	OUT	*突发大小该信号表明了突发中每次传输的大小。* *Burst_Size = 2^AWSIZE[2:0]*
m_axi_awburst [1:0]	OUT	*突发类型。突发类型和大小信息决定了如何计算突发内每次传输的地址。* *Burst_Type: "00" = FIXED; "01" = INCR; "10" = WRAP*
m_axi_awlock	OUT	*锁类型提供有关传输原子特性的其他信息。* *Atomic_Access: '0' Normal; '1' Exclusive*
m_axi_awcache [3:0]	OUT	*内存类型此信号表示如何在系统中进行事务。* *Memory_Attributes:* □ *AWCACHE[0] 可缓冲* □ *AWCACHE[1] 可缓存* □ *AWCACHE[2] 读取分配* □ *AWCACHE[3] 写入分配*
m_axi_awprot [2:0]	OUT	*保护类型。该信号表示事务特权和安全级别，以及事务是数据访问还是指令访问。* *访问权限：* □ *AWPROT[0] 授权* □ *AWPROT[1] 无安全措施* □ *AWPROT[2] 指令*
m_axi_awqos [3:0]	OUT	*服务质量，QoS。为每个写入事务发送的QoS标识符。* *Quality_of_Service: 优先级*
m_axi_awvalid	OUT	*写入地址有效此信号表示信道正在发送有效的写入地址和控制信息。*
m_axi_awready	IN	*写入地址准备完毕此信号表示从机已准备好接受地址和相关控制信号。*

备注 这些描述摘自AMBA AXI和ACE协议规范。

主写入数据通道接口信号

信号	指令	描述
m_axi_wdata [*]	OUT	*写入数据。* *[*] = [127:0] 用于* Octo *或 [255:0] 用于*
m_axi_wstrb [**]	OUT	*写入选通该信号表示哪些字节通道保存有效数据。写入数据总线的每八位有一个写选通位。* *[**] = [15:0] 用于* Octo *或 [31:0] 用于*
m_axi_wlast	OUT	*最后写入此信号表示写入突发中的最后一次传输。*
m_axi_wvalid	OUT	*写入有效。此信号表示有效的写入数据和选通可用。*
m_axi_wready	IN	*写入准备就绪此信号表示从机可以接受写入数据。*

备注 这些描述摘自AMBA AXI和ACE协议规范。

主写入响应通道接口信号

信号	指令	描述
m_axi_bresp [1:0]	IN	*写入响应此信号表示写入事务的状态。* *响应：* □ *"00" = OKAY* □ *"01" = EXOKAY* □ *"10" = SLVERR* □ *"11" = DECERR*
m_axi_bvalid	IN	*写入响应有效此信号表示通道正在发出有效的写入响应信号。*
m_axi_bready	OUT	*响应准备就绪。此信号表示主机可以接受写入响应。*

备注 这些描述摘自AMBA AXI和ACE协议规范。

对于 *m_axi_bresp*：

- *OKAY*：正常访问成功。表示正常访问已成功。也可以表示独占访问失败。参见OKAY：正常访问成功。
- *EXOKAY*：独占访问成功。表示独占访问的读部分或写部分成功。
- *SLVERR*：从机错误。当访问成功到达从机，但从机希望向发起主机返回错误条件时使用。
- *DECERR*：解码错误。通常由互连组件生成，以表示在事务地址处没有从机。

主读取地址通道接口信号

信号	指令	描述
m_axi_araddr[31:0]	OUT	*读取地址。读取地址给出了读取突发事务中第一个传输地址。*
m_axi_arlen[7:0]	OUT	*突发长度突发长度给出了突发中传输的确切次数。该信息决定了与地址相关联的数据传输次数。* *Burst_Length = ARLEN[7:0] + 1*
m_axi_arlen[2:0]	OUT	*突发大小该信号表明了突发中每次传输的大小。* *Burst_Size = 2^ARSIZE[2:0]*
m_axi_arburst[1:0]	OUT	*突发类型。突发类型和大小信息决定了如何计算突发内每次传输的地址。* *Burst_Type: "00" = FIXED; "01" = INCR; "10" = WRAP*
m_axi_arlock	OUT	*锁类型提供有关传输原子特性的其他信息。* *Atomic_Access: '0' Normal; '1' Exclusive*
m_axi_arcache[3:0]	OUT	*内存类型此信号表示如何在系统中进行事务。* *Memory_Attributes:* □ *ARCACHE[0] 可缓冲* □ *ARCACHE[1] 可缓存* □ *ARCACHE[2] 读取分配* □ *ARCACHE[3] 写入分配*
m_axi_arprot[2:0]	OUT	*保护类型。该信号表示事务特权和安全级别，以及事务是数据访问还是指令访问。* *Access_Permissions:* □ *ARPROT[0] 授权* □ *ARPROT[1] 无安全措施* □ *ARPROT[2] 指令*
m_axi_arqos[3:0]	OUT	*服务质量，QoS。为每个写入事务发送的QoS标识符。* *Quality_of_Service: 优先级*
m_axi_arvalid	OUT	*读取地址有效此信号表示信道正在发送有效的读取地址和控制信息。*
m_axi_arready	IN	*读取地址准备就绪此信号表示从机已准备好接受地址和相关控制信号。*

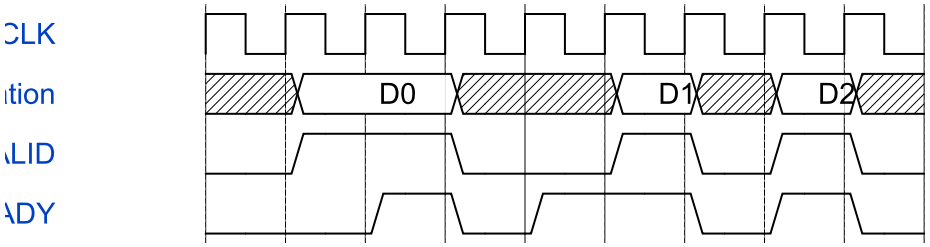
备注 这些描述摘自AMBA AXI和ACE协议规范。

主读取数据通道接口信号

信号	指令	描述
m_axi_rdata [*]	IN	*读取数据。* *[*] = [127:0] 用于* Octo *或 [255:0] 用于*
m_axi_rresp [1:0]	IN	*读取响应。此信号表示读取传输的状态。* *响应："00" = OKAY; "01" = EXOKAY; "10" = SLVERR; "11" = DECERR*
m_axi_rlast	IN	*最后读取。此信号表示读取突发中的最后一次传输。*
m_axi_rvalid	IN	*读取有效此信号表示通道正在发送所需读取数据的信号。*
m_axi_rready	OUT	*读取准备就绪。此信号表示主机可以接受读取数据和响应信息。*

备注 这些描述摘自AMBA AXI和ACE协议规范。

时序图



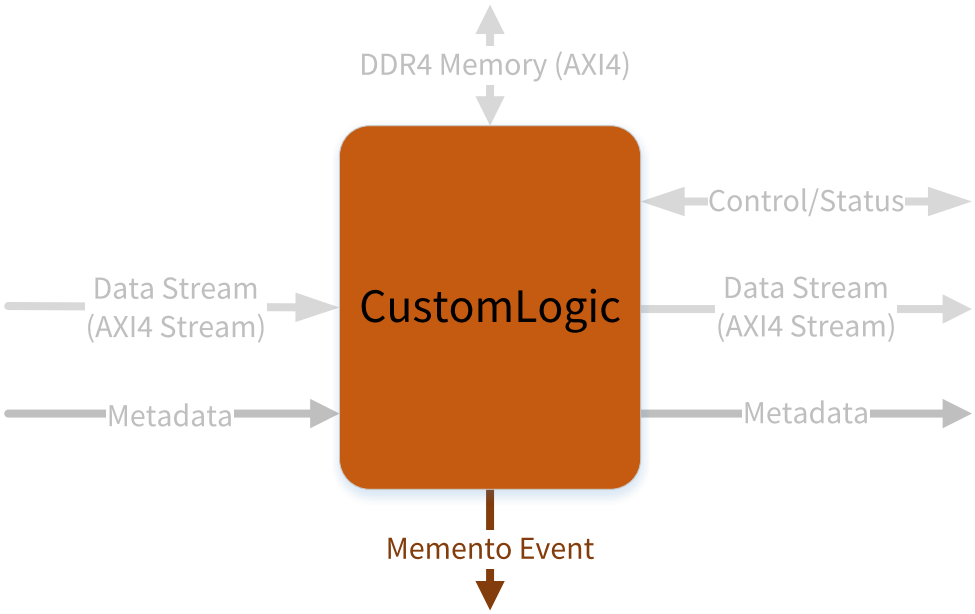
VALID/READY 信息交换时序图

2.4. Memento 事件接口

Memento事件接口允许 *CustomLogic* 向“Memento Logging”工具发送时间戳事件，其精度为1µs。

除了时间戳事件之外，在Memento中还报告了两个32位参数，如下所示：

```
ARG_0 ARG_1 [ts:0195.350699] 来自 CustomLogic 消息: 0x00000003 0x00000002
```

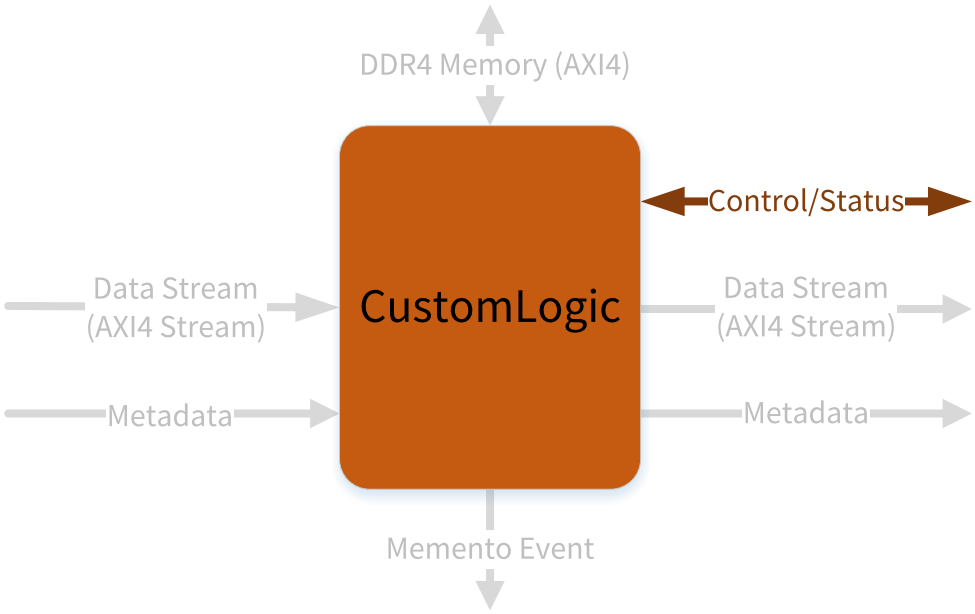


接口信号

信号	指令	描述
CustomLogic_event	OUT	一个周期的脉冲，表示一个 *CustomLogic* 事件。
CustomLogic_event_arg0 [31:0]	OUT	Memento Logging工具中报告的32位参数以及相应的 *CustomLogic* 事件。
CustomLogic_event_arg1 [31:0]	OUT	Memento Logging工具中报告的32位参数以及相应的 *CustomLogic* 事件。

2.5. 控制/状态接口

控制/状态接口允许您通过Coaxlink驱动程序API读取或写入 *CustomLogic* 内的寄存器。



此接口的使用很大程度上取决于 *CustomLogic* 如何定义控制/状态接口。推荐的定义是使用这个接口作为地址/数据控制寄存器，如参考设计文件所示 `control_registers.vhd`。

接口信号

信号	指令	描述
CustomLogic_ctrl_addr [15:0]	IN	16-位 WR/RD 地址。“WR/RD地址”选择要读取/写入的寄存器。
CustomLogic_ctrl_data_in_ce	IN	一个周期的脉冲，表明“CustomLogic_ctrl_data_in”中的一个更新。
CustomLogic_ctrl_data_in [31:0]	IN	32-位写入数据。将32-位矢量写入选定的寄存器。
CustomLogic_ctrl_data_out[31:0]	OUT	32-位读取数据。从选定的寄存器复制一个32位矢量。

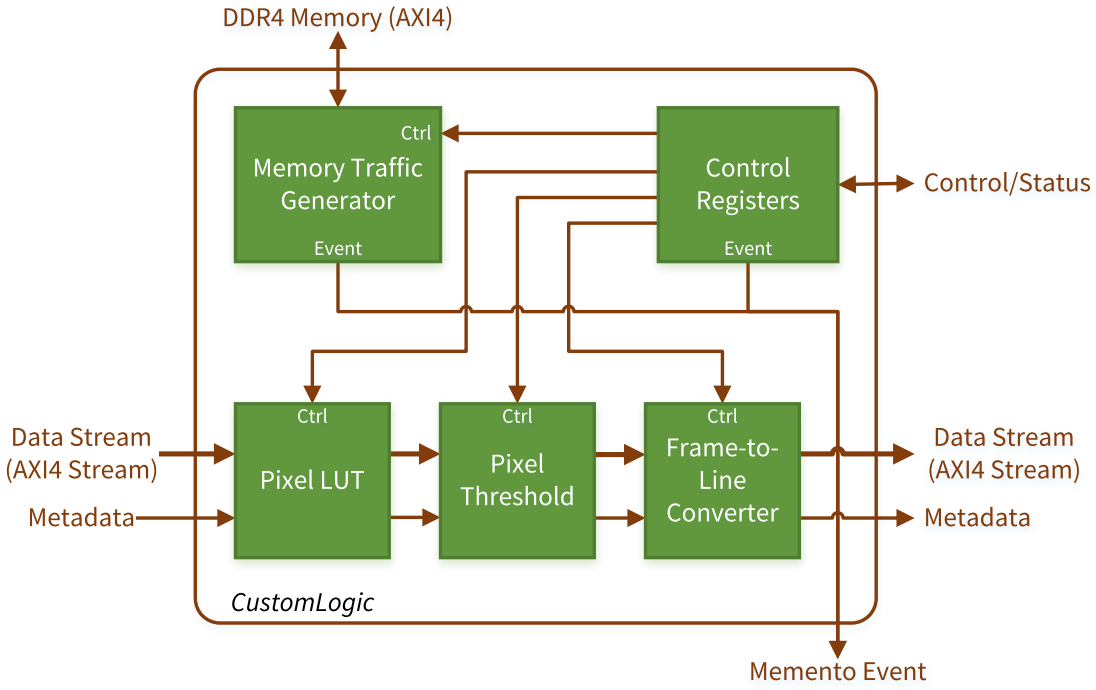
3. CustomLogic参考设计

警告 本规范为初步规范，可能会有所更改！

3.1. 简介

CustomLogic包随参考设计一起交付，以用作 *CustomLogic*的模板。参考设计公开了 *CustomLogic*中所有可用的接口。

参考设计以一组VHDL文件的形式交付，并带有以下框图：



参考设计框图

3.2. 全局信号

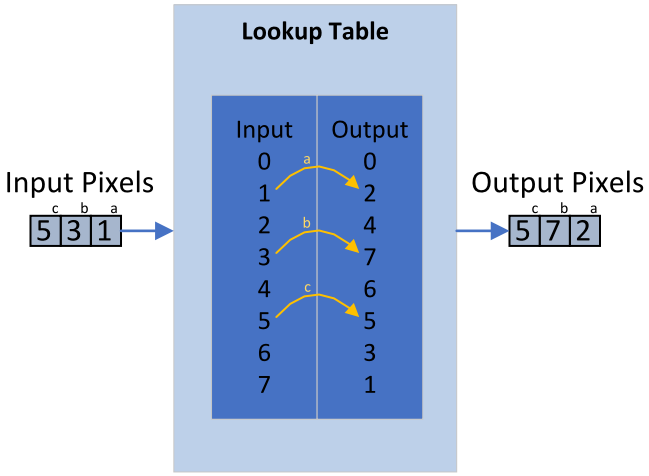
所有可用接口都在250兆赫的同一时钟域中，*CustomLogic* 全局信号如下：

信号	指令	描述
clk250	IN	250兆赫时钟源，通用于所有 *CustomLogic* 接口。
srst250	IN	在PCI Express复位期间，同步复位(clk250)置位。
PipelineClear	IN	当执行停止采集命令(DSStopAcquisition)时，脉冲置位。此信号应用于清除数据流沿线的 *CustomLogic* 内部管道。

3.3. 可用参考模块

Pixel LUT 8-位

Pixel LUT 8-位在 *CustomLogic* 参考设计渠道中提供查找表运算符。查找表格运算符可以通过其表格上的预定义值更改任何输入像素值。查找表有许多应用程序，例如，Gamma校正和对比度增强。下图说明了查找表运算符：



Pixel LUT 8位可以计算每个时钟周期16(对于 Octo) 或32 (对于) 8位像素。控制寄存器模块用于控制和上传查找表值。

像素阈值

像素阈值在 *CustomLogic* 参考设计渠道中，提供阈值运算符。对于每个输入像素，阈值运算符根据以下公式输出0或255：

```
OutputPixel= 255 when &InputPixel≥Th;
OutputPixel= 0 when &InputPixel<Th;
```

其中 `Th` 是阈值级别。

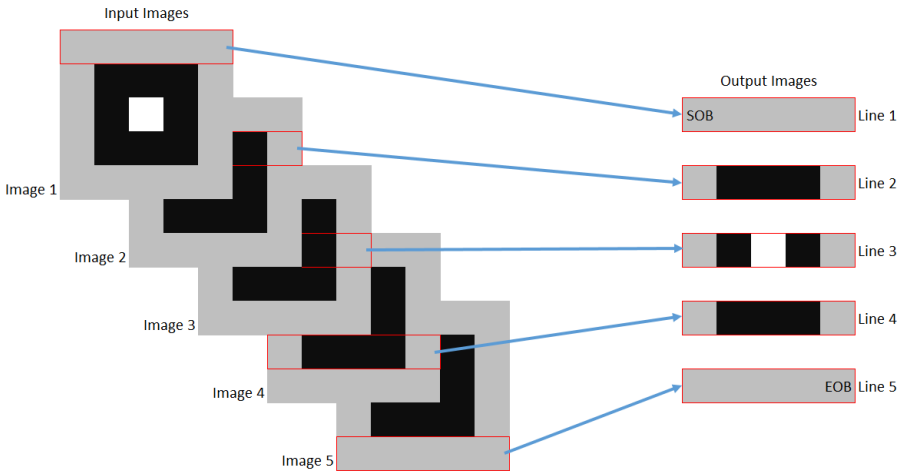
像素阈值计算每个时钟周期16(对于 Octo) 或32 (对于) 8位像素。控制寄存器模块用于控制像素阈值模块。

备注 该模块是基于使用Vivado HLS的C++代码生成的。要重新生成此模块，请按照 `/05_ref_design_hls/HLS_README.txt` 文件中描述的过程操作。

帧到行转换器

帧到行转换器，为每个输入图像输出一行。从输入图像中，按以下方式提取输出行：

- 从第一个输入图像中提取第一行。
- 从第二个输入图像中提取第二行，依此类推。
- 当“帧到行”转换器提取输入图像的最后一行(即输入图像的数量等于图像的 *ysize*)时，它在该行的最后一次传输时启用 *缓冲区结束* 标志，并开始新的采集周期。



“帧到行”转换器按顺序锁存第一个图像的输入元数据(源端)，并将其传输到输出元数据(目标端)。

该模块可通过“控制寄存器”参考设计进行控制。

“内存流量生成器”

“内存流量生成器”写入1024字节的数据突发，从0x0000000开始递增地址，并在0x 4000000(1GB)处换行。写入的数据由一个8位计数器组成。

在每个1024字节的突发之后，“内存流量生成器”回读相同地址的数据。它还会报告已发生的地址换行次数。

该模块可通过“控制寄存器”参考设计进行控制。

“Memento 事件”

参考设计中有两个“Memento 事件”来源：

- 一种是通过“控制寄存器”，在这里可以定义 *CustomLogic_event_arg0* 。
- 另一个事件是在“内存流量生成器”中发生地址换行时生成的。在这种情况下，*CustomLogic_event_arg1* 接收地址换行计数器的值。

“控制寄存器”

“控制寄存器”模块提供一种机制，通过“控制/状态”接口来控制/配置在 *CustomLogic* 中实现的模块。参考寄存器映射如下：

寄存器	地址	描述
暂存	0x0000	位 31:0(读/写) □ 32位暂存(重置值=>0x00000000)
Frame2Line	0x0001	位 0 (读/写) □ 当“0”=>“帧到行”转换器旁路被禁用。 □ 当“1”=>“帧到行”转换器旁路被启用(重置值) 。
MemTrafficGen	0x0002	位 0 (读/写) □ 当“0”=>“内存流量生成器”被禁用(重置值) 。 □ 当“1”=>“内存流量生成器”被启用时。
MementoEvent	0x0003	位 31:0(读/写) □ 此寄存器中的任何写入操作，都会生成一个Memento事件，此处定义的32位向量将复制到customlogic_event_arg0中。
PixelLut	0x0004	位0(写入，自动清除) □ 当“1”=>开始新的系数写入序列 位 4 (读) □ 当“1”=>表示系数写入序列的结束 (重置值 => ‘0’) 位 9:8(读/写) □ 当“01”=>启用Pixel LUT旁路(重置值) □ 当“10”=>禁用Pixel LUT 旁路 □ 当 其他=>没有变化
PixelLutCoeff	0x0005	位 7:0(写) □ 将系数写入Pixel LUT。每次写入该寄存器都会将系数索引从0增加到255。
PixelThreshold	0x0006	位 7:0(读/写) □ 当 0x00=>没有变化 □ 当 其他=>设置像素阈值级别(重置值=>0x01) 位 9:8(读/写) □ 当“01”=>启用像素阈值旁路(重置值) □ 当“10”=>像素阈值旁路被禁用 □ 当 其他=>没有变化

3.4. CustomLogic 交付

该 Coaxlink 针对Vivado 2018.3，并包含以下文件：

- `/README.txt`
  简要描述如何从Coaxlink *CustomLogic* 包生成Vivado项目。
- `/01_doc/D903EN-User Guide-CustomLogic-xxx.pdf`
  CustomLogic 用户指南(本文档)
- `/02_coaxlink/*.*`
  构建 *CustomLogic* 框架所需的专有文件(加密的HDL、NetList和TCL脚本)集合。
  注：这些文件不得修改。
- `/03_scripts/`
  - `customlogic_functions.tcl`
    Vivado tcl脚本包含生成.bit文件的函数，通过JTAG编程FPGA。
  - `configure_fgrabber.js`
    配置所有参考设计模块的示例脚本。
  - `create_vivado_project.tcl`
    用于为 *CustomLogic*创建Vivado项目的Vivado tcl脚本。
- `/04_ref_design/`
  - `CustomLogic.vhd`
    参考设计包装用户可以在此文件中包含自己的逻辑。
  - `CustomLogic.xdc`
    Vivado项目的目标约束文件。
  - `CustomLogicPkg.vhd`
    包含记录类型Metadata_rec定义的包。
  - `CustomLogicTop.vhd`
    项目的顶层。
  - `frame_to_line.vhd`
    “帧到行”转换器。
  - `mem_traffic_gen.vhd`
    “内存流量生成器”
  - `control_registers.vhd`
    “控制寄存器”
  - `pix_lut8b.vhd`
    Pixel LUT 8-位
  - `ip/lut_bram_8x256/lut_bram_8x256.xci`
    Xilinx IP用于Pixel LUT 8-位模块。

- pix_threshold.vhd
  像素阈值(HLS IP)。
- pix_threshold_wrp.vhd
  像素阈值包装

- /05_ref_design_hls/
  - HLS_README.txt
    简要说明如何生成HLS示例IP。
  - scripts/run_hls.tcl
    生成HLS示例IP的脚本。
  - srcs/CustomLogic.h
    *CustomLogic*全局标头。
  - srcs/pix_threshold.h
    像素阈值示例的标题。
  - srcs/pix_threshold.cpp
    像素阈值示例的C++代码。
  - srcs/pix_threshold_test.cpp
    像素阈值示例的C++测试平台代码。
- /06_release/
  - CoaxlinkOcto_1-cam.bit
    预建的FPGA比特流 ***3602 Coaxlink Octo***
  - CoaxlinkQuadCxp12_1-cam.bit
    预建的FPGA比特流 ***3603 Coaxlink Quad CXP-12***
  - CoaxlinkOcto_1-cam.ltx
    预建的Chipscope探针(空) ***3602 Coaxlink Octo***
  - CoaxlinkQuadCxp12_1-cam.ltx
    预建的Chipscope探针(空) ***3603 Coaxlink Quad CXP-12***

3.5. 参考设计构建步骤

要构建参考设计：

1. 根据Vivado要求，将包解压缩到文件夹中(路径中没有特殊字符)。例如：
   `c:/workspace/CustomLogic`
2. 启动Vivado
3. 在TCL控制台中执行脚本“create_vivado_project.tcl”。
   Tcl命令：`source c:/workspace/CustomLogic/03_scripts/create_vivado_project.tcl`
   As result, a Vivado project is created at the folder `07_vivado_project`. For example:
   `c:/workspace/CustomLogic/07_vivado_project`。
4. 运行实现。
   Tcl命令：`launch_runs impl_1`
5. 在TCL控制台中执行脚本“customlogic_functions.tcl”。
   Tcl命令：`source c:/workspace/CustomLogic/03_scripts/customlogic_functions.tcl`
   该脚本提供了以下两个功能：
   - customlogic_bitgen: *创建.bit文件*。Generate .bit file.
   - *customlogic_prog_fpga*：通过JTAG(易失性)编程FPGA。

备注 此功能需要一个XilinxJTAG程序员。

6. 此实现完成后，在TCL控制台中运行函数“customlogic_bitgen”。
   TCL命令：`customlogic_bitgen`
   This function updates the bitstream file in the folder `06_release`
7. 在生成位流之后，通过执行函数更新FPGA `customlogic_prog_fpga` in the TCL console.
   TCL command: `customlogic_prog_fpga`

备注 这一步是可选的。

4. 在调试中：

警告 本规范为初步规范，可能会有所更改！

CustomLogic 不需要任何额外的硬件来编程FPGA。

但是，要使用Vivado(Chipscope) 的调试功能，您需购买 *3613 JTAG Adapter Xilinx for Coaxlink* (1) 将 *Xilinx Platform Cable USB II programmer* (2) 连接到 Coaxlink FPGA。