Video Clipping IP Core

Video Clipping IP Core 对视频流进行裁剪. 您可以在编译时配置 IP 核, 或者在运行时使用 Avalon-MM 从接口配置它.

Video Clipping IP Core (简称 my_clipper)可以使您在视频流中选择一个激活的区域并丢弃掉剩余的数据. 您可以通过指定上下左右的偏移量来指定激活的区域.

通过读取 Avalon-ST 视频控制包, Video Clipping IP Core 可以处理分辨率变化的视频输入. 一个可选的 Avalon-MM 接口使您可以在实时运行时更改裁剪设置.

Video Clipping IP Core Parameter Settings

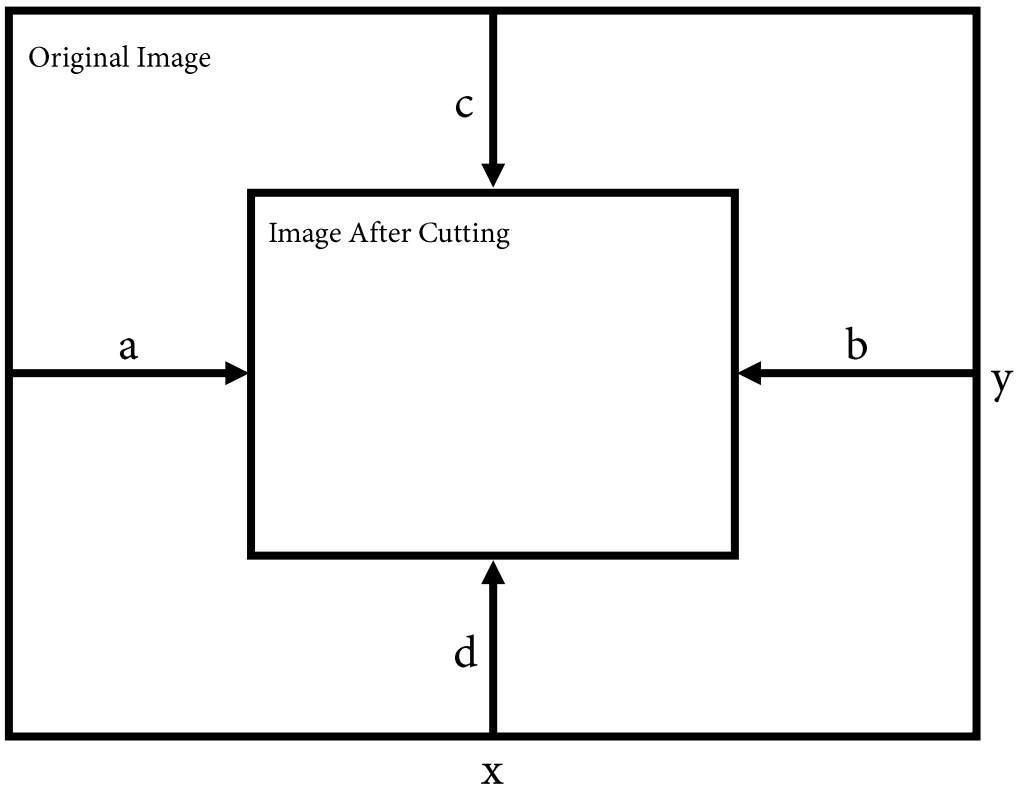
Table 1-1: my_clipper Parameter Settings

Parameter	Value	Description
Color Bits	4-32, Default = **8**	选择每个像素点 (颜色分量)的数据宽度.
Color Planes	1-3, Default = **3**	选择颜色分量的数量.
FIFO Depth	128-65536, Default = **1024**	IP Core 内部 FIFO 的大小(1)
Left Offset	Integer, Default = **24**	指定左侧的偏移量. 0 代表不对左侧进行裁剪.
Right Offset	Integer, Default = **24**	指定右侧的偏移量. 0 代表不对右侧进行裁剪.
Top Offset	Integer, Default = **0**	指定上侧的偏移量. 0 代表不对上侧进行裁剪.
Bottom Offset	Integer, Default = **0**	指定下侧的偏移量. 0 代表不对下侧进行裁剪.
Runtime Control	On or **Off**	打开这个选项来 实时修改裁剪参数.

(1) 内部 FIFO 用于对数据流进行缓存和处理, 该选项有一个推荐设置数值 D_p, 具体的计算方法稍后会详述. 通常情况下建议将该选项设置在 D_p 附近, 但如果由于资源限制等原因必须减少该选项的数值, IP Core 也可以正常工作, 但可能会由此带来输入输出数据流之间的延迟, 并且裁剪像素过多时可能会发生时序错误.

Calculation of Recommended FIFO Depth

Figure 1-1: Diagrammatic Sketch of Clipping



如图所示，原始图像的宽为 x，高为 y，左右各裁剪了 a 和 b 个像素，上下各裁剪了 c 和 d 个像素得到裁剪后的图像. 则 D_p 的计算公式为:

$$D_p = \frac{d}{y}(y - c - d)(x - a - b)$$

D_p 的值即为该条件下可能所需要的最大 FIFO 容量.

注意: 针对不同的 a, b, c 和 d 的取值, D_p 的结果也会有所不同. 如果需要在程序运行时实时改变这些参数的值，则只需将可能裁剪的最大值带入，计算出一个估计的 D_p 值作为参考值. 通常情况下，参数 d 的改变最为影响 D_p 的取值.

Video Clipping IP Core Signals

Table 1-2: Common Signals

这些信号会随着 my_clipper 的例化而生成.

Signal	Direction	Description
vst_clk	Input	视频流的主时钟.
vst_rst_n	Input	模块会在该复位信号为低时异步复位.
din_data	Input	din 端口 Avalon-ST 的 data 总线, 视频信号通过该总线传输进 IP 核.
din_ready	Output	din 端口 Avalon-ST 的 ready 信号, 当 IP 核准备好接收数据时该信号置位.
din_valid	Input	din 端口 Avalon-ST 的 valid 信号, 该信号指示此时 data 总线上的数据是否有效.
din_startofpacket	Input	din 端口 Avalon-ST 的 startofpacket 信号, 该信号标志了一个 Avalon-ST 包的开始.
din_endofpacket	Input	din 端口 Avalon-ST 的 endofpacket 信号, 该信号标志了一个 Avalon-ST 包的结束.
dout_data	Output	dout 端口 Avalon-ST 的 data 总线, IP 核通过该总线输出视频信号.
dout_ready	Input	dout 端口 Avalon-ST 的 ready 信号, 当下游的器件准备好接收数据时置位该信号.
dout_valid	Output	dout 端口 Avalon-ST 的 valid 信号, 该信号指示此时 data 总线上的数据是否有效.
dout_startofpacket	Output	dout 端口 Avalon-ST 的 startofpacket 信号, 该信号标志了一个 Avalon-ST 包的开始.
dout_endofpacket	Output	dout 端口 Avalon-ST 的 endofpacket 信号, 该信号标志了一个 Avalon-ST 包的结束.

Table 1-3: Control Signals

这些信号只会当您在 my_clipper 参数编辑器里将 **runtime control** 选项打开时出现.

Signal	Direction	Description
av_clk	Input	control 从端口的主时钟.
av_rst_n	Input	control 从端口的复位信号, 低电平有效.
av_address	Input	control 从端口 Avalon-MM 的 address 总线, 该地址指向某一寄存器, 单位为字(word)偏移.
av_read	Input	control 从端口 Avalon-MM 的 read 信号, 当您置位该信号时, control 从端口会将读数据发送到 readdata 总线上.
av_readdata	Output	control 从端口 Avalon-MM 的 readdata 总线, control 从端口通过该总线输出读数据.
av_readdatavalid	Output	control 从端口 Avalon-MM 的 readdatavalid 信号, 该信号用来表明此时 readdata 总线上的数据是否有效.
av_waitrequest	Output	control 从端口 Avalon-MM 的 waitrequest 信号, 当该信号置位时, control 从端口会忽略一切读写请求.
av_write	Input	control 从端口 Avalon-MM 的 write 信号, 当您置位该信号时, control 从端口会从 writedata 上接收新数据.
av_writedata	Input	control 从端口 Avalon-MM 的 writedata 信号, control 从端口通过该总线接收写数据.

Video Clipper IP Core Control Registers

Table 1-4: my_clipper Control Register Map

裁剪的参数会在每帧显示的开始读取到 IP 核内部缓存，因此有关的设置寄存器可以安全地在处理数据时更新.

Address	Register	Description
0	go	Bit 0 为运行寄存器, 其他位都没有使用. 设置该位为 0 会使得 my_clipper 停止工作.
1	status	Bit 0 是状态位, 其他位都没有使用. 当 my_clipper 在处理数据时会将该位置 1.
2	unused	Reserved
3	Left Offset	左侧裁剪像素数.
4	Right Offset	右侧裁剪像素数.
5	Top Offset	上侧裁剪像素数.
6	Bottom Offset	下侧裁剪像素数.

Document Revision History

Data	Version	Changes
September 2015	1.2	重写了核心代码，以修复一个 Bug，该 Bug 会导致 IP Core 来不及更新输入的图像尺寸信息.
September 2015	1.1	加入了参数 FIFO Depth，即在 IP Core 内加入了 FIFO，使得 IP Core 运行时效率更高，延迟更小.
July 2015	1.0	第一次发布