## ■ axi\_crossbar 特性描述：

**▲支持Master、Slave接口数量配置**

**▲支持全连接、部分连接**

**▲支持多threads、多issue效率匹配**

**▲支持输入BURST最大数量匹配**

**▲支持Master、Slave接口寄存，优化时序**

**▲支持Region、User、Secure配置**

## ■ axi\_crossbar整体框图：



## ■ axi\_crossbar接口使用注意：

rst\_n/rst：

接系统复位，低/高有效。

## ■ axi crossbar 配置参数说明：

DATA\_WIDTH：

默认值为32，AXI数据宽度。

ADDR\_WIDTH：

默认值为32，AXI地址宽度。

S\_ID\_WIDTH:

默认值为8，Slave port ID 宽度

M\_ID\_WIDTH:

默认值为8+clog2(8)，Master port ID宽度，多出的clog2(8)为响应路由

Sxx\_THREADS:

默认值为2， 允许同时出现的不同ID数量， 若是Master只能发出一种ID，这里设置为1，可以节省资源。

Sxx\_ACCEPT:

默认值为16， 允许缓存的数据长度，若是Master不能支持burst传输，这里的长度设置为1，可以节省资源。

Mxx\_ISSUE:

默认值为4，Masterxx 最大连续issue的请求个数，有可能连续issue多个AW/AR command。

Mxx\_BASE\_ADDR:

默认值为0，Masterxx入口起始地址。

Mxx\_ADDR\_WIDTH:

默认值为{M\_REGIONS{32’d24}}, Masterxx地址宽度。它与Mxx\_BASE\_ADDR共同决定了Masterxx的地址空间。

Mxx\_CONNECT\_READ:

默认值为{Master\_NUM{1’b1}}，其可以配置Master可以可以读到某Slave。

Mxx\_CONNECT\_WRITE:

默认值为{Master\_NUM{1’b1}}，其可以配置Master可以可以写到某Slave。

Sxx\_\*\_REG\_TYPE:

默认值0，1：允许S接口使用简单BUF，2：运行接口使用skid buffer，可以提升效率和时序。

Mxx\_\*\_REG\_TYPE:

默认值0，1：允许M接口使用简单BUF，2：运行接口使用skid buffer，可以提升效率和时序。

产生新的crossbar：

1. axi\_crossbar\_wrap.py -h 可以查看配置参数选项

如“axi\_crossbar\_wrap.py -p 2 3”,即产生2x3的全连接crossbar

1. 需要根据实际的master和slave数量，手动修改axi\_crossbar\_wrap\_top的

NB\_MASTER和NB\_SLAVE数量，以及Mx\_START\_ADDR和Mx\_ADDR\_WIDTH，其决定了AXI Slave的地址空间划分。