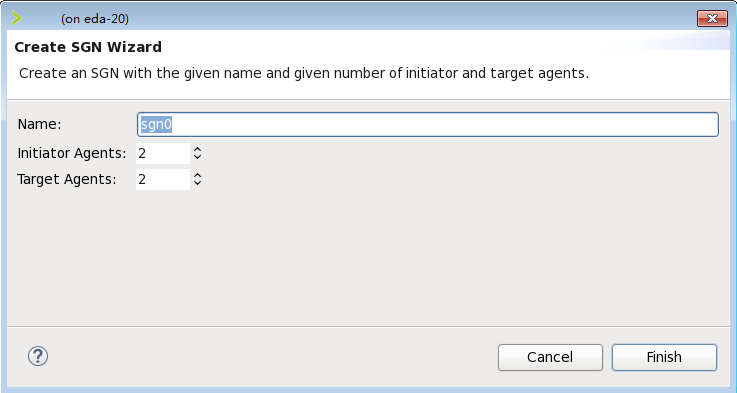
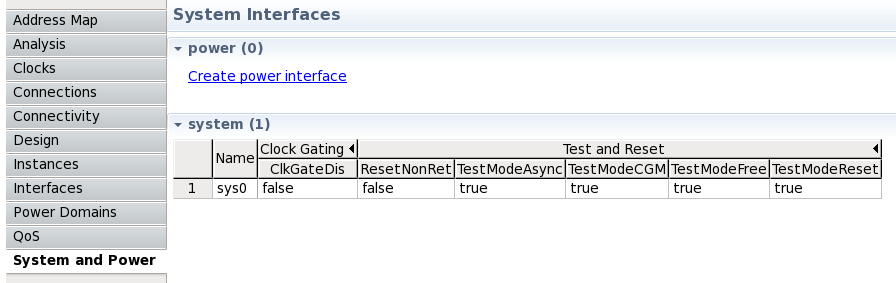
**NOC Review**

# 一、NOC 配置方法

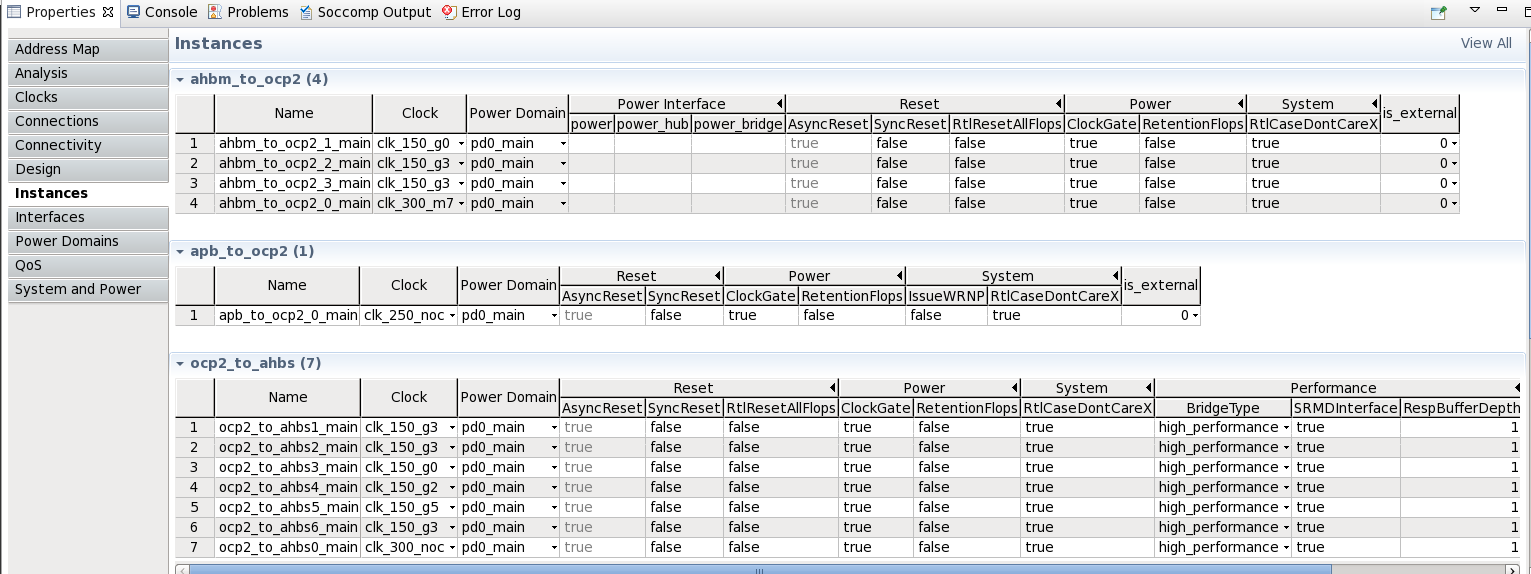
1. 创建SonicsGN，定义Initiator和Target的个数。



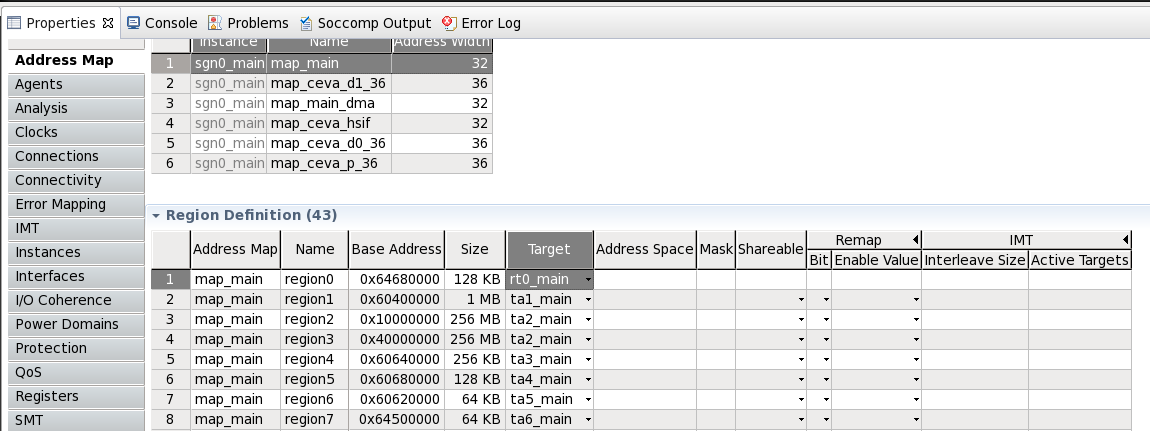
1. 将System and Power选项中的Clock Gating的ClkGateDis设为false

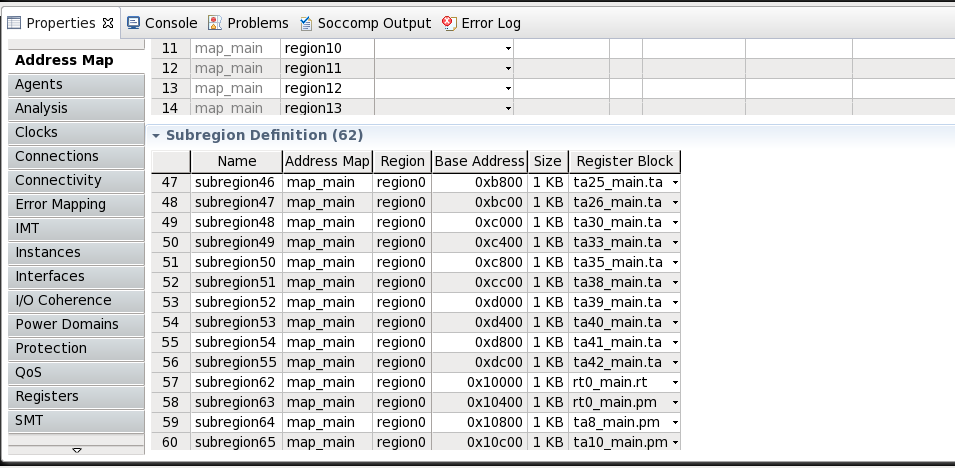


1. 为每个Initiator和Target选择相应的Instance，AXI、AHB、APB，以及对应的Clock和Power Domain。同时对应的Interfaces里面配置相应端口协议的参数，如Addr、Data等位宽。

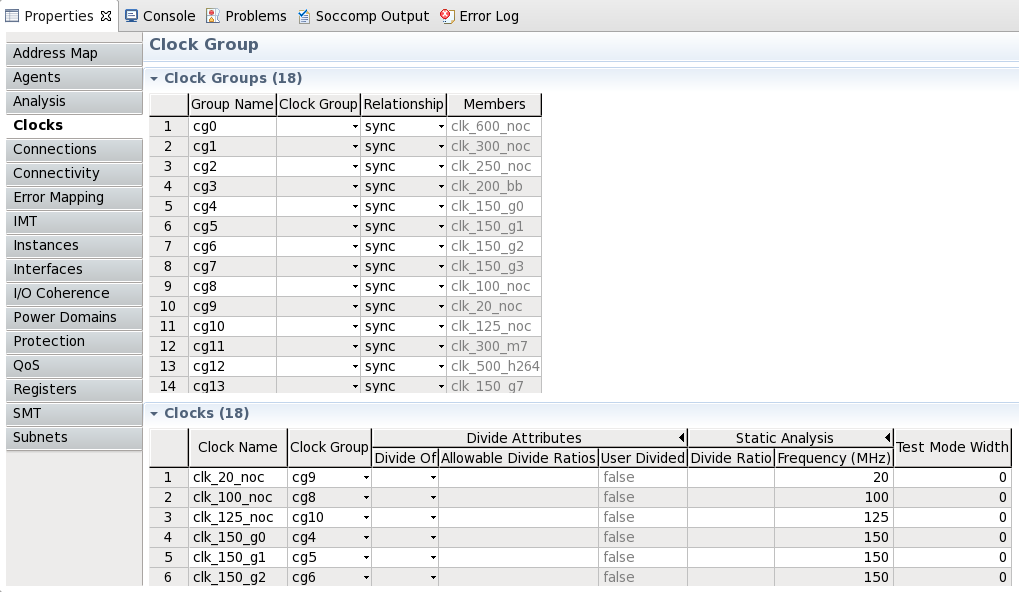


1. 在Address Map中，创建地址的表单，可为不同Initiator端口配置不同的表单，其中rt0\_main为NOC本身的寄存器。在对应的Subregion Definition里面加入需要以后可以访问到的寄存器，如ia0\_main.ia、ta0\_main.ta、rt0\_main.rt、rt0\_main.pm、ta0\_main.pm等。

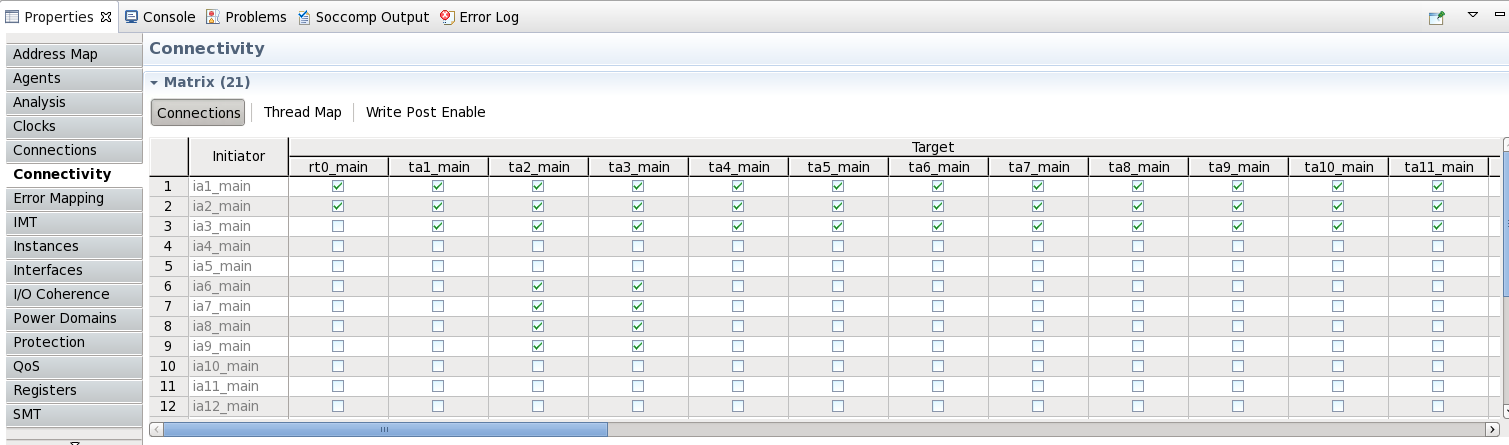




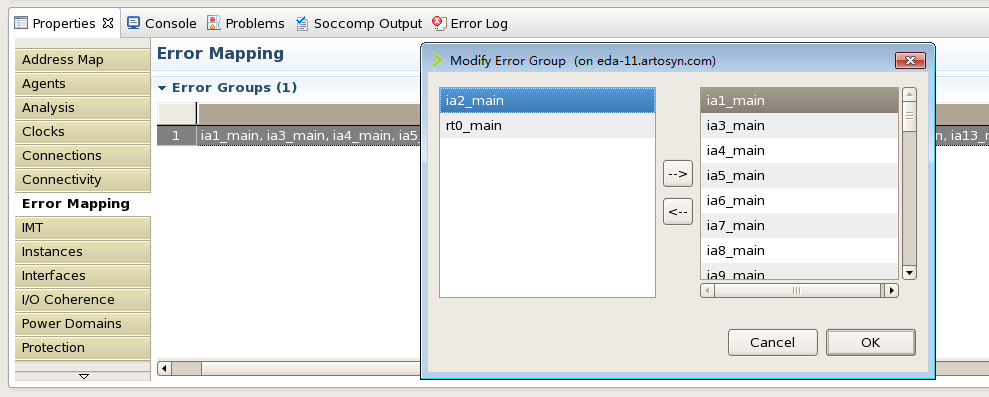
1. 在Clocks创建需要的始终及相应的频率和关系。



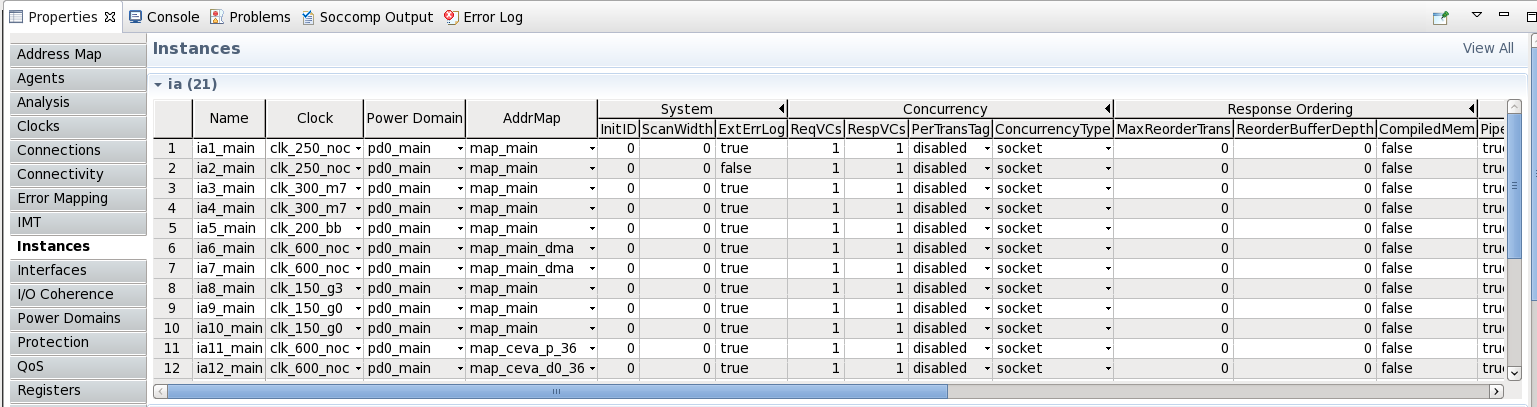
1. 在Connectivity中，确定每个Initiator可以访问Target的权限关系。



1. 在Error Mapping中，确定需要检测Error产生的端口（如访问了受保守保护的Target，发出的请求格式Target不支持等）的表单，rt0\_main以及Initiator为APB协议的不可以加入该列表中。

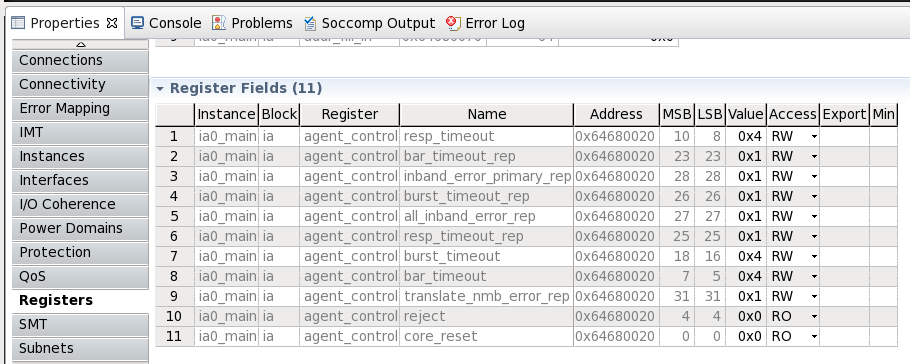


如果需要将相应的Initiator和Target可以被加入该表单，需要在Instance里面将相应的Initiator和Target的System中的ExtErrlog设为true。

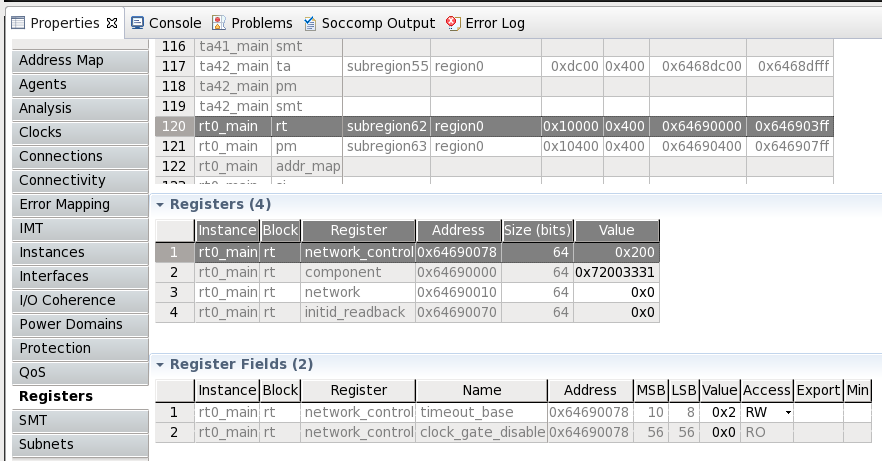


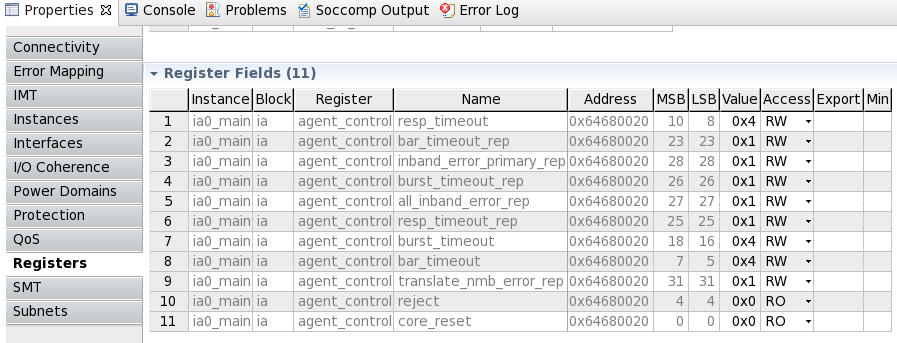
主要可以分为两种Error，sinterrupt signal用于timeout的错误，serror sginal用于除了timeout的其余错误，这两个信号为中断信号。首先需要使能rt0\_main.rt。

对于除了timeout的Error，需要设置AGENT\_CONTROL中的\*\_REP将需要报告的Error类型设为1。



对于timeout的Error，需要将rt0\_main.rt 中的network\_control.timeout\_base设为非0值；将ia/ta.agent\_control.xxx\_timeout设为非0值；将ia/ta.agent\_control.xxx\_timeout\_rep设为1。





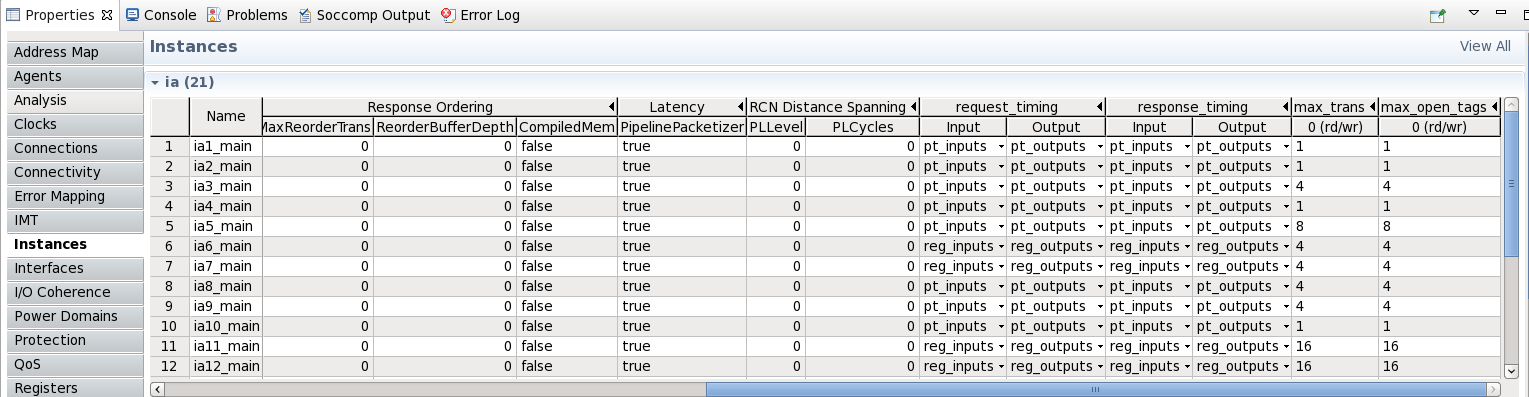
1. 在Instance中，主要确定

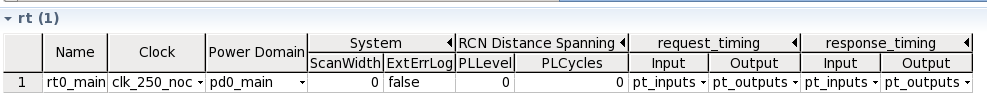
Initiator的Clock、Power Domain、AddrMap、request\_timing、response\_timing、max\_trans以及max\_open\_tags，其中max\_trans和max\_open\_tags决定该instance可以缓存的请求数目以及不同的ID个数；

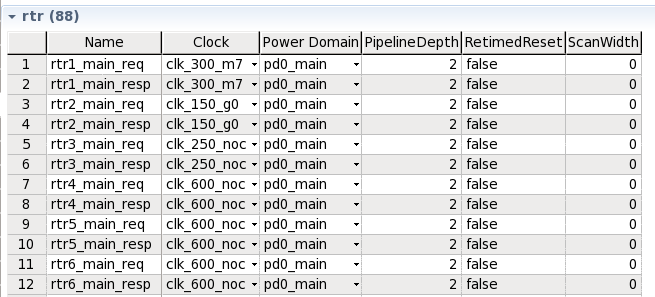
rtr确定Clock、Power Domain以及PipelineDepth，其中PipelineDepth决定rtr的可综合频率；

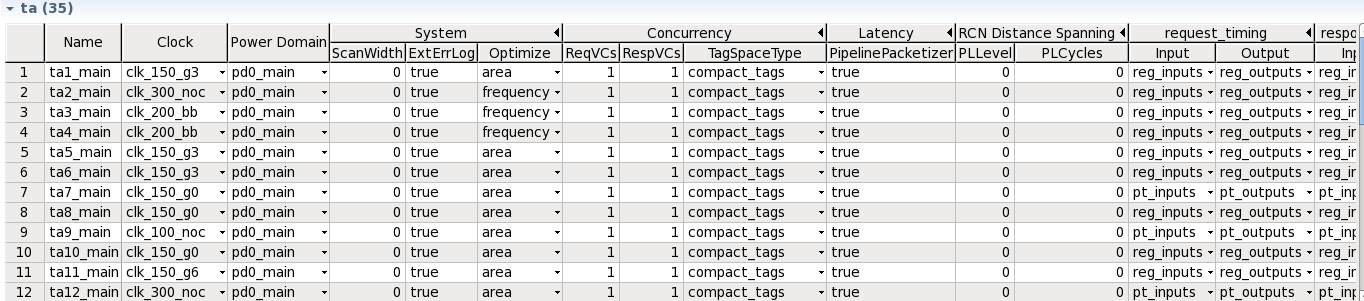
rt主要确定rt0\_main的Clock以及Power Domain；

Target与Initiator类似，其中System中的Optimize选择该Target优化的方向，有area和frequency两项；Concurrency中的TagSpaceType可选compact\_tags、compact\_tags\_with\_initid、pass\_through\_tags以及pass\_through\_tags\_with\_initid，一般选compact\_tags和pass\_through\_tags。

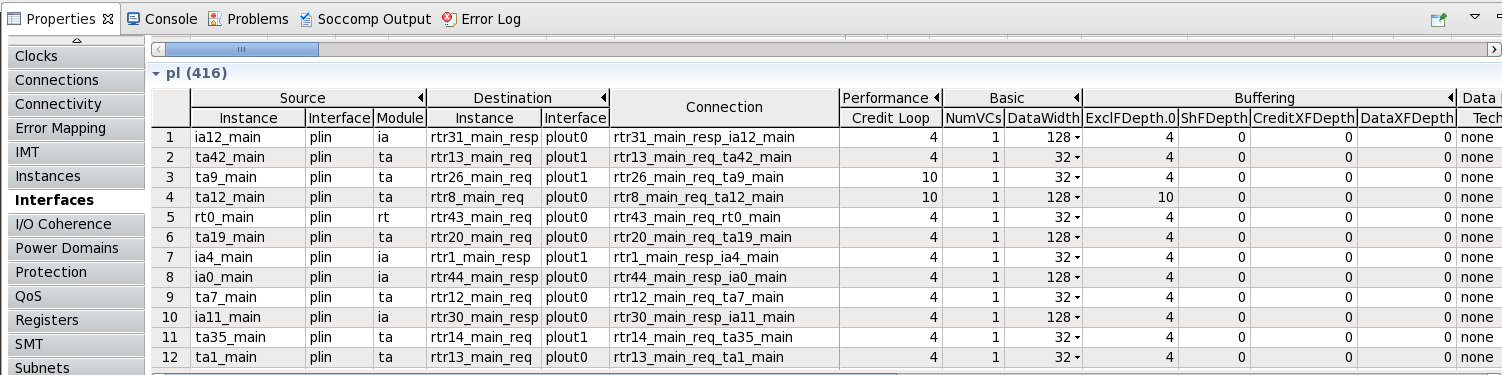


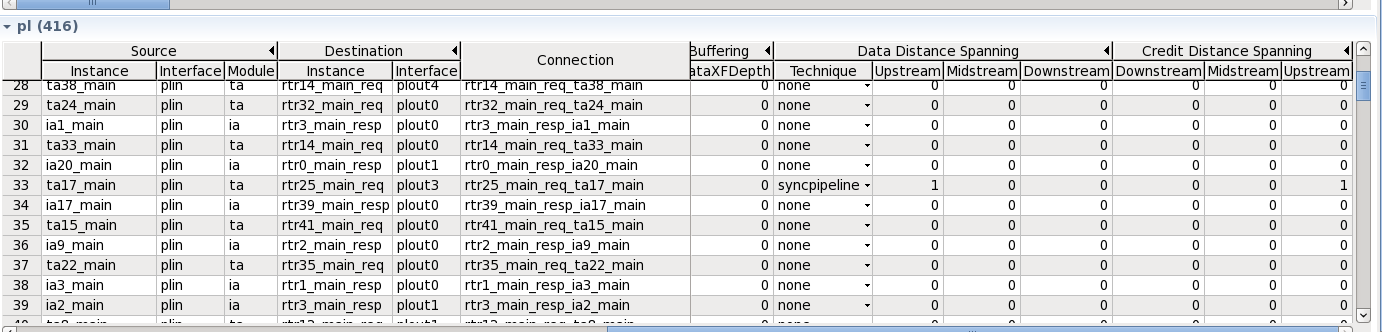




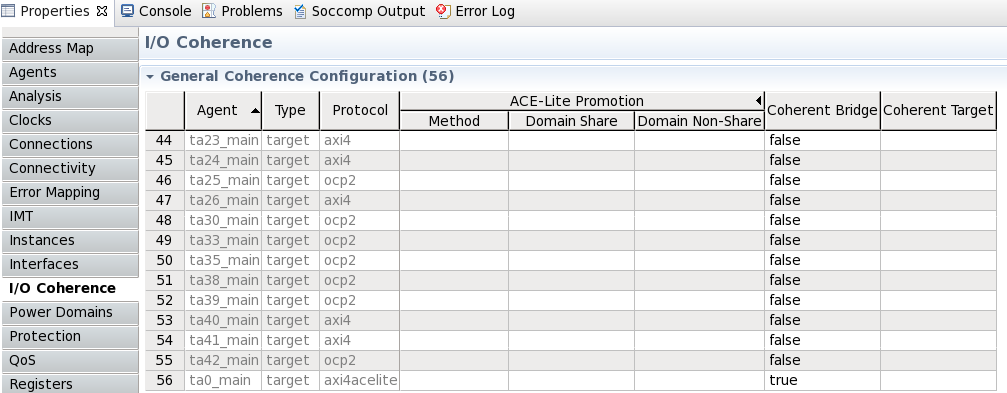


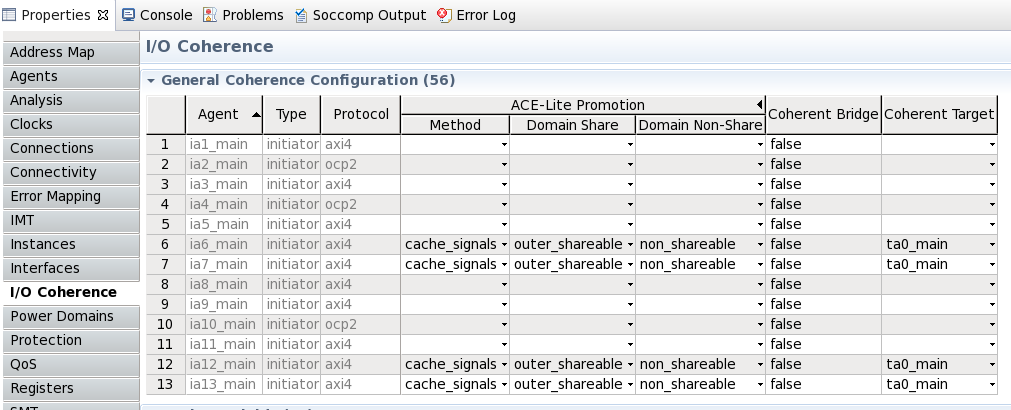
1. 在Interface中，主要设置rtr之间连线PL的Buffering中的ExclFDepth.0的值，一般与Performance中的Credit Loop的值相同，主要是为了确保该PL可以缓存的请求数可以达到理论值。同时，有可能用到Data Distance Spanning，主要功能是将该PL多加几级寄存器，在物理实现上更好实现，一般Data Distance Spanning中Technique选syncpipeline，为同步加寄存器。



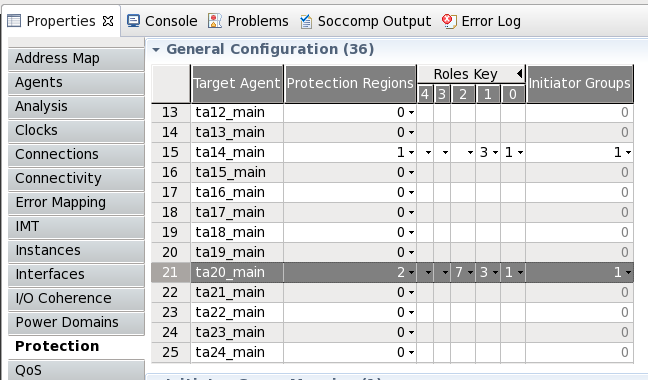


1. 在I/O Coherence中，主要配置与CCI相关的端口，连接到CCI Slave口的Target的Coherent Bridge设为true，同时该Target必须为axi4acelite协议；需要往该Target发coherence请求的Initiator如果为axi4的可以promote到axi4acelite，主要通过cache信号进行判决，cache[1]==1，同时cache[2]==1或cache[3]==1。





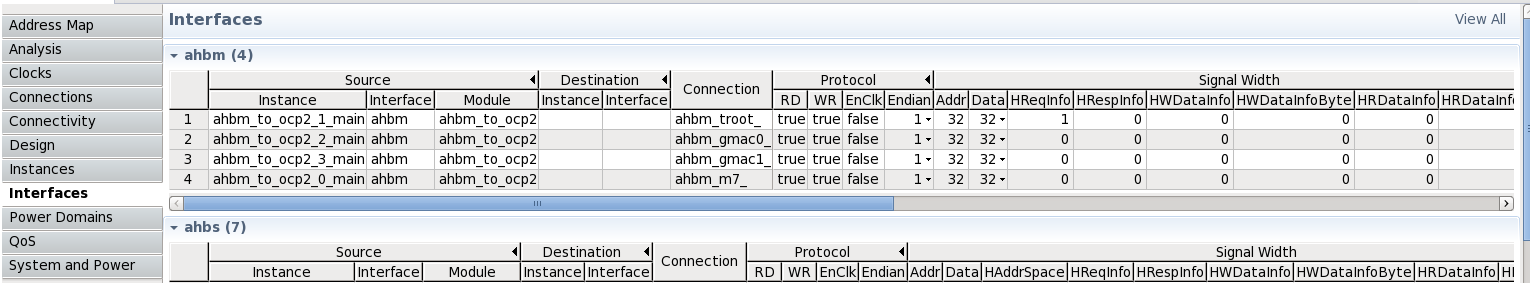
1. 在Protection中，主要确定需要NOC保护的Target的保护规则。

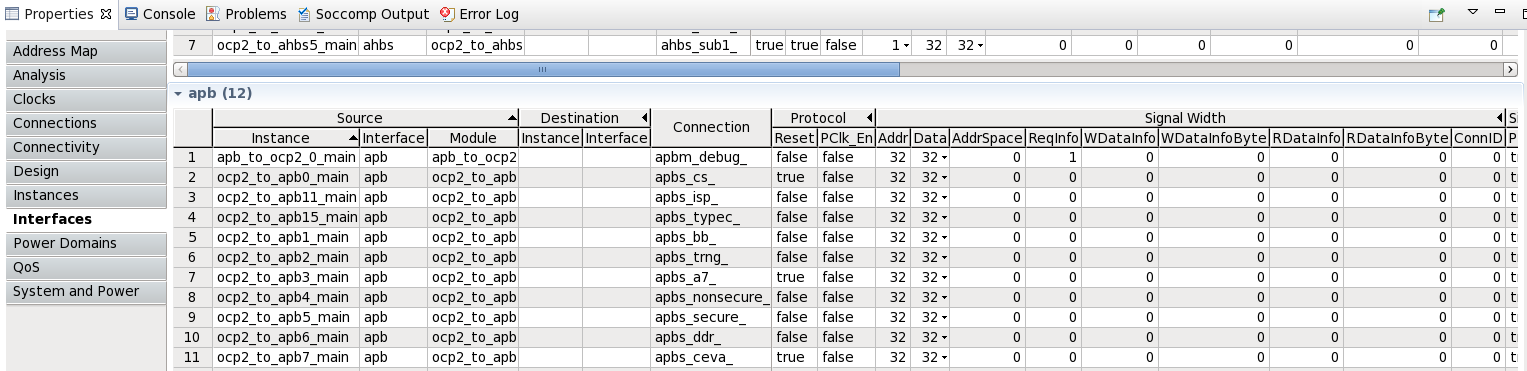


Protection Regions定义该Target的空间分为几个区域，每个区域可以有自己单独的保护规则；

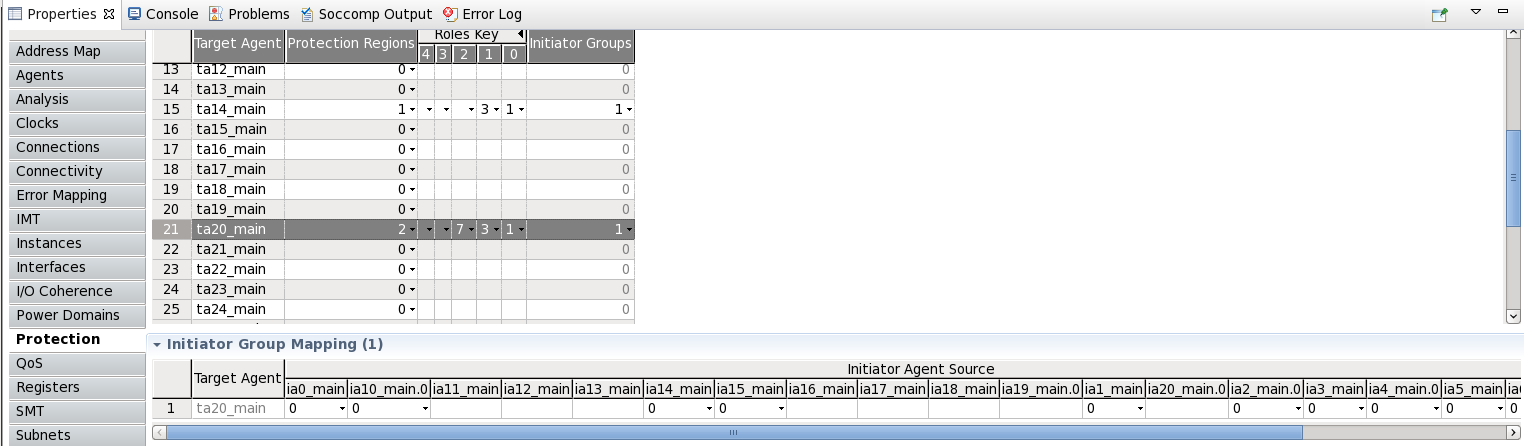
Roles Key定义保护规则，主要由5bits组成。其中Roles Key每个比特的选择信号来源是，[2:0]=prot[2:0]，[x:3]=internal reqinfo[x-3:0]。

AXI主要使用prot[1]区分unsecure和secure，AXI的user信号对应internal reqinfo信号；由于AHB和APB没有prot信号或者prot信号不代表secure和unsecure的区别，需要在AHB和APB的端口生成reqinfo信号；需要注意的是，internal reqinfo[0] = apb\_mreqinfo[0]， internal reqinfo[4:0] = {ahbm\_hreqinfo[0]，ahbm\_hprot\_i[3:1]，!ahbm\_hprot\_i[0]}。



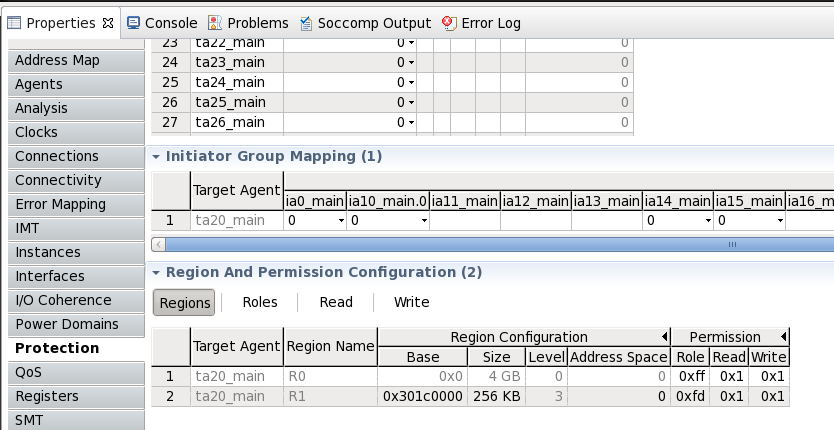


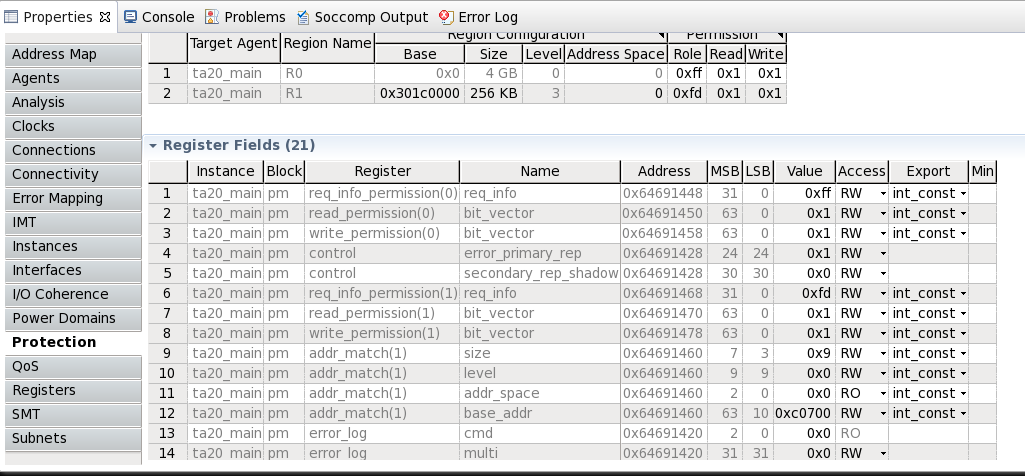
Initiator Groups定义多个group，各个group之间可以包含不同的Initiator，相同的group中的Initiator具有相同的访问规则，不同的group中的Initiator具有不同的访问规则；



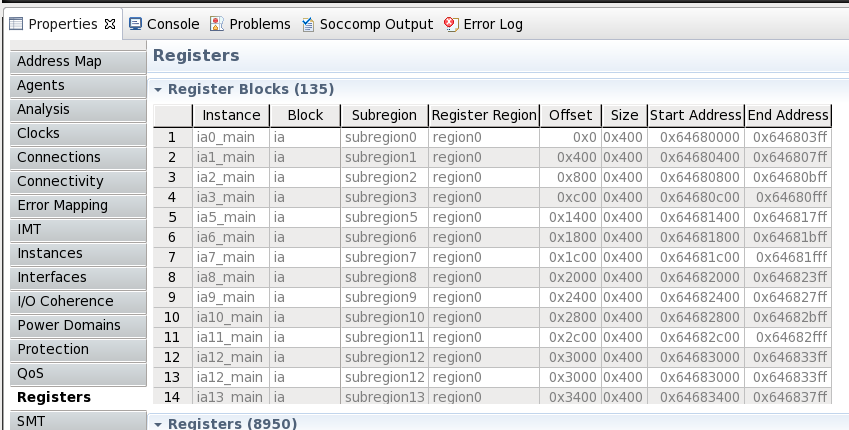
如果一个Target的空间区分为几个保护空间，每个空间的保护规则可以不一样，主要通过Permission中的Role来确定，Role的值由Roles Key[4:0]组成的每个值5’b00000到5’b11111，每个值对应的含义允不允许访问确认，5’b00000对应Role[0]，5’b00001对应Role[1]直到5’b11111对应Role[31]。一般改变保护规则，主要通过改变Permission中的Role值。

Permission中的Read和Write设为1，代表对应的操作需要经过Role规则，如果设为0，代表不允许读或者写。

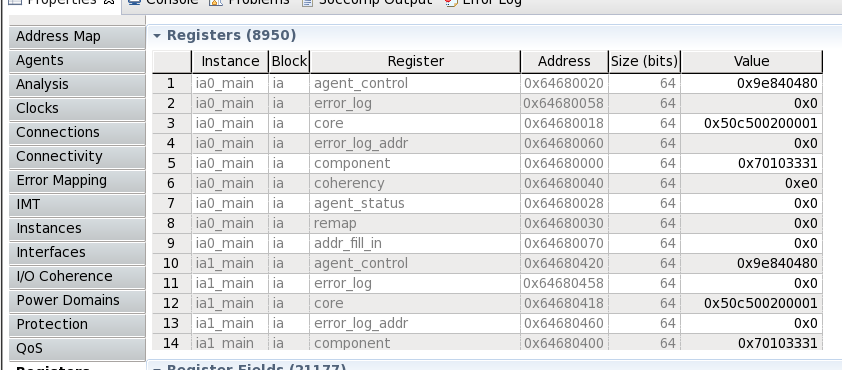




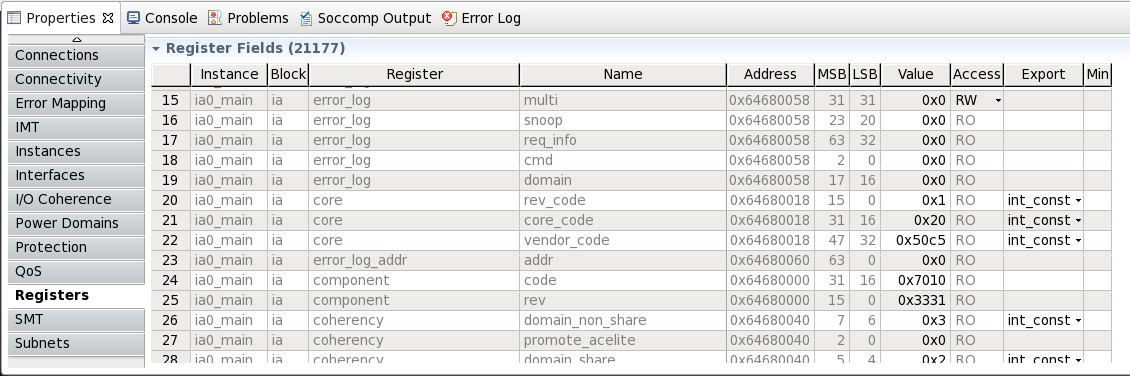
1. 在Register中，主要定义NOC相关寄存器的默认值，如ia0\_main.ia、ta0\_main.ta、ta20\_main.pm、rt0\_main.rt以及rt0\_main.pm等，同时可以看到各种寄存器有没有加到Address Map的Subregion Definition中，可不可以以后被系统访问到。



Registers中的Registers可以看到相应Instance相应Register的地址、大小以及值。



Registers中的Register Fields可以看到相应Register的地址、大小以及值，可以在这里更改相应的寄存器的值。



1. Sirius中用的NOC Director的版本是8.1.1。

# 二、NOC仿真环境

在徐卓的scompile环境下可以进行NOC的相关仿真。不需要配置额外的参数。

# 三、NOC相关的SOC Flow环境

1. DC

主要就是dc的约束文件用Sonics产生的。

NOC\_main：

NOC\_vision: 

NOC\_ceva：



1. PT

NOC\_main：

NOC\_vision: 

NOC\_ceva：

1. Formal Check

NOC\_main：

NOC\_vision: 

NOC\_ceva：

