### C++ 参考模型(Reference Model) 的API

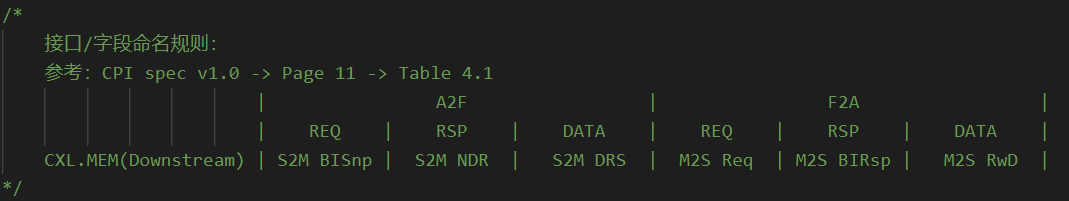
#### C++ Reference Model中有哪些接口/字段

* DUT（功能上是CXL Type3 HDM-DB设备）中接口/字段
* C++ Reference Model 中接口/字段

后者是前者的一个子集。

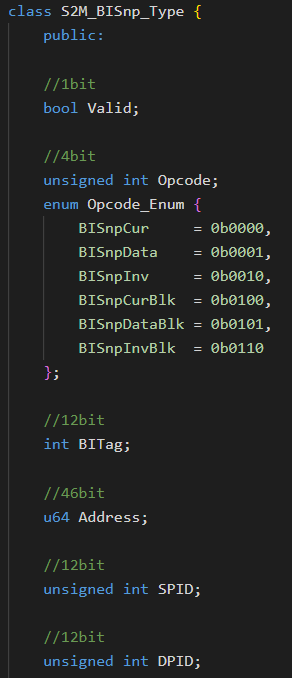
##### DUT中接口/字段

参阅CPI spec v1.0，并定位到CXL.mem(Downstream): 总共6个通道，每个通道有若干接口/字段。

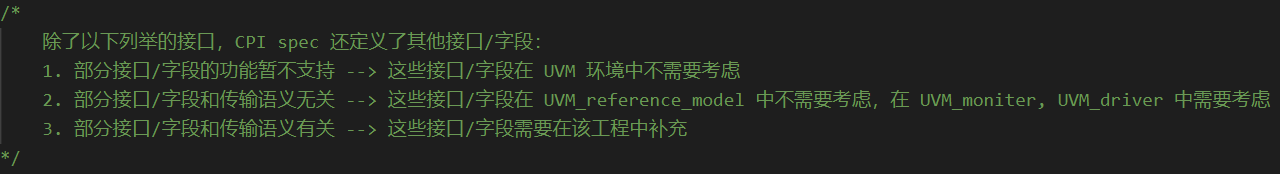


##### C++ Reference Model 中接口/字段

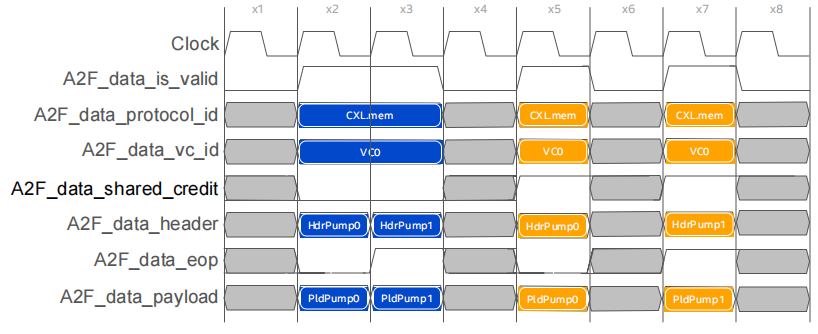
举例其中1个通道S2M\_BISnp，其接口/字段在C++中以class成员的方式被罗列，C++上该通道的接口/字段少于CPI spec中定义的接口/字段：



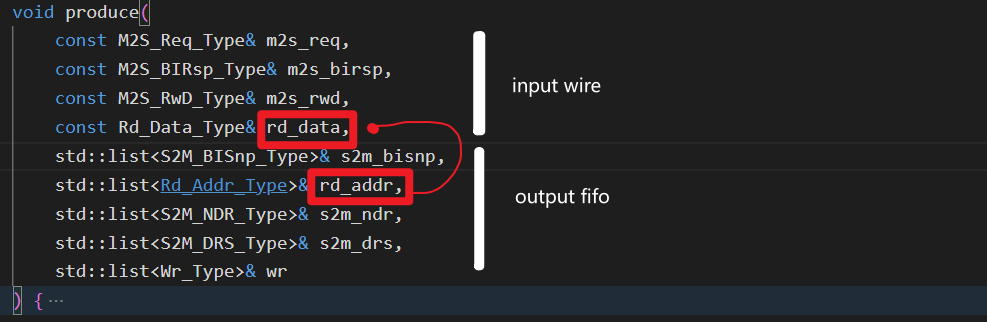
为什么暂无：



关于第2点举例：比如一个64B的Cacheline\_Data的多周期传输，这种时序可以在UVM\_moniter中考虑（UVM\_moniter在其全部64B发送后才产生一个Transaction给软件Reference Model）。



#### C++ Reference Model中这些接口/字段的 软件处理



* 一次 valid + data 的输入，此时软件处理的结果可能要分多次输出。

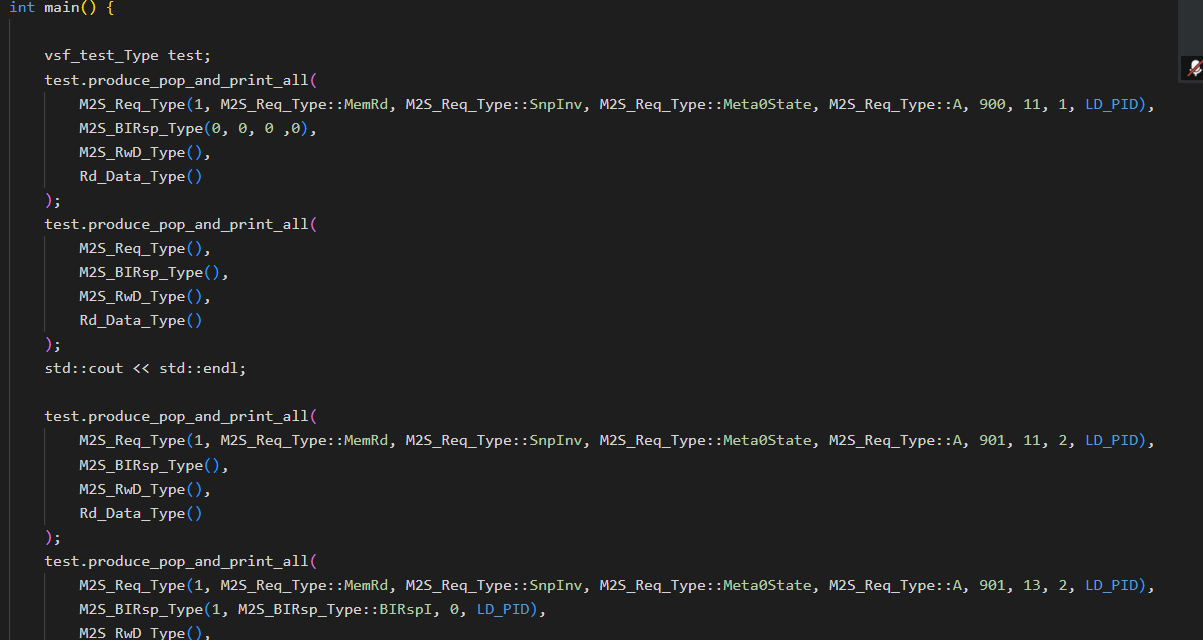
例如：一次SnpInv输入，Device DCOH后可能要BISnpInv给多个主机，但是只有一个BISnp通道，所以要分多次输出。

* 多次 valid + data 的输入，此时软件处理才会产生一次输出。

软件侧是采用一个produce函数来处理，函数的输出是有深度的fifo(ram)/list(vector)，用以支持一次valid + data的输入，多次输出的情况。produce函数输入就是一个wire/参数。

目前版本中produce函数中擅自使用有很多不在协议定义中的“处理策略”，此版极大概率有误！

#### 1.3 C++ Reference Model的testcase



后续改成Array/File形式的批量输入。

### 疑问

##### 2.1

* 对于固定输入序列（间隔时间不同但次序固定），不会有不同的序列输出。

目前C++代码采用这种。

对于固定输入序列（间隔时间不同但次序固定），可能有不同的序列输出（但是只是序列的乱序）。

这种也可以修改少部分代码以支持这种。

* 对于固定输入+输出序列（间隔时间不同但次序固定），可能有不同的序列输出（但是只是序列的乱序）。输入固定，输出序列不固定，会导致后续输出序列不一致。

##### 2.2

* HDM-Decoder？（采用CXL.io配置？）【目前HPA=DPA】
* 动态容量策略？（采用CXL.io配置？会影响到SF和DCOH吗？后续要写reference model吗？）后续加几个端口和HPA，DPA
* SF\_Alloc\_Victim策略？【目前采用主机向从机发送SnpInv + MemInv + I】
* BITag分配策略？【目前采用0->0xFFF递增循环】

#### 这个项目还需要

字段都加上去

* SVN版本控制
* 多文件
* 多几个测试用例(用const Array/File组织)
* 批注，Readme
* Linux上跑GCC环境
* U~~VM上跑（用C++ tlm还是C++ DPIC接口还是用sv）~~
* 主机激励（行为模型）

#### 四、UVM中Reference Model的位置

UVM架构，reference\_model在其中的位置

try\_get

Moniter1 ---> tlm-fifo(in\_transaction) ---> reference\_model --->Tlm-fifo(out\_transaction)-> scoreboard

Moniter2 ---> tlm-fifo(out\_transaction) ---> reference\_model

Subscriber