### An Introduction to the CXL Reference Model

#### 一、运行环境

* GCC12.2.0及以上版本
* Make3.81及以上版本

**环境配置：**需将gcc和make添加至环境变量

**支持操作系统：**Linux、Windows

#### 调试环境、代码编写环境

**开发工具：**Visual Studio Code（VS Code）

**支持操作系统：**仅Windows

**VSCode插件：**

**C/C++（必选）：**提供C++语法检查；支持GUI调试（基于GDB）。

**CodeGeeX:AI Coding Assistant（可选）：**AI代码补全。

**GitLens（可选）：**增强Git版本控制功能，支持本地及GitHub远程仓库管理。

#### 三、代码风格

本项目遵循以下代码风格规范，以保持一致性、可读性和可维护性：

##### 3.1文件规范

工程的文件组织如下：

│── .vscode/ #配置GUI调试环境（和Reference Model的功能无关）

│── doc/ #文档目录

│── include/ #头文件目录（存放.h文件）

│── out/ #编译输出目录（存放.o文件）

│── pic/ #图片目录

│── src/ #源代码目录（存放.cpp文件）

│── makefile #Makefile编译脚本，自动化.cpp --> .o --> .exe的编译过程（和Reference Model的功能无关）

要着重关注的是include/下的.h文件和src/下的.cpp文件：

**头文件（.h文件）包含如下：**

* **#ifndef，#define，#endif**用以避免.h文件的重复包含。
* class的声明（该class{}内**有且仅有**：**成员变量的声明，非内敛成员函数的声明，内敛成员函数的声明和定义，enum class的声明和定义**）。

Notes class的声明和定义具体见3.4

**源文件（.cpp文件）包含如下：**

* class中**非内敛成员函数的定义**。

除了两个文件define.h和main.cpp，.h和.cpp文件命名保持一致，并全部小写。

例：s\_vsf\_line.h和s\_vsf\_line.cpp。

##### 3.2命名规范

###### 3.2.1 class命名规范

单词间使用\_分割，首字母大写，末尾加一个\_Type。

**例：**class S\_Test\_Type{};

###### 3.2.2成员变量，成员函数命名规范

单词间使用\_分割，全部小写。

**成员变量例：**u32 host\_pid;

**成员函数例：**bool is\_idle();

###### 3.2.3 enum class命名规范

单词间使用\_分割，首字母大写，末尾加一个\_Enum。

**例：**enum class Opcode\_Enum{};

##### 3.3面向对象特性

在C++工程的任何部分，应避免使用复杂的面向对象特性和语法，因为项目需求并不依赖这些特性。具体编码规范如下：

* **不使用全局变量**，所有数据必须封装在class内，确保数据的可控性和封装性。
* **不使用继承和多态**，所有class均为独立实现，避免不必要的层级关系。
* **不使用friend**关键字。
* **不使用protected**访问权限，仅使用private、public访问权限。
* **不定义**新的数据结构，如**struct、typedef**，仅使用enum来定义枚举类型。
* **不使用**动态内存分配（**new、delete**），所有对象应为自动管理的栈对象或使用标准容器（如std::vector、std::list）。

##### 3.4 class声明和定义

###### 3.4.1类成员（包括成员变量、成员函数）类型限定

* 使用**基本数据类型**
* bool //具有（1bit）
* u32 //具有（32bit），即是unsigned int
* u64 //具有（64bit），即是unsigned long long
* u512
* 使用**标准库字符串**，即：
* std::string //项目工程内不使用using namespace std;
* 使用**标准库容器**，即：
* std::list
* std::vector
* std::unordered\_map
* 使用**本项目工程内声明的其他class**

###### 3.4.2类成员函数的参数约定

**输入参数：**统一使用const Type &。（&用以避免拷贝开销）

**输出参数：**统一使用Type &。

Notes 这里的“Type”是一个举例的类型名，不是某个具体类型名

**返回值类型：**void或者上述3.4.1所述的基本数据类型之一。

###### 3.4.3 类构造函数和析构函数

**必须实现类的构造函数。**用Type(T a) : a(0)。类型为**基本数据类型**的成员变量要赋初值0；类型为**标准库字符串，标准库容器，本项目工程内声明的其他class**的成员变量不初始化。

Notes 这里的“Type”“T”是一个举例的类型名，不是某个具体类型名。

Notes 这里的“a”是一个举例的成员变量，不是某个具体的成员变量。

**不要实现类的析构函数。**

##### 3.5标准库容器实现

* 项目工程中不定义带有push字段的方法。
* 标准库容器对象的元素要被增加时，其size要被检查，当对于一个std::list要push或者一个std::unordered\_map以[]作为左值时，必须检查size，防止栈/堆内存溢出。

标准库容器对象的元素要被删除时，其size要被检查：要先判断非空

**通过遍历以查询元素、修改元素时**：for(auto&x:list)，避免拷贝开销。

**通过遍历以删除元素时**：使用迭代器遍历for(auto it=list.begin();it!=list.end();)。

项目工程不通过迭代器增加元素

##### 3.6多输入多输出函数标注释

  std::list<S2M\_Snp\_type>s2m\_snp\_dpa;

  std::list<S2M\_NDR\_Type>s2m\_ndr\_dpa;

  std::list<S2M\_DRS\_Type>s2m\_drs\_dpa;

  main\_product(m2s\_rsp\_dpa,m2s\_rwd\_dpa,s2m\_snp\_dpa,s2m\_ndr\_dpa,s2m\_drs\_dpa,rd\_data,rd\_addr,wr);

之后，要写出s2m\_snp\_dpa,s2m\_ndr\_dpa,s2m\_drs\_dpa=main\_product(m2s\_rsp\_dpa,m2s\_rwd\_dpa)