Vitis HLS（原Vivado HLS）是一个高级综合工具，即可以用C/C++高级语言也可以用verilog等硬件语言来实现硬件逻辑。

根据Vivado HLS的使用指南，需要对输入程序作出以下规范：

\* **不使用动态内存分配**（malloc, free, new, delete）

\* 减少使用指针对指针的操作

\* **不使用系统调用**（如abort, exit, printf），可以在测试平台上使用，但综合时这些指令会被无视

\* 减少使用其他标准库里的内容（支持math.h中常用内容）

\* 减少使用C++中的函数指针和虚拟函数

**\* 不使用递归方程**

\* 精准表达交互接口

任意精度数据

C：ap\_cint.h——[u]int<W> (1024 bits)

C++：ap\_int.h——ap\_[u]int<W> (1024 bits，可扩展为32K位宽)

C++：ap\_fixed.h——ap\_[u]fixed<W,I,Q,O,N>

使用sizeof()时：对其到1、2、4等字节

Include：Xilinx\Vivado\_HLS(Vitis\_HLS)\版本号\include

声明方式

ap\_int<6> a\_6bit\_var = -22;

ap\_int<6> a\_6bit\_var(-22);

ap\_int<6> a\_6bit\_var{-22}; 2019不支持

ap\_int<6> a\_6bit\_var(“0b101010”, 2);

ap\_int<6> a\_6bit\_var(“101010”, 2);

ap\_int<6> a\_6bit\_var(“-22”, 10);

ap\_fixed<W,I,Q,O>

W数据总字长

I整数部分字长

Q量化模式（针对低位），默认AP\_TRN\_ZERO（舍去低位），可配置AP\_RND（四舍五入）

O溢出模式（针对高位），默认AP\_WARP（舍去高位），可配置AP\_SAT（饱和，所有非符号位填充1）

ap\_fixed<3,2> var1 = 1.25; 小数部分字长1，丧失精度=1

ap\_fixed<3,2,AP\_RND> var2 = 1.25; AP\_RND量化模式，最终=1.5

ap\_fixed<4,4> var3 = 19; 0b010011，取低四位0x0011=3

ap\_fixed<4,4,AP\_RND,AP\_SAT> var4 = 19; 饱和溢出模式，填充为0x0111=7

短数据+长数据，扩展短数据

有符号+无符号，扩展符号位

有符号短变量赋值长数据（ap\_int<2>=0x011 = 0x11 = -1）错误

强制类型转换(ap\_ufixed<6,4>)i4

常用运算 ap\_int和ap\_uint

大数据不溢出，小数据不损失

相加：

同类型相加，位宽为最大位宽+1

不同类型相加，如果无符号较大，则最大位宽+2，否则最大位宽+1

相乘：

同类型相乘，位宽相加

相除：

有符号相除，被除数位宽+1

无符号相除，结果位宽为被除数位宽

取模：

同类型取模，结果位宽为最小位宽

获取数据类型信息

#include <typeinfo>

typeid(var).name()

hls\_math.h库支持了C的math.h和C++的cmath.h，包括数据类型和方法

复合数据类型

支持结构体和枚举类型

结构体

Directive:HLS DATA\_PACK variable=i\_val field\_level

field\_level：结构体所有成员位宽分别对齐到1字节

struct\_level：保留每个成员实际位宽，但封装后总位宽要对齐到1字节

枚举

默认从0起，后面依次加1

HLS自动分配对应数据位宽

C++基本运算

算术运算 + - \* / % 大数据不溢出，小数据不损失

算术赋值 = += -= \*= /= %=

自增自减 ++ --

条件 ?:

关系运算 > < >= <= == !=

逻辑 ! && ||

位运算 << >> ~ & | ^

常数参与具体运算时，需要具体告知具体数据类型

如sum = din + din\_t(0.25);

**高效的test bench**

Testing workbench

验证正确性

En.wipipedia.org/wiki/Test\_bench

Driver/Stimulus（输入激励，即测试数据）

--Reference Model（参考模型，即正确结果（good value））

--DUT（Design Under Test） 对top函数进行调用

--Monitor

--Scoreboard 与good value进行比较，记录和输出结果

C test bench

验证C函数正确性（Csim）

验证RTL设计（C/RTL Cosimulation）

如何高效

可被多次执行--遍历更多可能性

输出可以与good value进行比较

Main返回值

0：正确

非0：错误

Stimulus，测试激励

直接定义和初始化变量、存储在数组中、从文件中读取

适用于 Vitis 内核流程的接口

Vitis 内核流程为已编译的内核对象 (.xo) 提供支持，以便从主机应用和赛灵思的 Xilinx Run Time (XRT) 来执行软件控制。如 Vitis 统一软件平台文档中的[内核属性](https://docs.xilinx.com/access/sources/dita/topic?resourceid=fiv1568160307462.html&Doc_Version=2022.2 English&url=ug1393-vitis-application-acceleration" \t "https://docs.amd.com/r/2022.2-%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87/ug1399-vitis-hls/_blank) 中所述，此流程具有非常具体的接口要求，Vitis HLS 必须满足这些要求。

Vitis HLS 支持多种存储器、串流和寄存器接口范例，其中每个范例都遵循某个接口协议并使用适配器来与外部世界进行通信。

·存储器范例 (m\_axi)：内核通过存储器（如 DDR、HBM、PLRAM/BRAM/URAM）来访问数据

·串流范例 (axis)：数据从其它串流源（例如，视频处理器或其它内核）串流至内核中，也可从该内核流出。

·寄存器范例 (s\_axilite)：内核通过寄存器接口来访问数据，软件则通过寄存器读/写来访问数据。

用户通过Vitis HLS将C/C++翻译成HDL，最终映射成FPGA内部的LUT、DSP及RAM资源等

Vitis HLS

第一个工程：xc7z020clg400-1

先执行C仿真，得到初步结果

再执行C综合，可以看到C代码被综合成RTL代码后，使用的硬件资源评估、性能评估等

然后跑RTL和C的联合仿真CoSimulation（dump trace选择port可以查看波形）

“Block-level IO Handshake”端口为握手端口信号

“SW-to-HW Mapping”为输入输出数组或变量端口，端口包含地址总线、使能信号、写使能信号、数据总线等

查看波形：

自动开启vivado

New Solution：

<https://blog.csdn.net/weixin_44007264/article/details/122949673>

右键myloop，插入PIPELINE

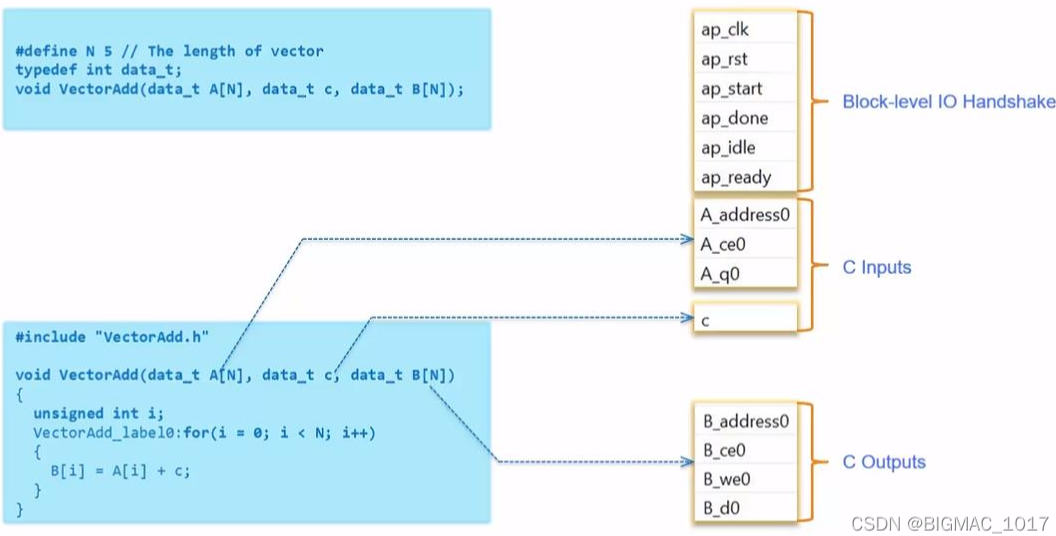
针对solution2进行综合

Project栏-Compare Reports对比两个solution的报告，Vitis HLS较新的版本中，即便没有设置directives，也会默认添加HLS PIPELINE指令，因此两个Report是相同的

对于每个C代码的For循环创建一个标签

Main正常执行返回0，错误返回1，便于判断仿真出错

C例程代码使用HLS综合后，生成的接口结构：



<https://blog.csdn.net/kzz6991/article/details/107405868>