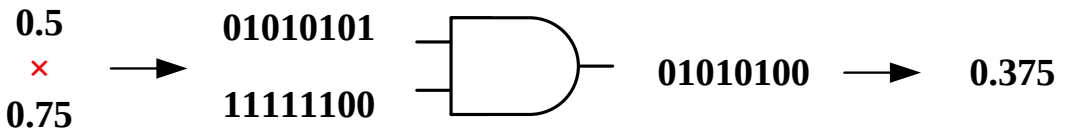
VLSI2021大作业1——基于随机计算的FIR滤波器的实现

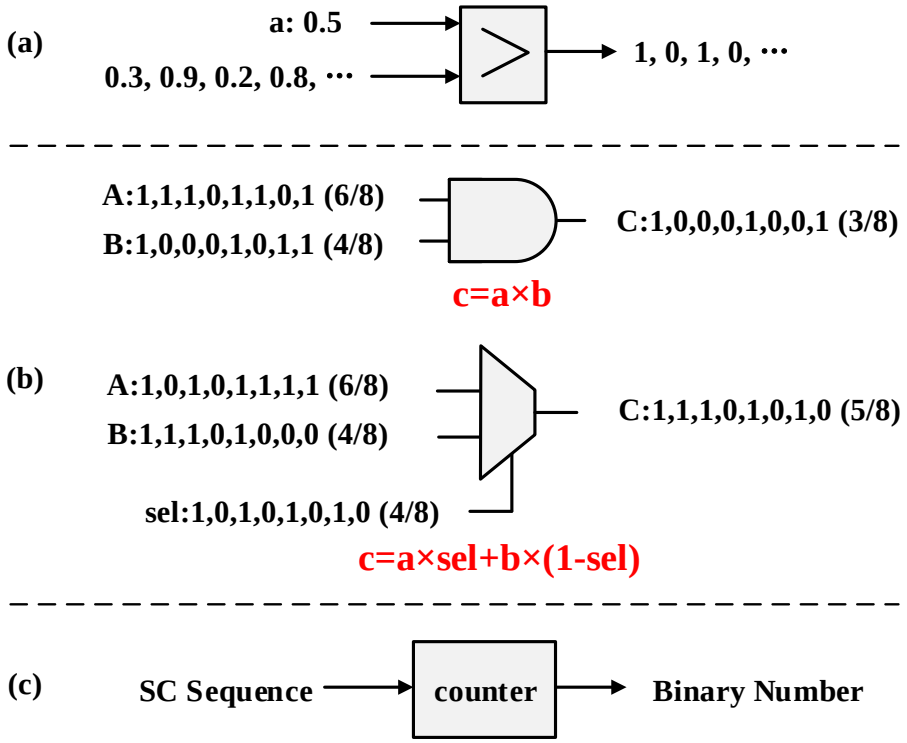
一、实验背景

1. 深度神经网络（Deep Neural Network，DNN）广泛地被应用在计算机视觉、自然语言处理、语音识别等各个领域，且取得了显著的成就。然而随着网络深度的增加，其计算量急剧增加，巨大的面积和功耗使得DNN在嵌入式设备中的应用变得不切实际。而且由于摩尔定律走到尽头，在传统领域解决面积功耗问题困难。因此，迫切需要新颖的替代计算范例克服这个障碍
2. 随机计算（Stochastic Computing, SC）则能解决此问题。SC是一种特殊的近似计算技术，可以从根本上简化硬件计算单元，从而有可能满足DNN的功耗和面积要求。SC将[0,1]范围内的二进制数转化为一串随机的0,1序列表示，序列中1所占的比例即为数的大小（但是序列的长度和1,0的位置都是不确定）。例如，序列{1,0,1,0}，{1,1,0,0}，{1,0,1,0,1,0,1,0}均可以表示数0.5。应用SC后，原来很复杂的运算可以由简单的电路来实现。例如下图说明了SC乘法的具体过程：通过将二进制数输入（0.5，0.75）转化为SC序列后，逐比特通过与门得到输出序列，根据与门性质，输出的序列表示的数（0.375）即为两数相乘的结果。因此乘法在SC中仅需要一个与门即可实现，从而大大降低面积和功耗。



二、随机计算基本原理

- 一个SC系统通常包括三个部分：序列生成单元，序列计算单元和序列转化单元。如下图所示，序列生成单元（a）将二进制数转化为SC序列，通常可以由比较器和随机数产生器（例如LFSR）得到；序列计算单元（b）在SC域进行各种运算：其中SC的乘法可以用与门实现，而带缩放的加法（例如c=(a+b)/2）可以用MUX实现；序列转化单元（c）将SC序列转化为二进制数，通过用计数器计算序列中有多少个1来实现。



VLSI2021大作业1——基于随机计算的FIR滤波器的实现

三、实验内容

- 在理解以上SC的相关知识后，运用SC设计8抽头FIR滤波器的计算，完成前仿真和综合。
- 顶层模块名称为sc_fir，接口信号包括：时钟clk，复位信号rst_n；输入数据data_in，输入信号有效data_in_valid，输入信号请求data_req；输出信号data_out，输出有效信号data_out_valid。
- 滤波器的抽头系数tap0~tap7给定且不变，均为[0,1)范围内的8位无符号数。为确保滤波器的输出在[0,1)之间，输出为归一化后的结果，即

$$data_{out}[n] = \frac{tap_0 \cdot data_{in}[7+n] + \cdots + tap_7 \cdot data_{in}[n]}{tap_0 + \cdots + tap_7}, \mathrm{n} = 0,1,\cdots$$

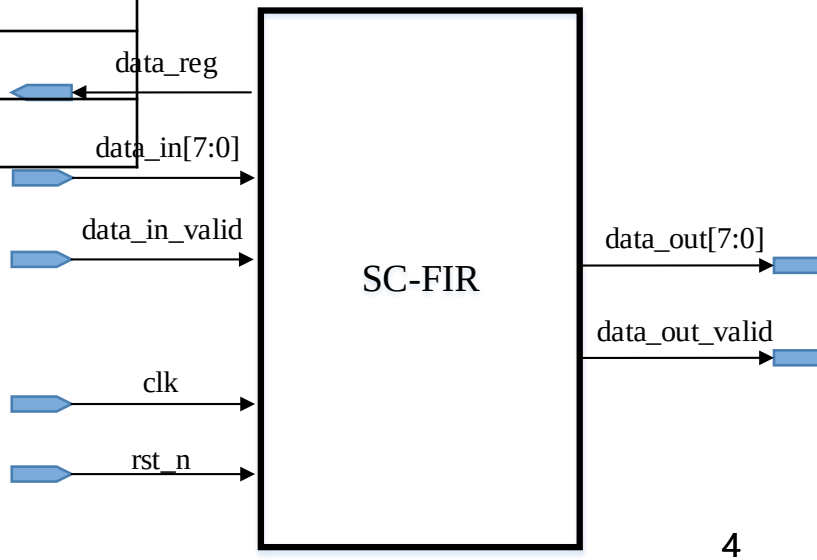
三、实验内容-软件实现

- 用高级语言验证设计符合平均误差的要求
- 输入文件data_in.txt：共1007行数据，每行数据为1个8bit的输入data_in
- 系数文件taps.txt：共8行数据，每行数据为1个8bit的系数tap
- 输出文件data_out.txt：共1000行数据，每行数据为1个8bit的输出数据data_out，输出数据的顺序与输入数据顺序严格对应
- 要求平均误差至少小于0.05，定义为平均误差=avg(abs(SC计算结果-精确结果))

三、实验内容-硬件实现

端口名称	比特宽度	I/O属性	说明
clk	1	I	主时钟
rst_n	1	I	复位信号，低有效，高电平系统正常工作
data_in	8	I	8位无符号数输入，数据在[0,1)的范围内
data_req	1	O	请求读数的信号，拉高后会在下个周期提供有效输入data_in
data_in_valid	1	I	输入数据data_in的有效信号
data_out	8	O	8位无符号数输出
data_out_valid	1	O	输出数据data_out的有效信号

$$data_{out}[n] = \frac{tap_0 \cdot data_{in}[7+n] + \cdots + tap_7 \cdot data_{in}[n]}{tap_0 + \cdots + tap_7}, n = 0,1,\cdots$$



VLSI2021大作业1——基于随机计算的FIR滤波器的实现

四、提示

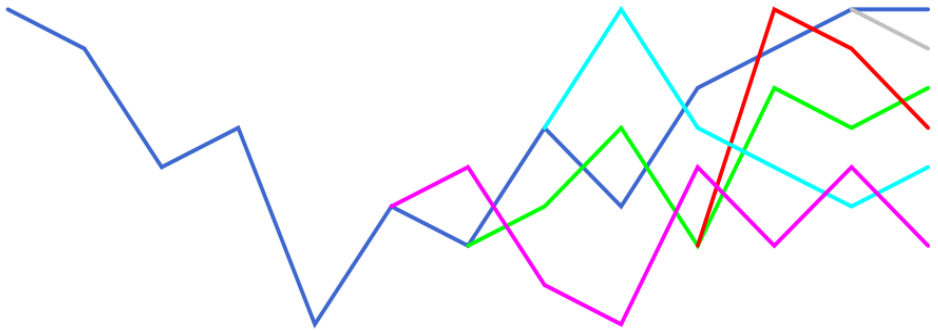
- 禁止使用除了SC乘法器以外的乘法器
- 可以预处理滤波器系数
- 在实现硬件代码之前，可能需要通过Matlab等高级语言验证设计符合平均误差的要求
- 完成sc_fir模块及所需的子模块，提供testbench，验证结果的正确性（平均误差至少小于0.05）
- 使用vivado综合选择v7系列vc709，给出综合后timing以及area信息，并给出关键评价指标面积效率 throughput/area
- **参考文献**:
- Sim H, Lee J. A new stochastic computing multiplier with application to deep convolutional neural networks[C]//Proceedings of the 54th Annual Design Automation Conference 2017. 2017: 1-6.

严禁任何形式的抄袭，提交请将实验报告、RTL源代码、testbench、C/python等算法实现、以“proj_学号_姓名”的命名格式打包成压缩包。
例如：proj_518021910xxx_张三；组队：proj_518021910xxx_张三_518021910xxx_李四。
截止时间：06月26日23：59之前（不允许迟交），作为附件发送至Canvas相应作业。

VLSI2021大作业2——维特比译码器的实现

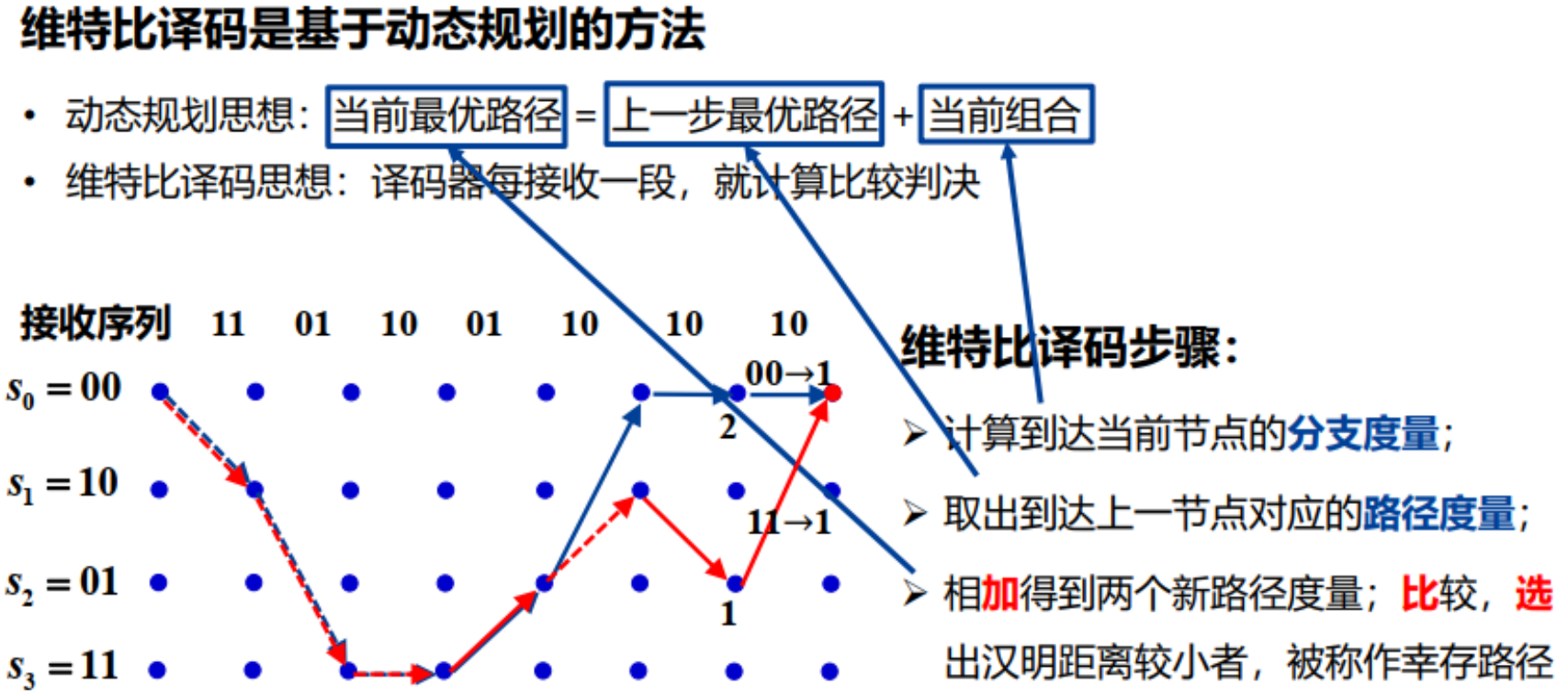
一、实验背景

- 在数字通信系统中，卷积码得到了广泛应用。在卷积码的译码中，为了避免对N个比特组成2N种可能序列路径做穷举，我们可以使用维特比译码的方法进行译码。该译码算法使用两个度量：分支度量和路径度量。分支度量计算的是发射和接收内容之间的“距离”，它是为每条分支路径定义的。在硬判决译码中，给出一组已经数字化的接收监督比特，分支度量就是监督比特预测值和接收监督比特之间的汉明距离。路径度量值与整个路径的每个节点有关。对硬判决解码，它对应于从初始状态到当前状态的路径与接收监督比特序列间的汉明距离。具有最小汉明距离的路径使误码最小化，并且当误码率较低时具有最大似然度。



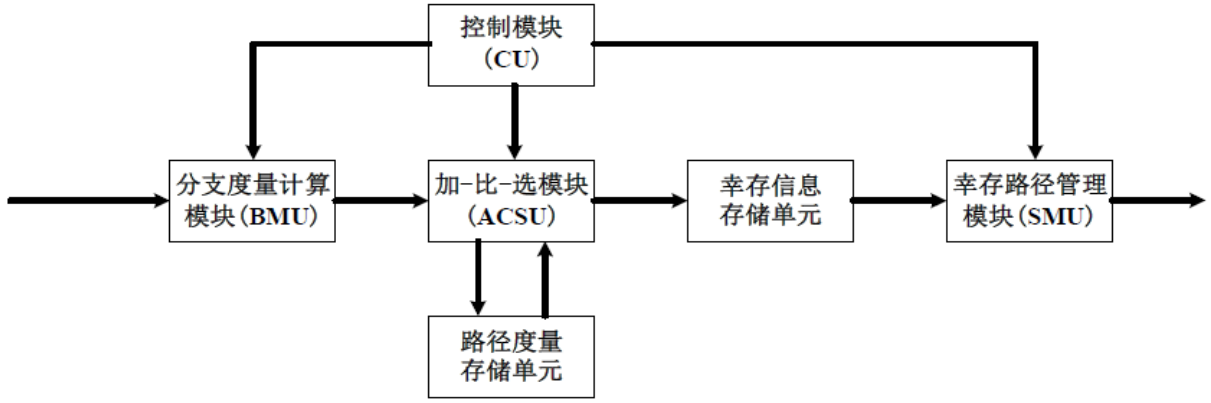
二、实验内容-软件算法

- 基于信道编解码实验，仿真验证**(2,1,3)**卷积码的维特比译码的功能，并使用Matlab截取维特比译码的输入和输出二进制序列用于硬件验证。实现的算法不做限制，但请尽量保持软件和硬件的一致性。维特比译码的要求符合IEEE 802.11a协议，采用硬判决，不需要实现凿孔。



二、实验内容- 硬件实现

- 可以参考课程中对于维特比译码器的设计进行实现，设计具体的硬件结构，能够进行连续的维特比译码。系统的整体框图如下所示，各个模块内部需要进一步设计。



- 通过Verilog代码实现，并利用Modelsim或Vivado验证电路的正确性，通过Vivado综合对性能进行评估。
- 要求如下：

a. 功能正确，尽量与软件结果一致；

b. 性能要求如下，使用Vivado综合时器件选取V7系列vc709，要求Vivado综合过后最大时钟频率能够达到**200 MHz**，利用所学的硬件设计知识，**尽可能优化吞吐性能**。

c. 评价标准：

1、吞吐率单位bps，即每秒处理的输入bit数量，参考文献有Mbps以及Gbps量级；

2、延时单位s，假设使用64QAM，延时代表完整处理一组64QAM对应bit的时间。

三、提示

- 所有的设计亮点请在报告中进行强调。报告内容至少包含以下：
  a. 硬件实现可行方案描述，完整的硬件设计原理框图；
  b. 分析硬件各个模块，给出各模块设计框图，关键路径；
  c. 各个功能模块的输入输出接口信号描述；
  d. 功能仿真的测试向量以及验证结果；
  e. 综合结果、关键路径（时序报告第一条路径）、面积或资源开销；
  f. 性能整理，请整理图表非截图，包括时钟频率，延时，吞吐率，面积或资源（LUT、DSP等）等；
  g. 有组队的同学给出分工情况。
- 参考文献：
- [1] I. Habib, Ö. Paker and S. Sawitzki, "Design Space Exploration of Hard-Decision Viterbi Decoding: Algorithm and VLSI Implementation," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 5, pp. 794-807, May 2010.
- [2] M. A. Anders, S. K. Mathew, S. K. Hsu, R. K. Krishnamurthy and S. Borkar, "A 1.9 Gb/s 358 mW 16–256 State Reconfigurable Viterbi Accelerator in 90 nm CMOS," in IEEE Journal of Solid-State Circuits, vol. 43, no. 1, pp. 214-222, Jan. 2008.

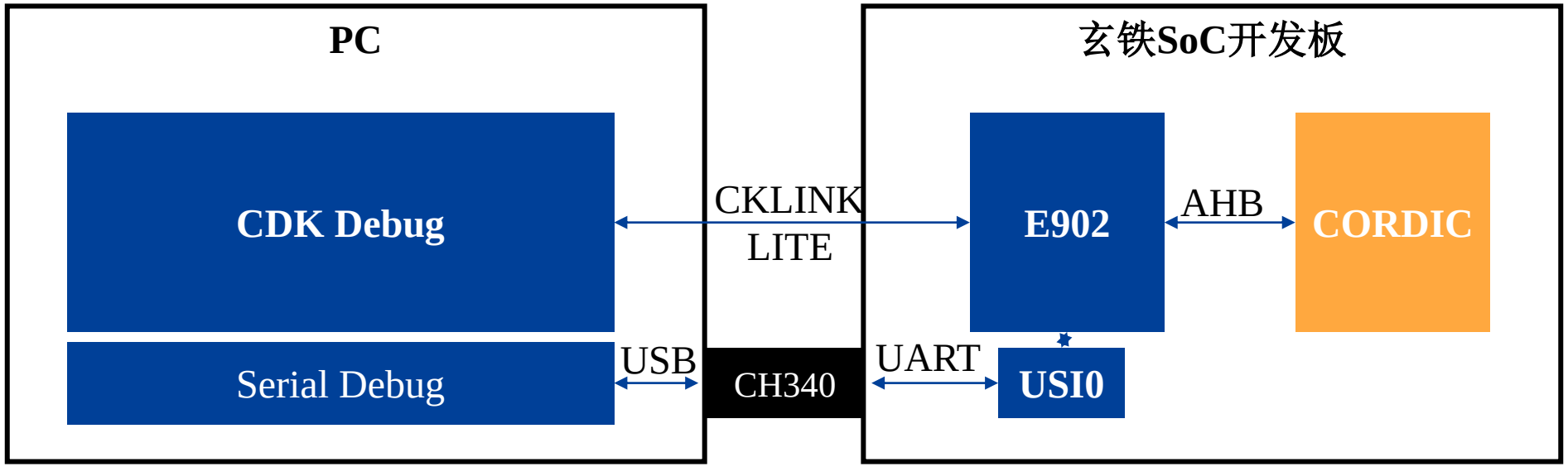
一、实验背景

1. RISC-V是UC Berkeley的David A. Patterson教授团队在2010年提出第五代精简指令集(RISC)。与其他指令集架构(ISA)不同的是，RISC-V是一个开源指令集架构，具有广泛的应用前景。RISC-V指令集完全开源，设计简单，易于移植Unix系统，模块化设计，完整工具链，同时有大量的开源实现和流片案例，已在社区得到大力支持。它虽然不是第一个开源的指令集(ISA)，但它是第一个被设计成可以根据具体场景可以选择适合的指令集架构。基于RISC-V指令集架构可以设计服务器CPU、家用电器CPU、工控CPU和传感器中的CPU。目前国内外已经有许多公司或者组织在基于RISC-V实现对应的SoC，比如SiFive、平头哥等等。本次实验中我们将使用平头哥的E902无剑SoC平台以及对应的FPGA开发板。平头哥设计了一套针对平头哥产品的集成开发环境CDK，能够实现软件程序的编写、编译、烧录上板、调试等等功能。

2. CORDIC（Coordinate Rotation Digital Computer）算法即坐标旋转数字计算方法，是J.D.Volder1于1959年首次提出，主要用于三角函数、双曲线、指数、对数的计算。该算法通过基本的加和移位运算代替乘法运算，使得矢量的旋转和定向的计算不再需要三角函数、乘法、开方、反三角、指数等函数。

二、实验目的

- 本次实验将完成基于RISC-V的CORDIC协处理器的设计与实现。完成相应的CORDIC协处理器硬件设计、软件程序编写、FPGA上板打印计算结果。
- 本次实验提供E902无剑SoC的SoC开源硬件代码、CDK集成开发环境、软件开发工具包以及相应的文档。



三、实验内容——硬件部分

- CORDIC协处理器的输入角度为16位有符号整数，采用角度制（例如90 = 90°）；输出的计算结果采用16位有符号定点数，补码表示，即1位符号位+7位整数位+8位小数位。支持输出不同的三角函数的计算结果：sin，cos，tan，arctan。输出的选择通过软件进行配置。
- CORDIC的架构既可以使用迭代架构也可以使用流水线架构，迭代次数为6次，补偿系数采用6次迭代时的补偿系数。
- CORDIC中特殊的三角函数可能用到的除法或乘法等计算单元可通过例化FPGA的IP实现。
- 将整个CORDIC协处理器挂载在E902无剑SoC上。具体的操作为将整个SoC Verilog代码中例化的main_dummy_topX模块替换成CORDIC协处理器，此时CORDIC的地址域即所替换的Dummy的从设备地址域。具体的地址分配详见wujian100_open Userguide v1.0.docx。

0x4001_0000~0x4001_FFFF	Dummy	64K	S7	main_dummy_top0
0x4002_0000~0x4002_FFFF	Dummy	64K	S8	main_dummy_top1
0x4010_0000~0x401F_FFFF	Dummy	1M	S9	main_dummy_top2
0x4020_0000~0x7FFF_FFFF	LSBUS	1024M-2M	S10	AHB LS BUS
0x8000_0000~0x9FFF_FFFF	Dummy	512M	S11	main_dummy_top3

三、实验内容——软件部分

- 编写不同的函数sin(),cos(),tan(),arctan()完成对CORDIC协处理器的配置和调用，并输出对应的计算结果。无剑E902 SoC采用RISC-V rv32ec指令集，不支持定点和浮点的直接表示，因此本次实验CORDIC的输出结果可以直接以整型进行打印输出

三、实验内容——软件编译与仿真

- 参考README.md中的仿真说明在workdir文件夹下调用脚本完成仿真（脚本会首先利用riscv工具链编译C程序，再调用verilog仿真器VCS进行仿真）

三、实验内容——FPGA上板调试

- 完成FPGA的综合实现，在平头哥玄铁FPGA开发板上烧录bit文件。
- 通过CDK连接FPGA开发板进行软件调试，在CDK中编译自己的软件，并进行调试，打印CORDIC的计算结果

三、评价指标

本次实验包括软件开发、硬件设计、上板调试，我们的主要评价指标住还要依据整个工作的完成度。

在报告撰写方面，请将所有的设计亮点在报告中进行强调。报告内容至少包含以下：

- 软件设计思路，可给出伪代码
- CORDIC完整的硬件设计原理框图，模块的输入输出接口信号描述；
- 功能仿真的测试向量以及验证结果；
- CDK工具使用以及FPGA上板打印结果；
- FPGA综合结果、面积或资源开销；
- (可选)性能整理，**请整理图表非截图，包括时钟频率，延时，吞吐，面积或资源等；**
- 有组队的同学给出分工情况。

严禁任何形式的抄袭，提交请将实验报告、RTL源代码、testbench、C/C++等算法实现、以“proj_学号_姓名”的命名格式打包成压缩包。

例如：proj_518021910xxx_张三；组队：proj_518021910xxx_张三_518021910xxx_李四。

截止时间：06月26日23：59之前（不允许迟交），作为附件发送至Canvas相应作业。

四、提示

1. 本次实验的仿真需要在linux虚拟机中完成（虚拟机已经安装的仿真工具为VCS），上板需要在windows下使用CDK来完成。虚拟机和软件安装文件的分享连接为：**https://jbox.sjtu.edu.cn/l/aFKkki (提取码：ickl)**
2. **实验的提供的源代码和文档都在虚拟机桌面的wujian_open文件夹中，虚拟机的密码为123456**
3. 选择该题的同学请务必在开始实验内容之前熟悉整套平头哥无剑SoC的开发环境以及技术文档，阅读wujian_open文件夹中的README文档，对环境的熟悉会减少后续的实验过程中可能碰到的问题。
4. 第一次打开虚拟机需要选择“**我已经移动该虚拟机**”，这是因为EDA工具的license与虚拟机的MAC地址绑定，选择“我已复制虚拟机”会改变虚拟机的MAC地址，导致license失效
5. 每一次开机需要在终端输入“**lic**”开启synopsys license管理进程，方可使用VCS，DVE