Lab1　设计一个FIR滤波器分离鸟类声音

目标

在完成本实验后，您应当学会：

- 如何使用Vitis HLS构建一个项目
- 在Vitis HLS中进行仿真、综合与IP导出

简而言之，您将掌握使用HLS进行加速核设计与部署到PYNQ的基本流程。出于篇幅限制，本实验仅介绍基本工具操作流程。

环境要求

- PYNQ-Z2远程实验室服务或物理板卡Vitis HLS
- Vivado

实验步骤（Linux版本的HLS）

1. 在Vitis_HLS中设计FIR

1.1 Linux环境设置（**如果您使用校内公共服务器平台的话，这些设置都已配好，直接拷贝即可**）

1. 设置以下信息，比如bash环境下修改自己根目录下的**.bashrc文件**（路径请根据自己情况修改）

   ```
   # Xilinx License

   export XILINXD_LICENSE_FILE=/mnt/home/software/vivadoML21.2/Xilinx2037.lic

   # HLS basic settings

   source /mnt/home/tools/xilinx/Vitis_HLS/2021.2/settings64.sh

   # PATH setting
   ```

export PATH=$PATH:/mnt/home/tools/xilinx/Vitis_HLS/2021.2/lnx64/tools/clang/bin

Library settings

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/mnt/home/tools/xilinx/Vitis_HLS/2021.2/lib/lnx64.o:/mnt/home/tools/xilinx/Vitis_HLS/2021.2/lib/lnx64.o/Default

2. 测试自己的环境是否已经OK

输入以下命令，看能否找到：

which vitis_hls

1.2 代码脚本设置

新建一个目录　　mkdir FIR

进入该目录　　cd FIR

将目标C文件拷贝到该目录下　　cp -r /mnt/home/students/zhaokang/Lab/ ./

用vim新建一个TCL脚本文件，命名为“run.tcl”，将以下内容输入并保存。

```
# 1. project name
open_project prj -reset

# 2. set top name
set_top fir_wrap
# 3. add files
add_files fir.cpp
```

```
add_files -tb fir_tb.cpp

# 4. solution name
open_solution "solution1" -reset

# 5. device name
set_part virtex7

# 6. clock setting
create_clock -period 13 -name default
config_export -version 2.0.1;

# 7. C-simulation
csim_design
# 8. synthesis
csynth_design
# 9. co-simulation
cosim_design
```

1.3 执行HLS，生成RTL硬件描述

在该目录下，执行命令：　　vitis_hls run.tcl

则会自动执行c-simulation, synthesis, co-simulation等过程，并生成日志文件vitis_hls.log，同时输出到屏幕。

1.4 查看HLS的生成结果

（1）　查看LOG日志

vim vitis_hls.log

你会看到，HLS先执行c-simulation，并输出结果；之后是详细的综合（synthesis）过程，既包括前端编译结果，也包括后端的调度和资源绑定结果；最后是co-simulation结果。

（2）　查看生成的RTL文件

cd prj/solution1/syn/

ls

你会看到，除了报告report外，生成了verilog和VHDL两种RTL硬件描述。

（3）　查看综合synthesis报告，评估其性能

cd prj/solution1/syn/report/

打开顶层module名字对应的rpt报告文件，比如该顶层文件名叫“fir_wrap”，则打开fir_wrap_csynth.rpt：　　vim fir_wrap_csynth.rpt

可以看到其预估的timing值、latency值：

```
== Performance Estimates
================================================================
+ Timing:
    * Summary:
    +--------+----------+----------+------------+
    |  Clock |  Target  | Estimated| Uncertainty|
    +--------+----------+----------+------------+
    |ap_clk  |  13.00 ns|  9.490 ns|     3.51 ns|
    +--------+----------+----------+------------+

+ Latency:
    * Summary:
    +---------+---------+----------+----------+-----+-----+----------+
    |  Latency (cycles) |  Latency (absolute) |  Interval  | Pipeline|
    |   min   |   max   |    min   |    max   | min | max |   Type   |
    +---------+---------+----------+----------+-----+-----+----------+
    |        ?|        ?|         ?|         ?|    ?|    ?|        no|
    +---------+---------+----------+----------+-----+-----+----------+
```

也可以看到预估的各类资源的使用量：

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+-----------------+---------+------+--------+--------+-----+
|       Name      | BRAM_18K|  DSP |   FF   |   LUT  | URAM|
+-----------------+---------+------+--------+--------+-----+
|DSP              |        -|     -|       -|       -|    -|
|Expression       |        -|     -|       -|       -|    -|
|FIFO             |        -|     -|       -|       -|    -|
|Instance         |        4|   297|    5686|    6813|    -|
|Memory           |        -|     -|       -|       -|    -|
|Multiplexer      |        -|     -|       -|     270|    -|
|Register         |        -|     -|    3400|       -|    -|
+-----------------+---------+------+--------+--------+-----+
|Total            |        4|   297|    9086|    7083|    0|
+-----------------+---------+------+--------+--------+-----+
|Available        |     1590|  1260|  728400|  364200|    0|
+-----------------+---------+------+--------+--------+-----+
|Utilization (%)  |       ~0|    23|       1|       1|    0|
+-----------------+---------+------+--------+--------+-----+
```

（4）　查看更精确的性能latency结果，即co-simulation结果

cd prj/solution1/sim/report/

vim fir_wrap_cosim.rpt

可以看到精确的latency值：

Latency(Clock Cycles)		
min	avg	max
NA	NA	NA
442	442	442